Performing a High Realistic Video Game Environment

Razvan Alexandru Tocitu

Final Degree Work
Bachelor's Degree in
Video Game Design and Development
Universitat Jaume I

May 25, 2022

Supervised by: Inmaculada Remolar Quintana.

Abstract

One of the most important parts of video game development is the art style of the game. There are different types of art, from cartoon to realistic. In this case we use the realistic style which is characterised by its closeness to reality. Thanks to the approach to reality that this type has, it facilitates the immersion of the player in the video game.

Therefore, in this Final Degree Project a video game will be made based on an Escape Room using these high realistic graphics, developing different 3D elements for it with a high quality graphics to engage the player to enjoy an immersion in the video game. The video game consists of an Escape Room, that is to say, a video game in which the player will be trapped in a room and will try to escape by solving 4 different puzzles to unlock the main exit. In addition, the game will have a variety of mechanics that the player will have to use to solve the different puzzles that will be found throughout the game play. It will be a first-person adventure in which the protagonist will have no memory of how he/she ended up locked in that room, being forced to investigate what happend with the clues in the room.

Contents

List of Figures

CHAPTER 1

INTRODUCTION

Contents

1.1 Work Motivation

The development of the project is based on the ambition to recreate a realistic space for a video game. This has its origin in the desire to study the needs and demands that are expected on the way to achieve this visual quality. In addition to this, it is desired to learn about the different game engines that are currently used in the industry and the capacity of each one when calculating the lighting.

1.2 Objectives

The main objectives to be achieved in the project are as follows:

- Achieve High Realistic Quality.
- Learn About Video Games Engines.
- Learn how to create Visual Effects in video games.
- Put into practice modelling skills acquired during the degree course.

The basic objective of the project is to create an original space that aims to achieve a high visual quality. In order to do this, we first have to study and choose a game engine that meets the requirements needed for the project.

Once the engine has been chosen, we have to work on the creation of the scenario. This involves personally modelling the vast majority of elements throughout the game, from interface elements, main level elements and scene decoration. Of course, after modelling each element, it has to be textured to achieve a realistic finish.

Once the modelling of the scene is completed, the scene has to be assembled in the game engine. Then the player has to have mechanics such as movement and interaction with objects in order to solve the puzzles he has to face, as well as writing the code for the puzzles that are scattered around the scene.

1.3 Related Subject

This project is related to several subjects taught in the degree course. Any subject related to programming can be linked to this project through the way of approaching the different related problems such as moving the character, creating the different classes needed, etc.

On the other hand, the main objective of this final degree project is to acquire as much realism as possible from the student's own models. Therefore, other related subjects are mainly **VJ1204-Graphic Expression**, where we get a first contact with the modelling program Blender and **VJ1216- 3D Design**.It will also require the knowledge acquired in the subjects **VJ1204-Artistic Expression** and **VJ1223-Video Game Art** in which we learn how to form a composition when choosing colours and themes so that the scenario has a unitary sense.

When developing the narrative of the game, we have used the knowledge acquired in the subject **VJ1222-Conceptual Design of Video Games**, in which we learn all the preparations prior to the development of the narrative of a game.

Finally, we mention **VJ1227-Game Engines**, in which we learn what a game engine is, and what differentiates one from the other. Being an important part due to having to analyze several game engines to choose the best among them with which you can achieve greater visual realism.

1.4 Environment and Initial State

The project starts from scratch. This involves spending time choosing a game engine and modelling the environment. Therefore it will be necessary to use the game engine, 3D modelling software such as Blender [1] and 3DSMax [7]. The project has to have a programming part which is required as part of the final degree project, therefore the idea was born not only to recreate a realistic space but to create a video game from it. The conclusion reached was to create an Escape Room [2] style game in which the player will be forced to examine the environment and in this way give importance to the visual quality of the game.

Chapter 2

Planning and resources evaluation

Contents

"Those who plan do better than those who do not plan, even should they rarely stick to their plan." [10]

2.1 Planning

In this section I will draft the biggest sections that will be necessary for the development of the project. Each section will consist of a block of work and a Gantt (Figure 2.1) diagram will be attached to see the whole of it together with the expected time for its conclusion.

The Bachelor Final Project should take around 300 hours including creating the game, the report and the exposition. In the following table there is an estimated value of the hours that it will take for each part of the project.The total amount of hours is expressed in the following table.

Work to be performed	Expected amount of time
Studying Video Game Engines	10h
Model environment	70h
Create magic VFX	60h
Learn Unreal Engine 4 basics	30h
Code the basic movement	50h
Testing and error fixing	30h
Writing the project report	40h
Preparing exposition	10h

2.1.1 Study Video Game Engine

The first step to consider in the project is to analyse the different engines that can be used and which of them best suits the final objectives. It is mainly a decision between Unity [3], Unreal Engine 4 and Unreal Engine 5 [4]. After choosing the game engine, it is also necessary to choose a 3D modelling software that can accommodate the needs of the project, being a decision between Blender[1] and 3DS Max[7].

2.1.2 Model Environment And VFX

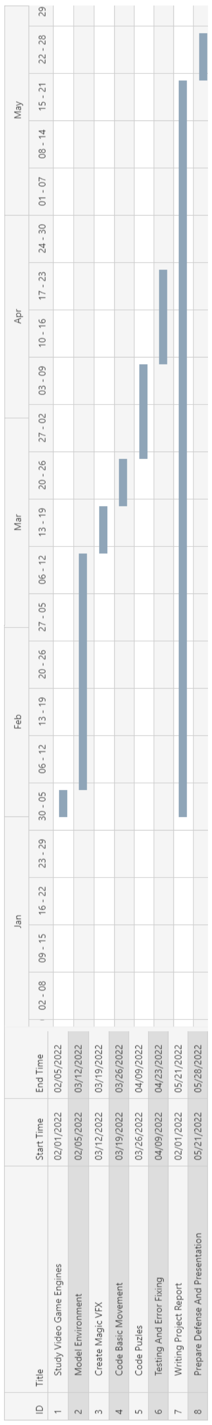
At this stage of the project, all the elements of the video game must be modelled in 3D. These elements range from those necessary to complete the game and with which the player must interact with them to purely decorative elements to give a greater realism to the visual environment.

It is desired to introduce special effects, also known as VFX[1] related to magic in the project, since in the game story is related to the magic ambience. For this purpose, it is planned to spend some time studying and learning through online research the method of working with these effects. This will be developed using the tools offered by Unreal, specifically a particle system called Niagara Particle Effect [12].

2.1.3 Coding And Solving Errors

For the functioning of the project, the code that allows the functioning of the mechanics that will be in the game has to be created. This entails basic mechanics such as character movement and interaction with objects to the code for the inner workings of each of the puzzles that must be solved to complete the game. In this part we have to study the way of programming offered by the chosen game engine and I have to adapt myself to it, since not all the existing game engines are programmed in the same language.

[1] *Visual Effects*



ID	Title	Start Time	End Time
1	Study Video Game Engines	02/01/2022	02/05/2022
2	Model Environment	02/05/2022	03/12/2022
3	Create Magic VFX	03/12/2022	03/19/2022
4	Code Basic Movement	03/19/2022	03/26/2022
5	Code Puzles	03/26/2022	04/09/2022
6	Testing And Error Fixing	04/09/2022	04/23/2022
7	Writing Project Report	02/01/2022	05/21/2022
8	Prepare Defense And Presentation	05/21/2022	05/28/2022

Figure 2.1: Estimated time over the months to complete the project.

2.1.4 Work Report

During the project development process, a report, i.e. this document, will be developed in parallel, detailing the process and all interesting and important details to be emphasised and considered relevant.

2.2 Resource Evaluation

These are the Softwares that have finally been used for the development of the project and the use that has been made of each of them

- **Blender [1]:** This design software has been used as the main modelling resource with which almost the entire project has been modelled.
- **Autodesk 3DS Max [7]:** This design software has been used as a support for Blender. Especially at the time of the UVs in certain models I have used it as support simply to be able to check the correct state of these.
- **ZBrush [14]:** This software used this software specialised in modelling by means of the spitting technique to give some light touches to a specific model. This will be detailed later in its corresponding section.
- **Unreal Engine 5 [4]:**Game engine chosen for the main development of the whole project.
- **Adobe Photoshop [8]:** Used to develop mainly the different textures used in the 3D models and the adjustment of these to their corresponding UV's.
- **Adobe Illustrator [6]:** Used for the creation of interface elements. This has been used instead of Photoshop because I have considered that it makes my work much easier thanks to the primitive figures that it offers and the vectorial design of elements with which I can work without losing quality of the elements.

Chapter 3

System Analysis and Design

Contents

3.1 Game Design

3.1.1 Game Concept

The story of a video game helps the player to become fully immersed in it. After studying various ideas, this resulted in a story of magic represented as a synopsis of the project:

You just remember being quietly in your tower, resting and doing your magic studies. But when night falls, something strange happens and you wake up in a room you don't recognise. You try to open the door but, you realise that it is sealed, you are trapped here. No matter how or why, right now there is only one thing on your mind: Escape from this place.

The game as explained above is a escape room style game. These types of games consist of escaping from a room by solving different riddles and puzzles to do so. Some games of this style can be for example *The Witness* [13] or *The Room* [9].In the video game the player will be trapped in a room without a clue. Therefore, he will have to explore the room to locate the four different puzzles. Solving each of these puzzles will destroy a lock on the main door. Once all four locks have been unlocked, the player will have completed the game, allowing them to interact with the main door and finish the game.

3.1.2 Game Art

The art of the video game will be inspired by a mix of fantasy with fictional touches of magic and the style of romanticism. Therefore, decorations similar to the aforementioned style will be used to set the project in that style. The magical feature of the environment will be reflected when observing elements in the environment such as magic circles, crystal balls, etc.

3.1.3 Game Controls

- **W Key** : Walk forwards.
- **A Key** : Walk right.
- **S Key** : Walk backwards.
- **D Key** : Walk left.
- **E Key** : Interact Button 1.
- **F Key** : Interact Button 2.
- **ESC Key** : Close Menu.
- **Mouse**: Move Camera / Move Object.
- **Mouse Left Click** : Grab Object.

3.2 Requirement Analysis

In order to develop the project we will need several elements. First of all we will need to choose a game engine in which to develop the project. For this, after an analysis of the characteristics of each one, I have opted for Unreal Engine 5. I have picked Unreal because, unlike Unity, it has a more powerful algorithm for calculating the lighting, and with it we achieve a closer approximation to the reality we want to obtain. On the other hand, I have not chosen Unreal Engine 5, even though it is more powerful than its previous version, due to lack of documentation because of its recent release.

On the other hand, 3D modelling software will be needed. In this case, depending on the needs, I will use Blender or 3DS Max. As main software I will use Blender, but

3DS Max libraries can be very useful for modelling. Therefore, I will alternate between these 2 programs.

When it comes to programming I will have to start learning from scratch about how to use Unreal Engine's Blue Prints [4]

3.2.1 Functional Requirements

Input:	The player will have to solve a puzzle to unlock each lock of the main door to escape. There will be 4 locks on the door.
Output:	The player will have to solve a puzzle to unlock each lock of the main door to escape. There will be 4 locks on the door.
The player will have to solve a puzzle to unlock each lock of the main door to escape. There will be 4 locks on the door.	

Table 3.1: Functional requirement «Unlocking Main Door»

Input:	The player will press a specific keyboard keys to move the character around the map.
Output:	The player will press a specific keyboard keys to move the character around the map.
The player will press a specific keyboard keys to move the character around the map.	

Table 3.2: Functional requirement «Movement»

Input:	The player will press a specific keyboard keys to interact with specific objects. These interactions can be classified as a "Clicking" and "Holding" a key to either activate and object functionality or grab it.
Output:	The player will press a specific keyboard keys to interact with specific objects. These interactions can be classified as a "Clicking" and "Holding" a key to either activate and object functionality or grab it.
The player will press a specific keyboard keys to interact with specific objects. These interactions can be classified as a "Clicking" and "Holding" a key to either activate and object functionality or grab it.	

Table 3.3: Functional requirement «Interact with Environment»

Input:	The player will be able to check his progression on the interface. There will be marks that show the total spells he managed to achieve to open the main door .
Output:	The player will be able to check his progression on the interface. There will be marks that show the total spells he managed to achieve to open the main door .
The player will be able to check his progression on the interface. There will be marks that show the total spells he managed to achieve to open the main door .	

Table 3.4: Functional requirement «Progression Marks»

Input:	The player will be able to use the mouse for a puzzle to move and drag objects to place them in their correct position.
Output:	The player will be able to use the mouse for a puzzle to move and drag objects to place them in their correct position.
The player will be able to use the mouse for a puzzle to move and drag objects to place them in their correct position.	

Table 3.5: Functional requirement «Mouse Controls»

3.2.2 Non-functional Requirements

The main non-functional requirement is the polygon loading of the 3D models in the project, therefore the total cost that can generate these polygons has to be taken into account in order to obtain a smooth result that does not cause the game to run slowly. The development process of each object in the project is represented in the following diagram (Figure 3.1).

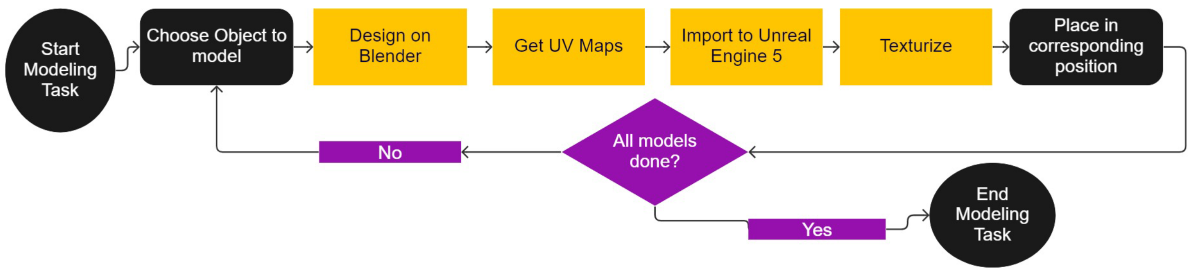


Figure 3.1: Modeling working process.

3.3 System Design

The main activity flow of the video game consists of a first phase where the player searches for one of the puzzles in the room. Once he has found it, he has to try to solve it. If the player does not succeed, he/she tries again until the puzzle is solved. Once the puzzle is completed, the seal corresponding to the puzzle located on the main door will be destroyed. In order to complete the game you need to solve all the 4 puzzles and open the main door to escape from the room. In this diagram (Figure 3.2) you can see the flow of activity in the game.

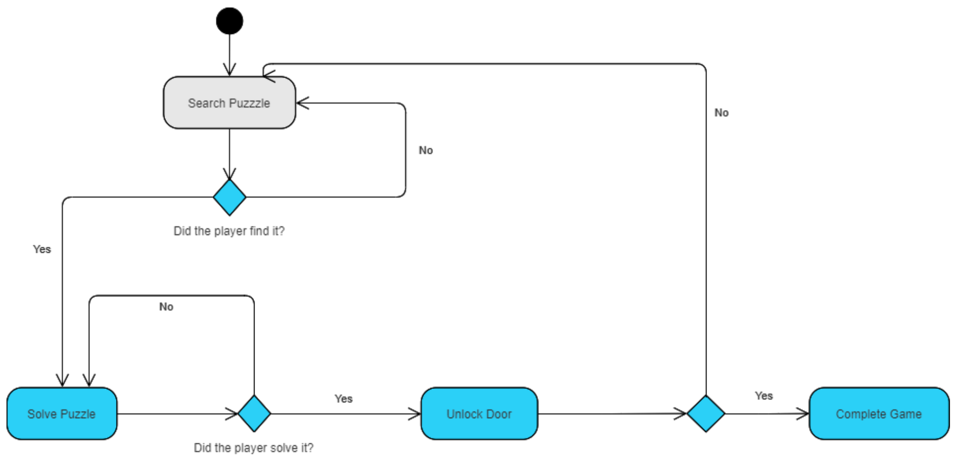


Figure 3.2: Activity diagram of the main cycle in the game.

3.4 System Architecture

Minimum system requirements, drawn by comparison after studying different similar games when it comes to the level of realism. Also, since the project does not need to be installed on the computer as it is an application, the minimum memory required in this case is the memory occupied by the project itself.

- OS: 64-bit Windows 7 or 64-bit Windows 8 (8.1).
- Processor: Intel CPU Core i5-2500K 3.3GHz, AMD CPU Phenom II X4 940.
- Graphics: Nvidia GPU GeForce GTX 1050 or AMD GPU Radeon HD 8970.
- RAM: 6GB.
- Disk space: 10 GB.

3.5 Interface Design

In recent years many big games have opted for minimalism in their interface design. This is because in many cases the interface can be distracting to the player and by cleaning up the screen with a minimalist interface it forces the player to admire the environment in a certain way. Because of this, and taking into account that we want to develop an Escape Room, I have also opted for a minimalist interface. This will consist of a simple white dot in the centre of the screen so that the player can locate the point the player is looking at. Initially it was intended to leave the screen without this dot, but after testing the game several times I came to the conclusion that this dot is necessary to help the player's orientation, especially in one of the puzzles that will be described in one of the following sections.

Also as part of the interface I have integrated some messages on the screen following the example of games like *Call of Duty : Vanguard* (Figure 3.4) or *Rise of The Tomb Rider* (Figure 3.3) an indicator of the key that the player has to press or keep pressed to be able to interact with the elements of the environment.

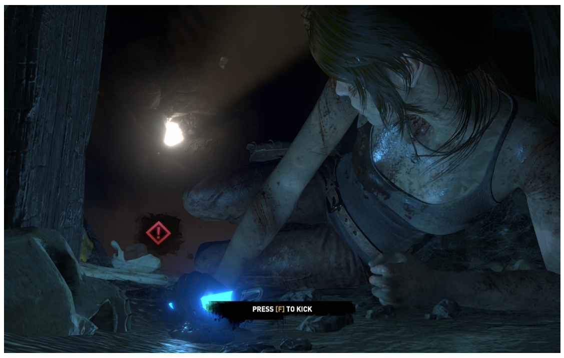


Figure 3.3: The Rise Of The Tomb Rider.

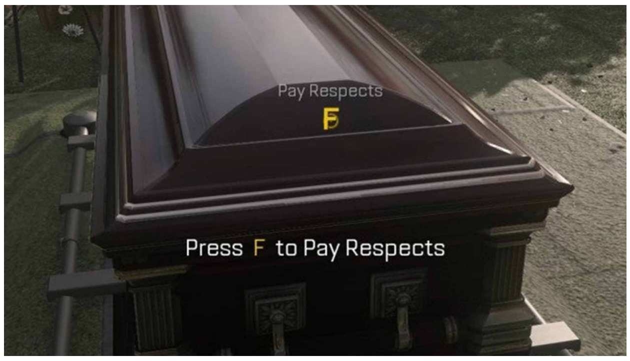


Figure 3.4: Call Of Duty: Vanguard.

Chapter 4

Work Development and Results

Contents

The project can be divided into 6 development blocks, which will be analyzed below. Each block corresponds to the tasks that have been planned in the Gantt chart (Figure 2.1) inserted previously.

4.1 Work Development

4.1.1 Study Game Engines

At the beginning of the project, the first thing to analyze was which game engine to use. Unity had been discarded in the first place because, although it is one of the best game engines on the market and is still used by a large number of companies, it is more specialized in cartoon graphics. This does not mean that its process of calculating the lights is bad, in fact, this is a great point in favor of the game engine, but when compared to other more powerful game engines, this one is left behind.

Unreal Engine, on the other hand, is an engine that, like Unity, is used in many video games due to its great power of light calculation. This engine is used for Triple A[1] games such as *Kingdom Hearths III, Sea Of Thieves, etc* [5]. Usually the learning curve of Unreal Engine is more complicated compared to Unity but it must be taken into account that the visual quality of this engine is notoriously superior to that of Unity. Therefore the first version of the project was created in Unreal Engine 4.

After a month of development, the Epic Games platform published an update called Unreal Engine 5, being this an engine similar to the previous one at the time of the interface and the way of use. The big difference that this one brought was the Lumen and Nanite technology, two technologies that I will explain below and for which I finally decided to dedicate a day of work to migrate all the progress I got to this new engine. Below is an analysis of the advantages and disadvantages between Unreal Engine 5 and Unreal Engine 4.

Points of interest	Unity Engine	Unreal Engine 4	Unreal Engine 5
Available information	A lot information available	A lot information available	Less information available
Graphics Specialised	Cartoon	Realistic	Realistic
Lightning	No Dynamic Lighting with Ray Tracing	Dynamic Lighting with Ray Tracing	Dynamic Lighting. combined with Lumen and Nanyte technology

Table 4.1: Analysis table of each video game engine.

Lumen implements real-time indirect lighting with pretty amazing precision, and emissive lighting contributes with Global Illumination. So if you have a big light material, it will light up your scene. Lumen also provides reflections and integrates Global Ilumination into reflections. This was not possible with the previous lighting calculation version of Unreal Engine 4 which used Ray Tracing. In addition, Unreal Engine 5 not only relies on lumen technology, but also implements the help of Nanite technology which works with internal meshes for the rendering of object details, so that only what is visible to the player is rendered.

4.1.2 3D Modeling Process

Before starting the modelling process we needed to be clear about the concept that the project should convey. After analysing several games, researching about magic and

[1] *Informal classification used to categorise games produced and distributed by a mid-sized or major publisher, which typically have higher development and marketing budgets than other tiers of games.*

fantasy environments, we came to the conclusion that the most suitable for this project was to create the room of an alchemist.

From the beginning, after arriving at this conclusion, we had a mental image of how we wanted the room in which the video game would take place to look like. However, a simple and small inspiration (Figure 4.1) was developed and used to reinforce the ideas and aesthetics. The function of the inspiration board was rather to observe small aesthetic details that were used later to decorate the room and give it a less artificial appearance.

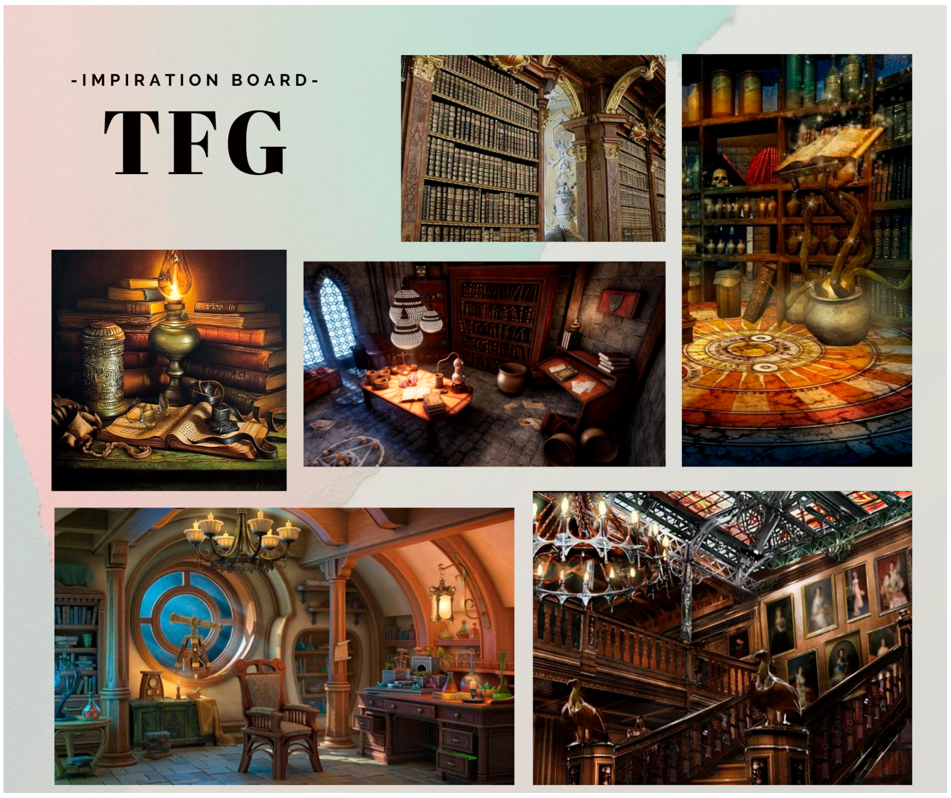


Figure 4.1: Inspiration Board used on the project.

House Base

Now that the idea was clear, the modelling process began. I started with the walls of the room. I wanted to create a modular model, that is to say, a simple fraction of a wall that could be cloned when assembling a room. Finally a satisfactory result was achieved, which could be linked together without the cut between the individual wall blocks being noticeable. All this process was done in Blender, as it was a simple model and the only difficulty it had was the fact of being able to concatenate one with the other. I solved this problem using the Blender Mirror modifier (Figure 4.2). This modifier allows us to make an inverted copy in one of the axes making the mirror effect. Therefore all the vertices of the ends, had the same height and the same finishing point, allowing to concatenate several units of this model without being perceivable the cut between one and another.

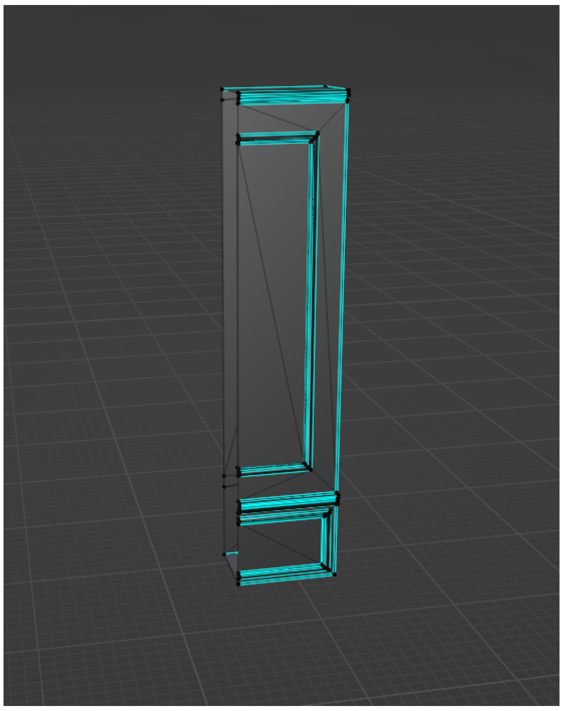


Figure 4.2: Half Unity Wall without Mirror Modifier.

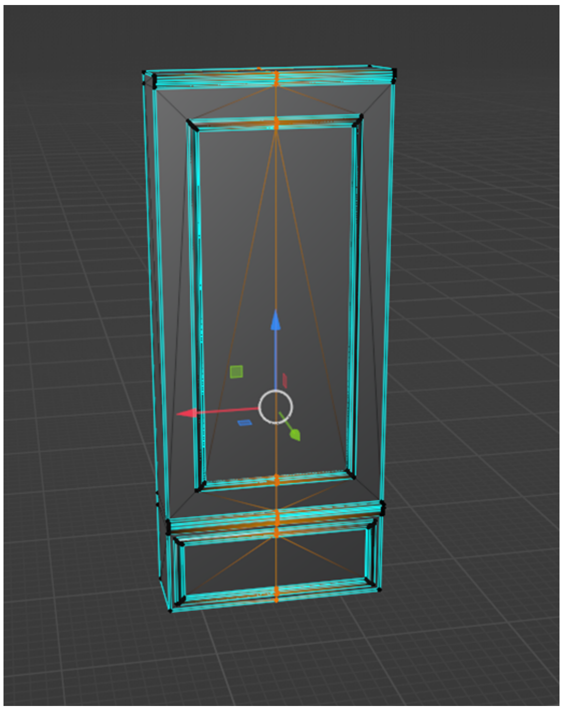


Figure 4.3: Half Unity Wall with Mirror Modifier applied.

The next step in modelling is the texturing of these walls. This process is identical for all 3D models. The first step is to select certain edges to declare them as seams. These edges are the cutting points of the UV's where they are separated when generating the maps. The map generation can be done by hand or automatically. In the case of this project, it is started from the automatic UV's map (Figure 4.7) and then it is studied carefully to correct errors such as texture directions, scaling, etc. The next step is working on Photoshop, where the UV map are imported and placed on it the textures that I want to be captured at this point. Normally models can have more than one material, so it is often not necessary to capture all the elements of the model in a single texture. Therefore in Photoshop, it is only ensured that the most visible parts of the models are placed in the correct position.

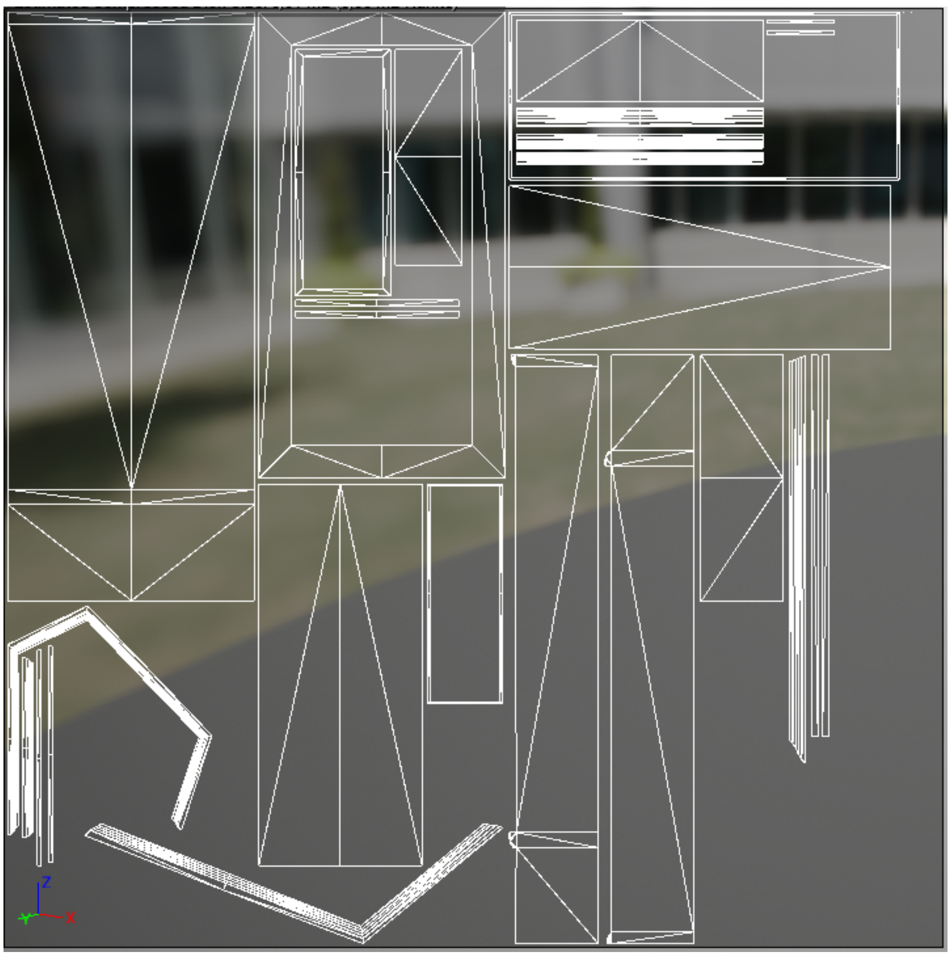


Figure 4.4: Unreal UV's map inside Unreal Editor from wall Plane Decoration

In this model, a texturing map called Alpha is also used (Figure 4.5). This map delimits the elements that are rendered. In the case of Unreal Engine materials, the parts of the material where the Alpha texture is black are rendered and the white colour of the Alpha texture is not rendered.

The second modelling step was dedicated to modelling the 4 puzzles that were in the game as they are the main elements of the game. However, the same modelling process has been used for all the objects in the project.

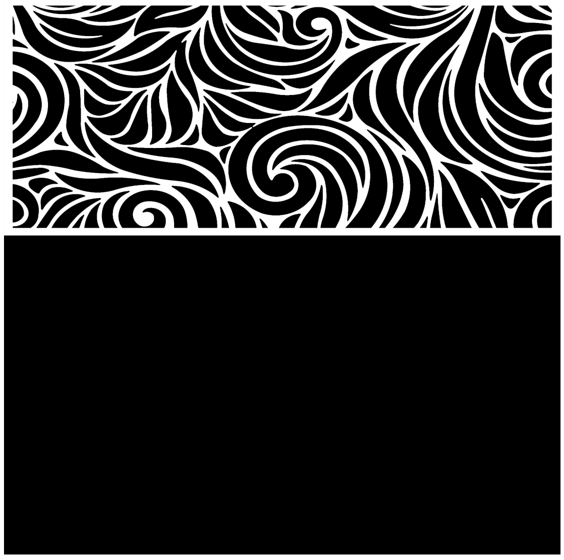


Figure 4.5: Alpha Map used on Unreal from Plane Decoration.

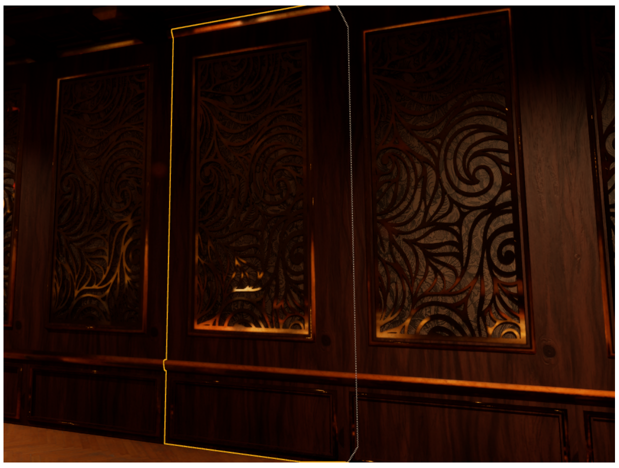


Figure 4.6: Final wall result in Unreal Engine.

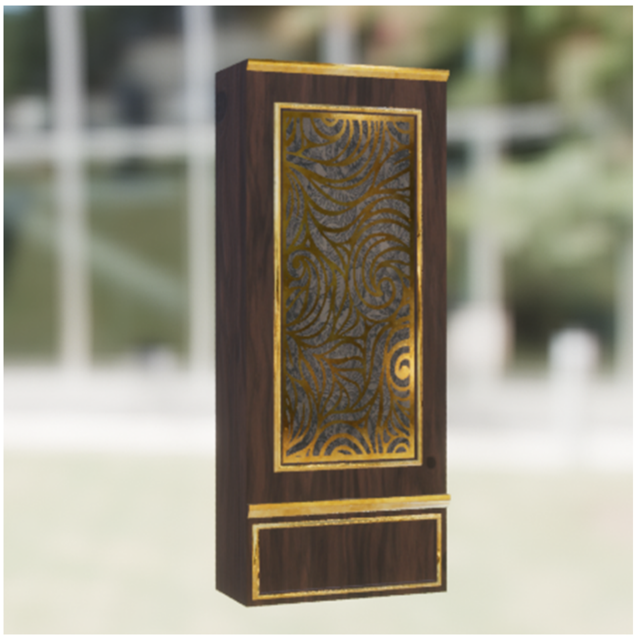


Figure 4.7: Wall Final Model

Puzzle 1

The first puzzle to be modelled was the lamp puzzle. For this one we had to model a table, a parchment, simple wooden tiles and a hanging lamp model. Each of the tokens had a personalised material on which a different symbol was placed. For the hanging lamps, 4 of the symbols on the tokens were chosen and one was placed on each of the lamps. On the parchment a texture was created with a drawing of the room and 4 marks marking the position of each lamp. The idea was for the player to place the correct token in the correct position using the lamps as indications of the position of each one.

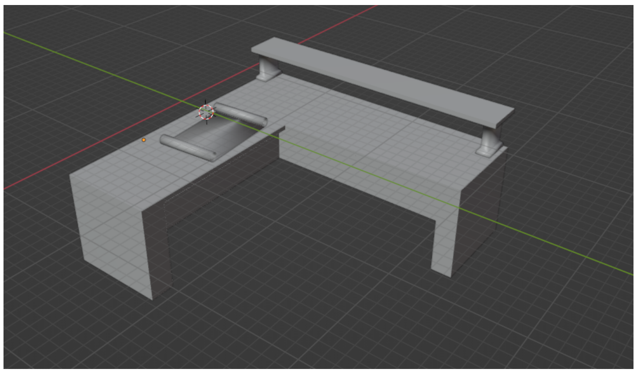


Figure 4.8: Blender First Table Model

After finishing the main elements, purely aesthetic elements were modeled to decorate the table since, with only the parchment, it was too empty and gave a very artificial feeling. To decorate the table, elements such as different types of books (new books, worn books, etc.), closed scrolls, storage boxes, jars, glass containers, etc. were modeled (Figure 4.9).

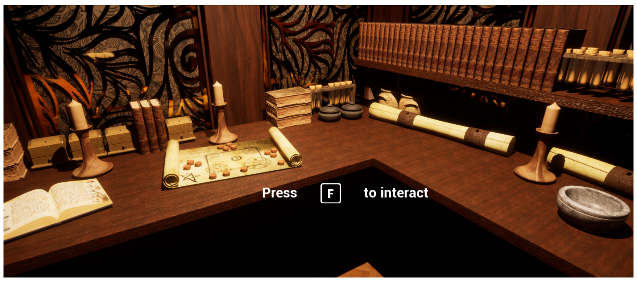


Figure 4.9: Blender Table Model with decorations on Unreal Engine.

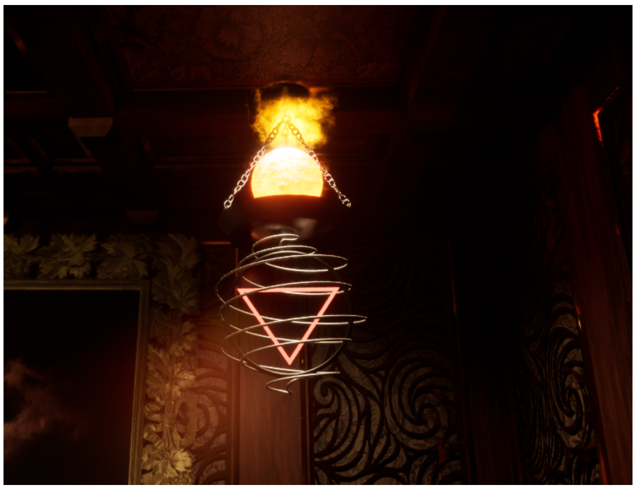


Figure 4.10: Lamps for the Puzzle 1 that mark the position with a symbol under them.

Puzzle 2

The second puzzle is a well-known puzzle that has been recreated in several games throughout the history of video games. It is a puzzle in which the player has to rotate certain objects, usually sculptures, until he finds the correct position for each one of them. For example, this puzzle can be found in games such as *Uncharted* (Figure 4.11), *Genshin Impact* (Figure 4.16) and many more.

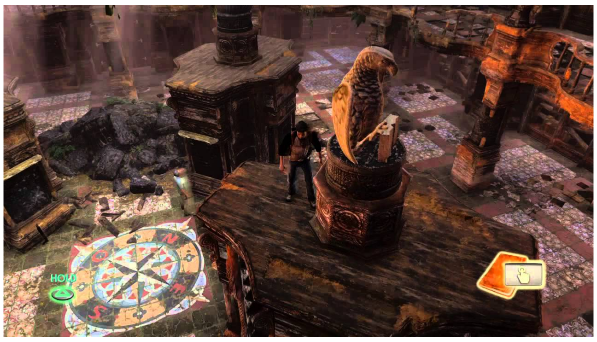


Figure 4.11: Uncharted Rotating Puzzle.

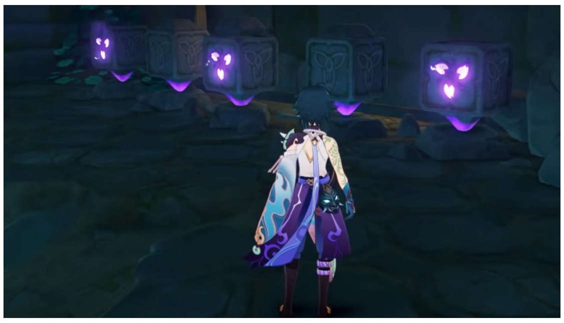


Figure 4.12: Genshin Impact Rotating Puzzle.

This puzzle concept has been developed for this project. Tables have been modelled on which a modelled figure will be placed. This modelled figure has been downloaded from a website, and is the only non-personal model[11] of the whole project. Even so, some imperfections have been edited in the ZBrush [14] program, specifically on the chest of the sculpture. The feathers of the owl's wings have also been polished and its textures have been modified to fit in with the aesthetics of the project. Finally, it has been implemented in the game scene. The reason why this model has been downloaded from the internet is because we wanted to implement and study the Nanite technology offered as an innovation by Unreal 5.

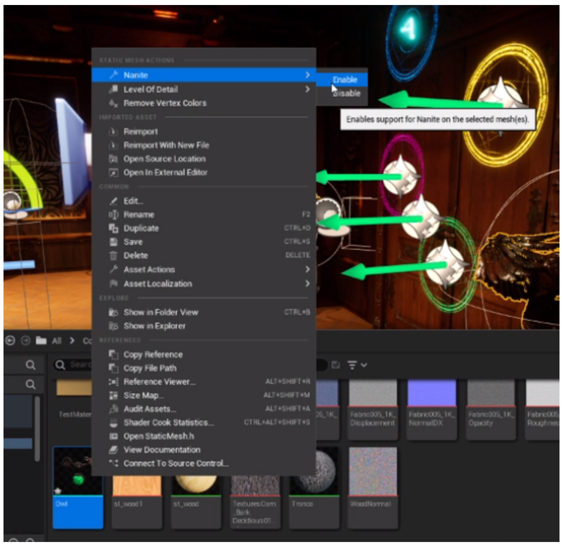
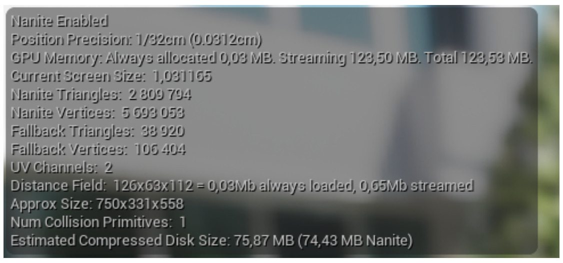


Figure 4.13: How to enable Nanite in a mesh.



```
Nanite Enabled
Position Precision: 1/32cm (0.0312cm)
GPU Memory: Always allocated 0,03 MB. Streaming 123,50 MB. Total 123,53 MB.
Current Screen Size:  1,031165
Nanite Triangles:  2 809 794
Nanite Vertices:  5 693 053
Fallback Triangles:  38 920
Fallback Vertices:  106 404
UV Channels:  2
Distance Field:  126x63x112 = 0,03Mb always loaded, 0,65Mb streamed
Approx Size: 750x331x558
Num Collision Primitives:  1
Estimated Compressed Disk Size: 75,87 MB (74,43 MB Nanite)
```

Figure 4.14: Nanite Owl Stats.

Nanite is a technology that implements the latest version of Unreal 5 which uses a new geometric virtualised system to render a new mesh format. With this new technology, it is able to render pixel-scale detail with high polygon count objects. This means that when the Nanite option is activated and applied to a mesh, this process analyses the mesh and decomposes it into clusters of triangle groups (Figure 4.14). Finally, when rendering, the system only renders the clusters according to the position of the player, showing only the details that are visible to the player and ignoring those that are hidden.

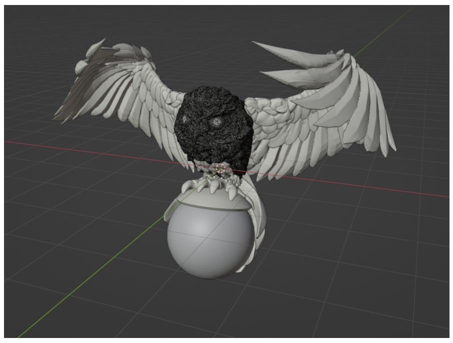


Figure 4.15: Blender Owl Model.

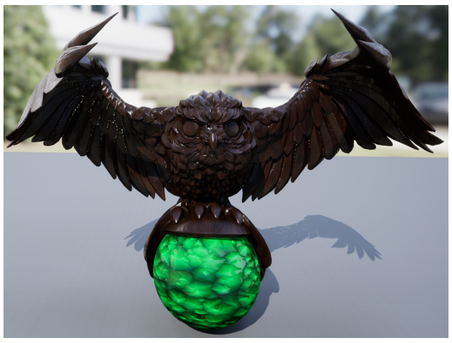


Figure 4.16: Unreal model with new texture and small changes.

For this second puzzle to test this new technology, the owl model has approximately 5 million polygons. In the first version we tried to instantiate all 5 owl models at the same time in the scene. I didn't manage to get to the point of producing a visible frame delay. Later, when programming the movement of the owls, it was observed that there really was a frame delay when the player interacted with the sculpture. By activating the Nanite this stopped happening.

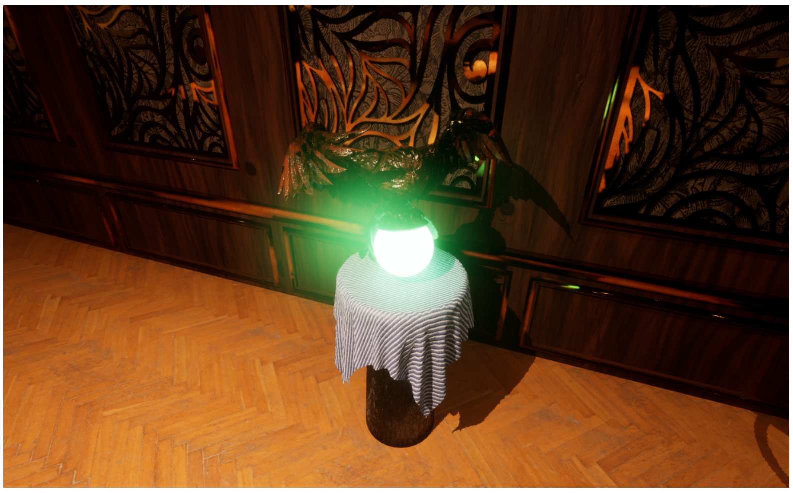


Figure 4.17: Final Owl Sculpture in the environment.

Puzzle 3

The third puzzle consists of an arrangement of objects. This one didn't require much effort for modelling as it consisted of a cupboard and a single trophy-like model (Figure 4.20)to place the textures.For the order of the moons, the player was expected to find the order of a picture that changed the image it projected. When it comes to the painting, a very flat and simple container has been modelled and to achieve all the details that can be seen in the final result of this one, Normal maps have been used. Thanks to the normal maps (Figure 4.21), the rendering system calculates the lights and recreates a displacement in the plane of the object creating a sensation that the object really has relief when in fact it is a totally flat object. (Figure 4.19)

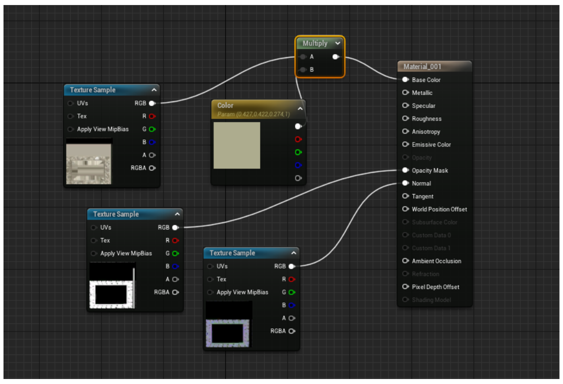


Figure 4.18: Frame material with displacement and normal map.

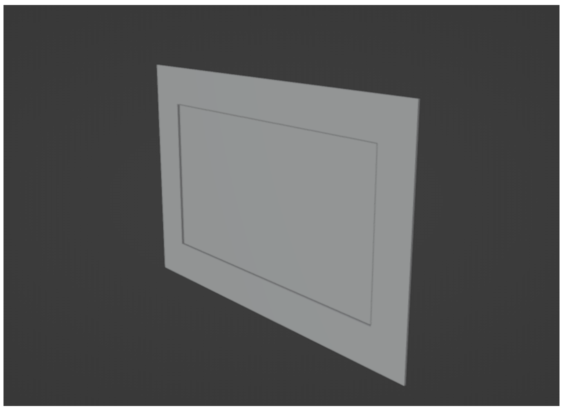


Figure 4.19: Frame in blender model.

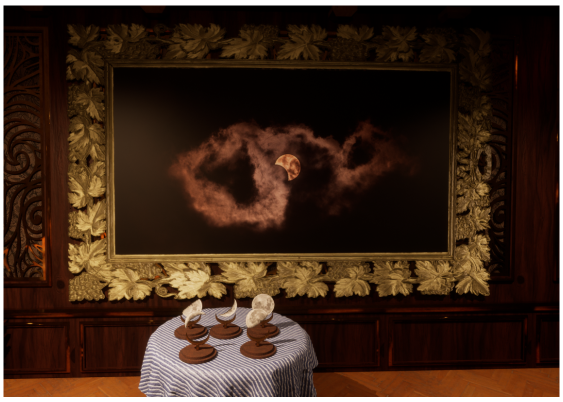


Figure 4.20: Final Frame and Moon with textures in Unreal.

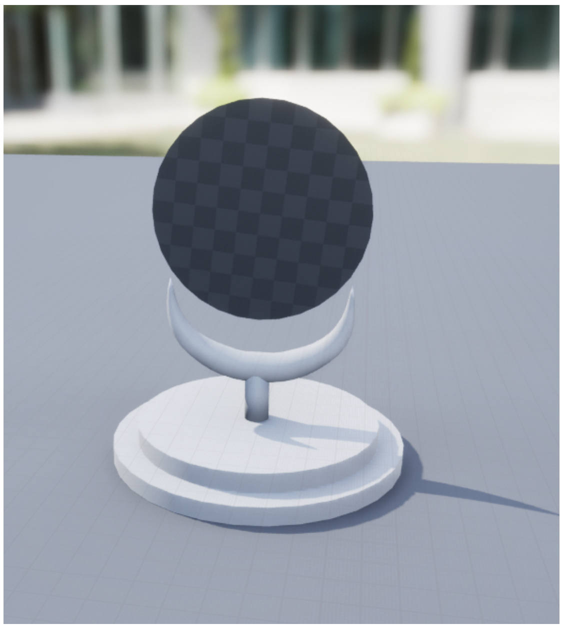


Figure 4.21: Base Moon sculpture mesh.

The central table of the room has also been modelled, on which a sphere will be placed, with which the player will come into contact in order to change the image of the painting. Everything has followed the same modelling process as explained in Puzzle 1. However, the orb that the player comes into contact with is not a 3D model. This is a Niagara particle system. This has been created from a template provided by Unreal Engine that generates particles constantly. Then different materials have been created using variables that multiply noise maps. In the Niagara system these materials are applied and a timeline is created which increases and decreases these variables to produce the animation effect. Some Niagara properties have also been added. Once the material was applied, the base colour of the material was tested in order to achieve a more satisfactory result in terms of colour.

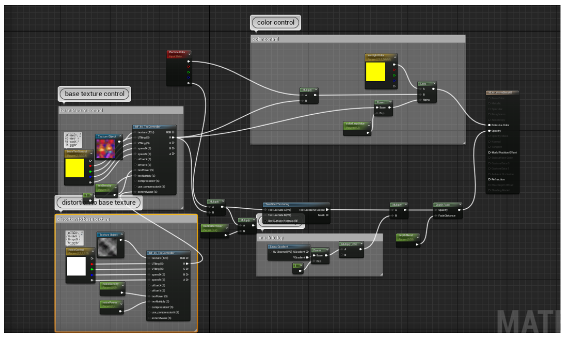


Figure 4.22: Orb Material in Unreal Engine with movable variables.

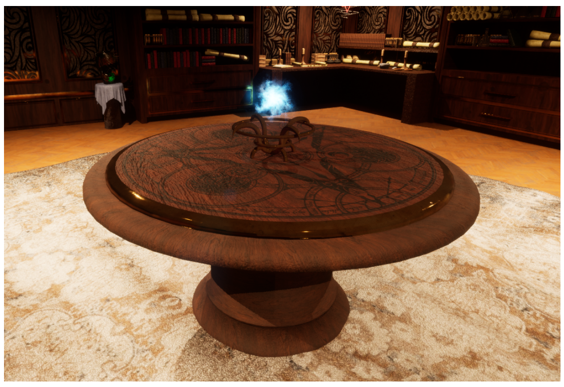


Figure 4.23: Base Moon sculpture mesh.

Puzzle 4

The fourth puzzle is about sliding several pieces together to get a clear picture. To do this, a container had to be modelled in which the puzzle would be solved and a piece that, later in Unreal, would be duplicated and assigned to the appropriate part of the image. The image has been divided into a 3 x 3 grid, and to cut the image in an exact way I have simply worked in Photoshop[8] with some tools called "rulers" to be able to measure to the millimetre the cutting area of each piece.

General Decoration

Once all the puzzles of the project had been modelled, they were assembled in the Unreal scene. Once assembled, purely aesthetic elements have been modelled to make the room come to life and give the sensation that it is indeed a real stage. These are purely aesthetic objects that the player cannot interact with, but they give a lot of life to the room. Among these objects we can find elements such as the sofa, scrolls, glass vases, books, etc.

None of these models have been a big problem to model, as the same techniques used for the puzzles have been used. The only difference is that, in 2 objects in particular have been used a modifier called Cloth of Blender, which recreates a simulation in the behaviour of the object to which it has been applied of how that element acts over time once applied a value of engraving and collision when in contact with other objects.The following are images of objects modelled using the methods explained above. These results are image captures inside the Unreal Engine 5 editor after going through the modelling process and generating the UV map in Blender, and the subsequent texturing in Unreal Engine 5.

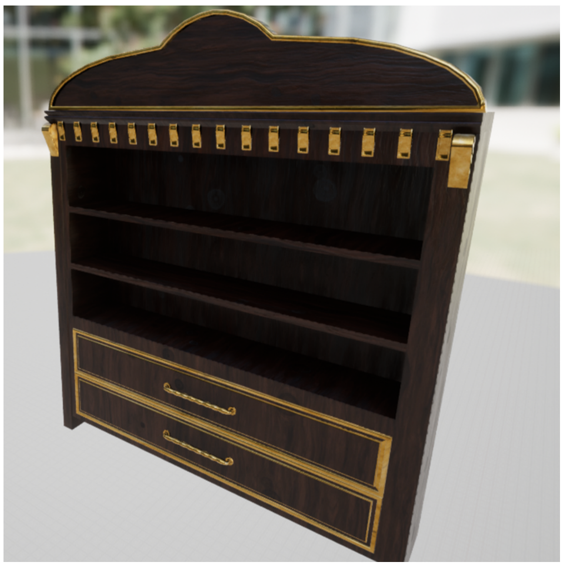


Figure 4.24: Cabinet Model.

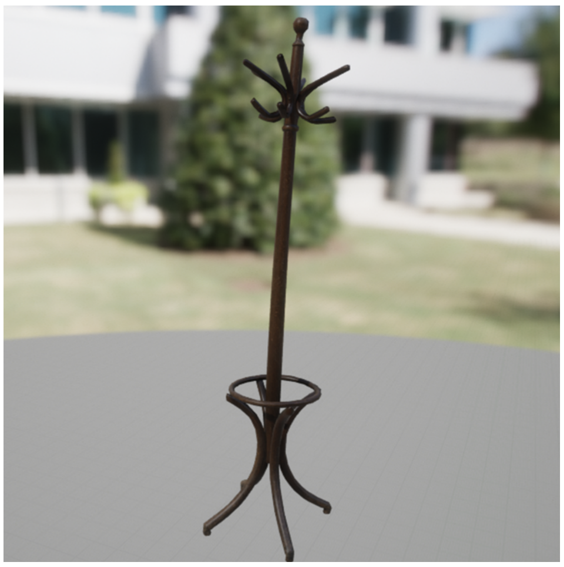


Figure 4.25: Coat Rack model.

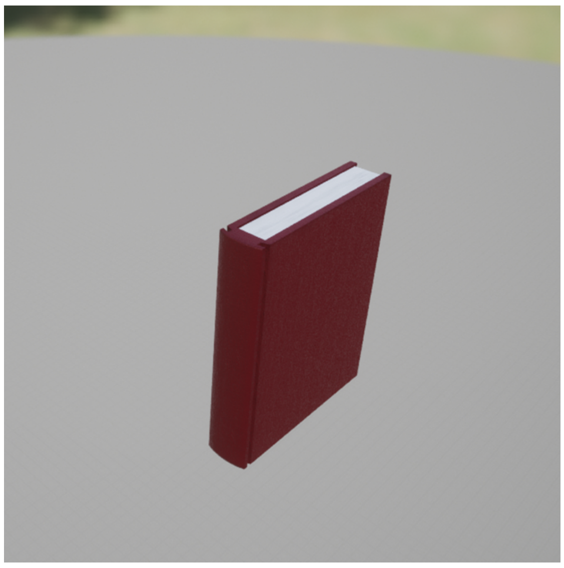


Figure 4.26: Book 1 Model.

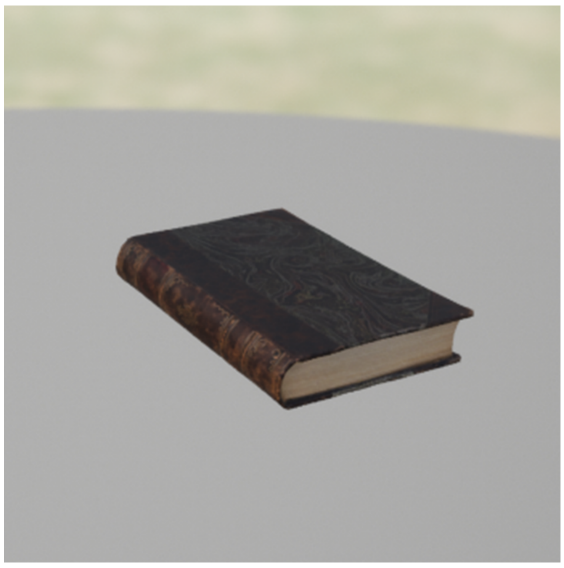


Figure 4.27: Book 2 model.

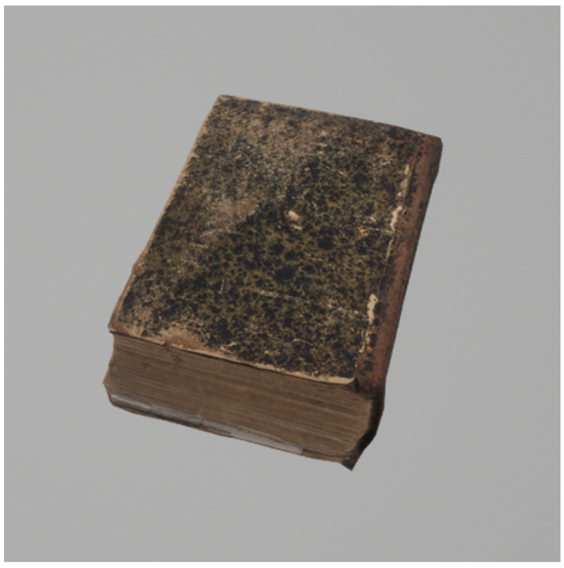


Figure 4.28: Book 3 Model.

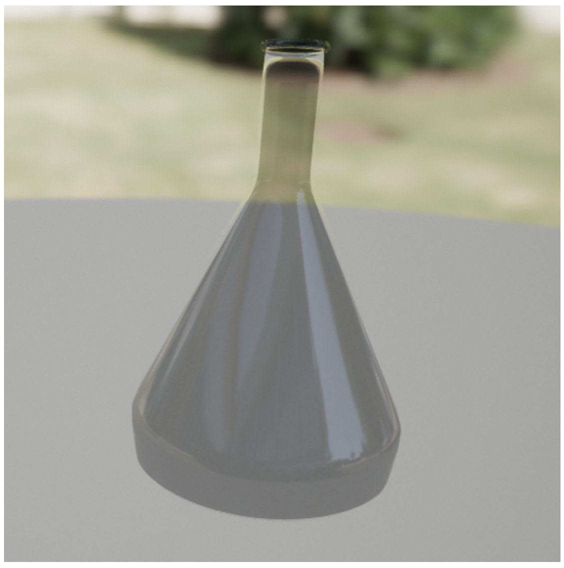


Figure 4.29: Glass Bowl Model.

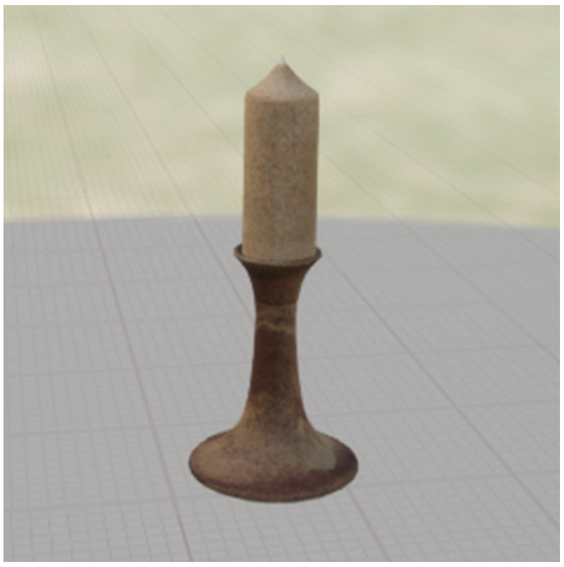


Figure 4.30: Candle Model.

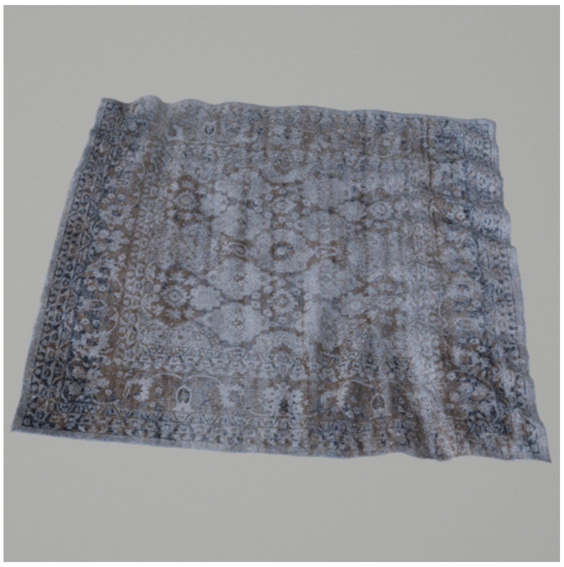


Figure 4.31: Carpet Model.

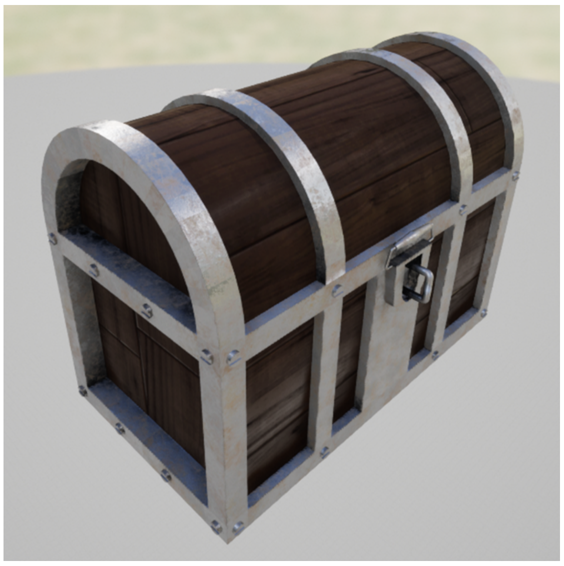


Figure 4.32: Chest Model.

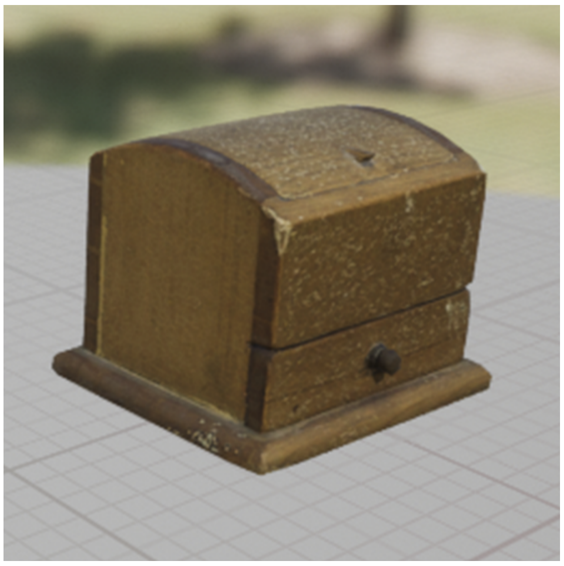


Figure 4.33: Small Wooden Chest Model.

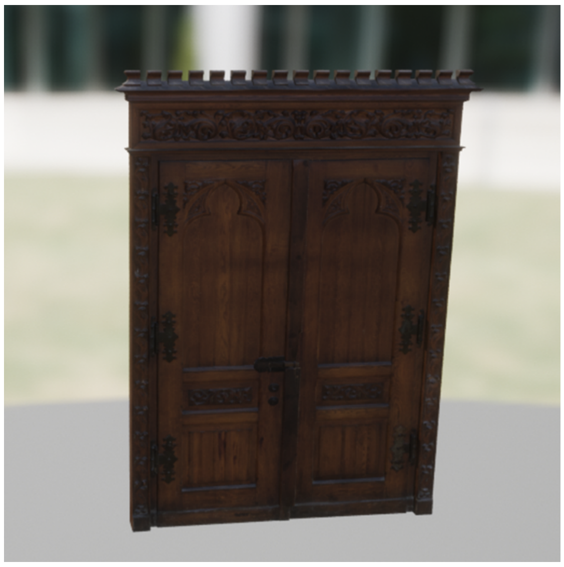


Figure 4.34: Door Model.

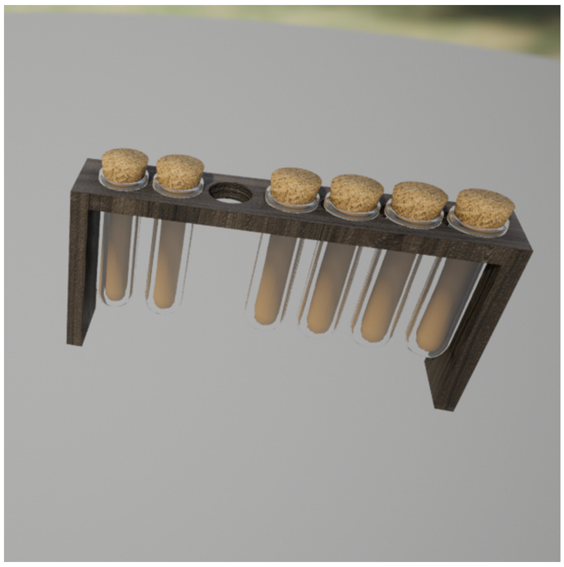


Figure 4.35: Sample Bottles Model.

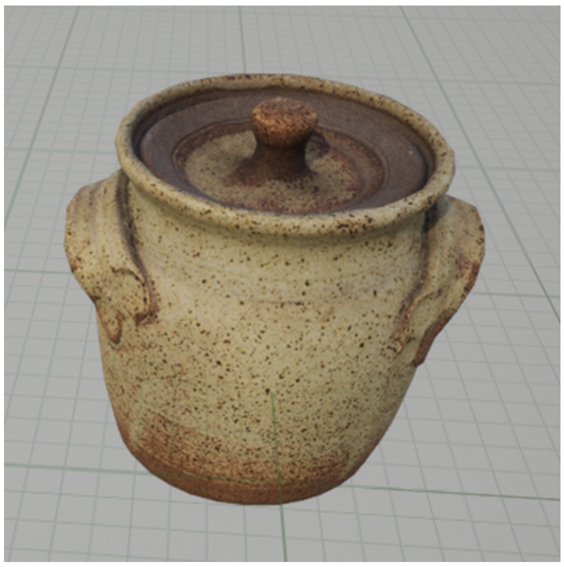


Figure 4.36: Jar Model.

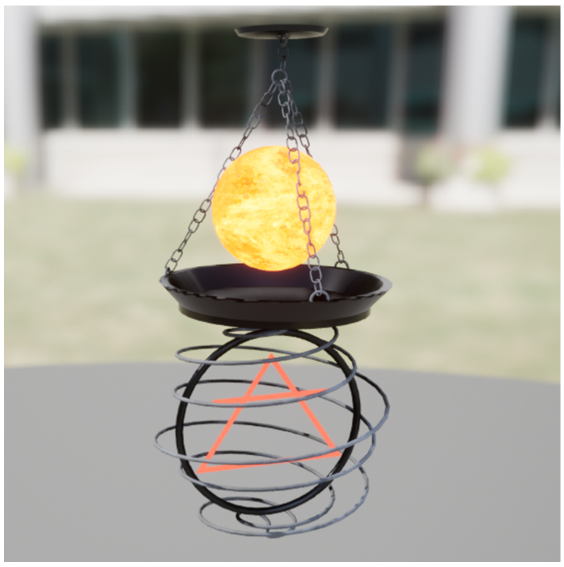


Figure 4.37: Lamp Model.

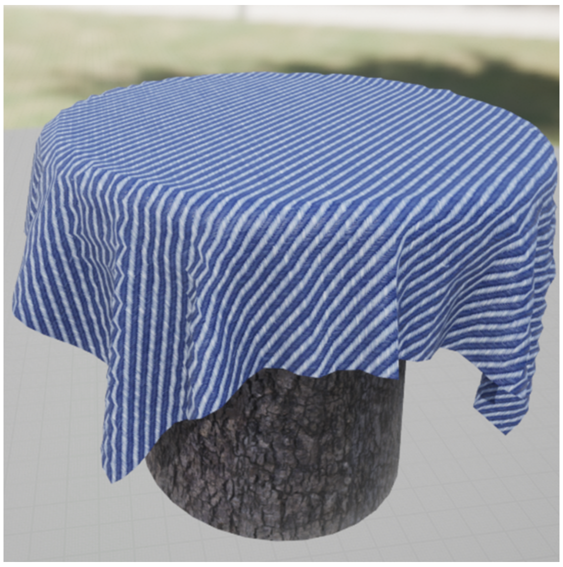


Figure 4.38: Table Model.

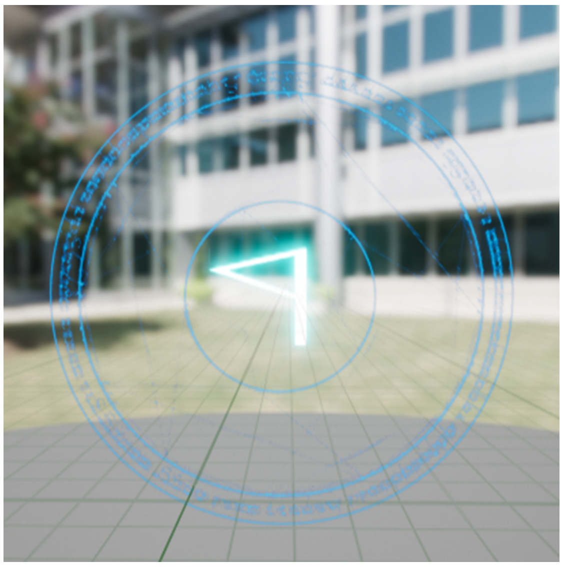


Figure 4.39: Niagara Particle Effect Visualized.

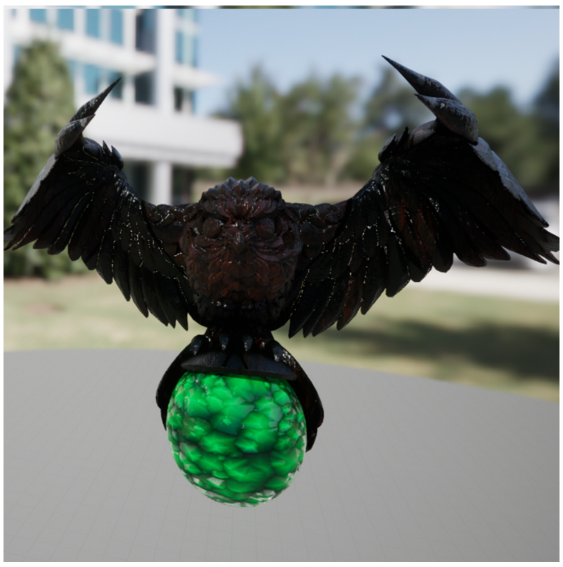


Figure 4.40: Owl Model.

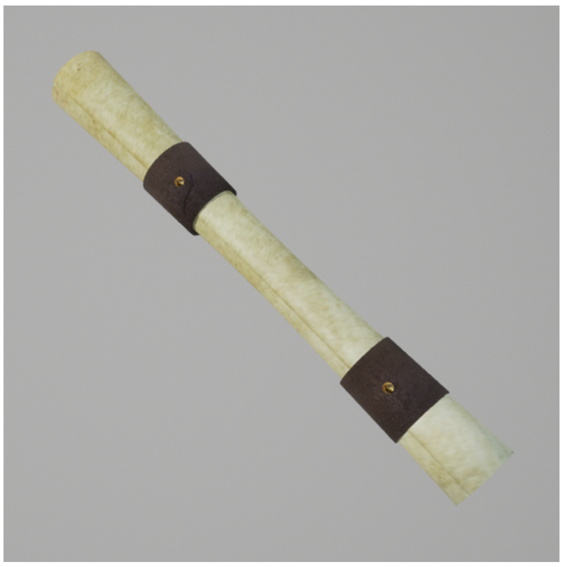


Figure 4.41: Parchment 1 Model.

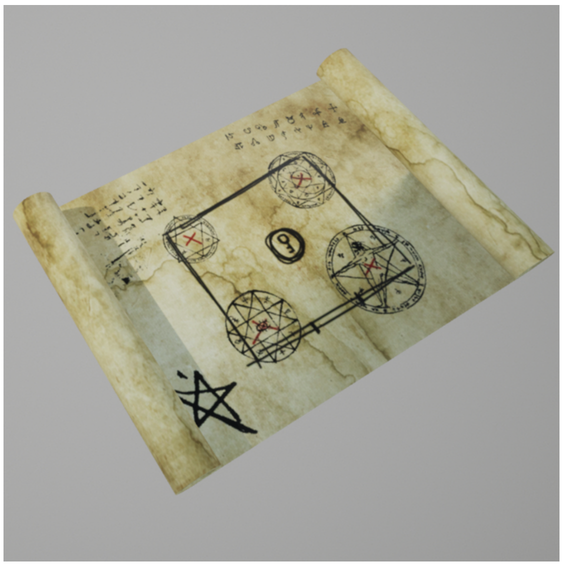


Figure 4.42: Parchment 2 Model.

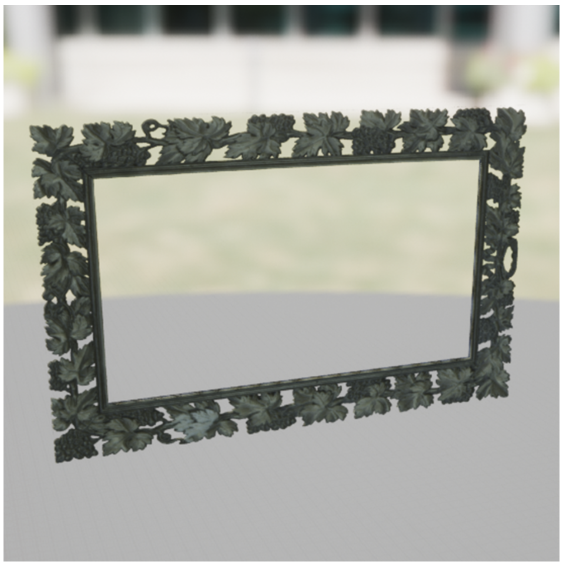


Figure 4.43: Picture Frame Model.

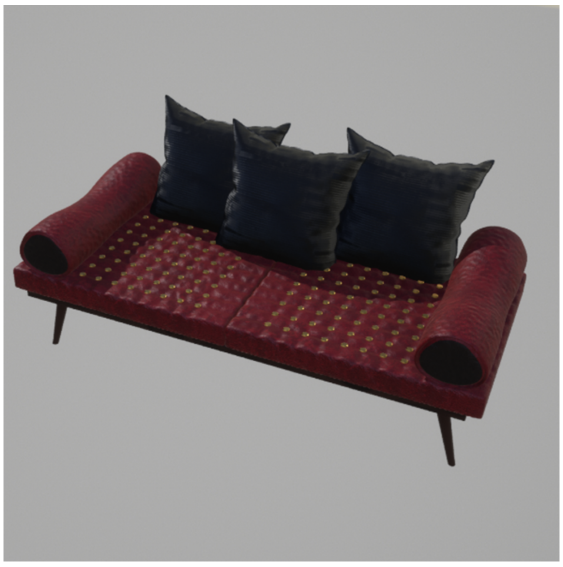


Figure 4.44: Sofa Model.

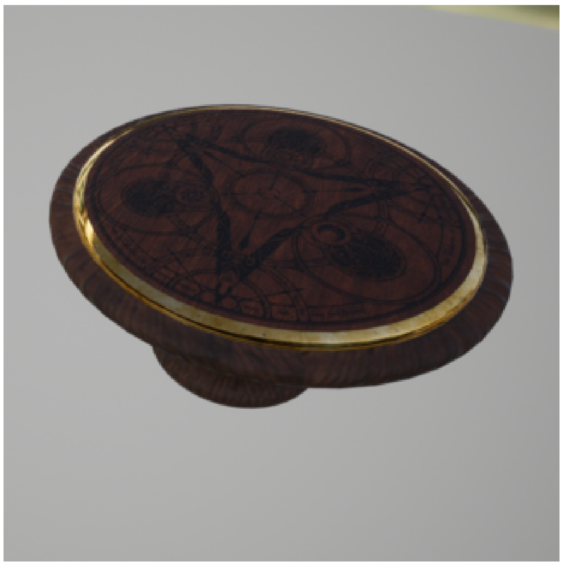


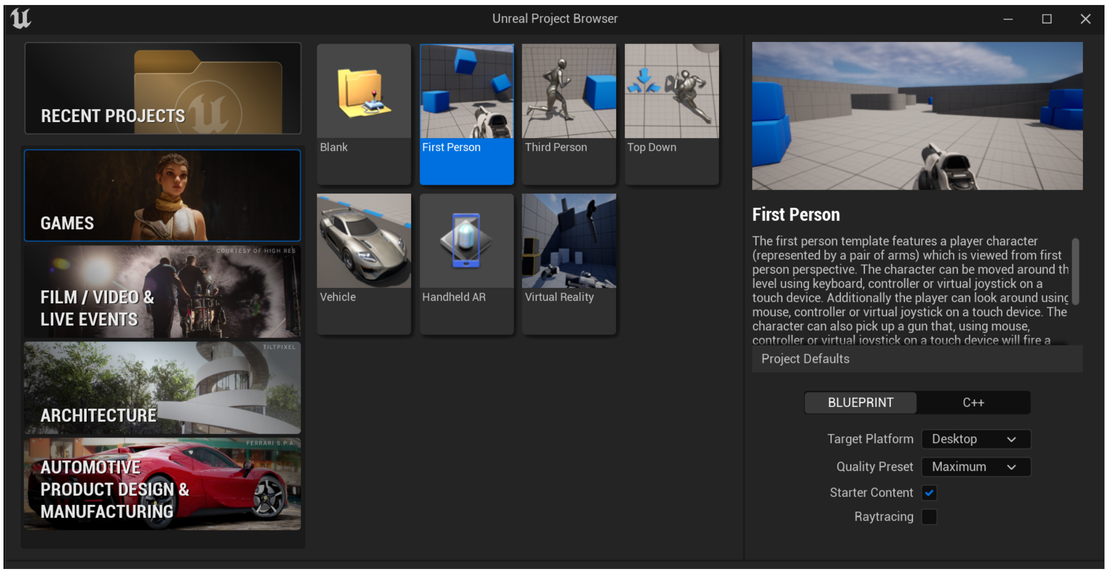
Figure 4.45: Table 2 Model.

4.1.3 Programming

As explained above, a system integrated in Unreal Engine has been used for programming, which avoids the process of writing code, instead it works in blocks and these are connected to each other in order to carry out the different functions. After some time programming for the project, we have come to the conclusion that, in reality, this method is not so different from the traditional method of programming, as in written programming, variables are declared, functions have to be created and called, etc. Therefore, rather than a new programming style, I would call the Blueprints system nothing more than a method to visually see the code.In the following, we will detail the most important elements for the programming of the project, explaining how they work in detail.

4.1.4 Player Movement

For the main movement of the player, the Unreal 5 First Person Shooter template has been used as a basis, which provides us with a simple code already created. The only thing that needed to be changed was to disable the character's fire button and remove the mesh component of the weapon inside the main player actor. The template is chosen at the moment when you create a project, and there are different types of templates (Figure 4.46).



The first person template features a player character (represented by a pair of arms) which is viewed from first person perspective. The character can be moved around th level using keyboard, controller or virtual joystick on a touch device. Additionally the player can look around usinc mouse, controller or virtual joystick on a touch device. The character can also pick up a gun that, using mouse, controller or virtual joystick on a touch device will fire a

Target Platform: Desktop
Quality Preset: Maximum
Starter Content: ✓
Raytracing: ☐

Figure 4.46: Unreal Project Templates

Puzzle 1: Interaction with player and Camera Change

The first programming challenge has been to interact with the objects. Therefore, in order to develop it, a Trigger component was added, which has a function that calculates when the player's collider comes into contact with the scroll's Trigger. A new camera has also been added for camera switching to have a sky view of the scroll and to have a good visibility of the scroll. (Figure 4.47)

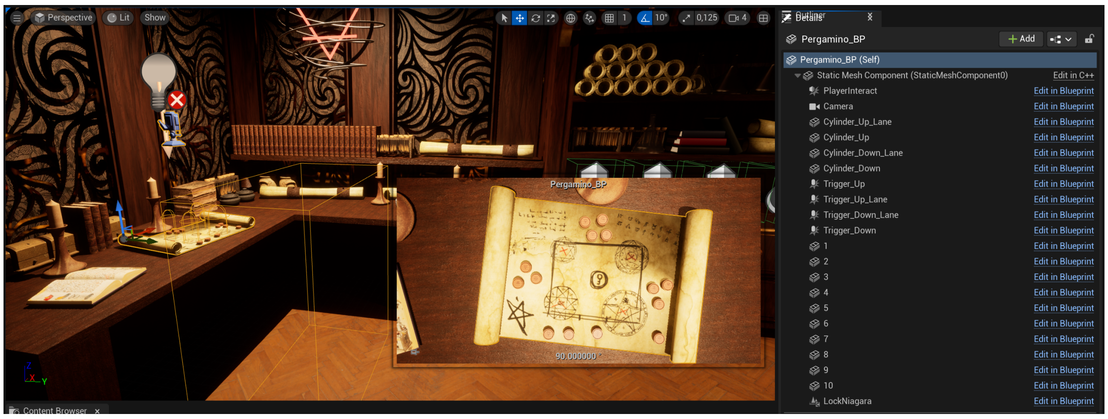


Figure 4.47: Unreal Puzzle 1 Components Inside Editor

As soon as the player comes into collision with the Trigger of puzzle one, it sends out a call that is made to all objects that the player comes into contact with and can interact with. What this does is to check if the colliding object is the player and if so, it allows to read a keyboard input of the key indicated in the message to interact with it. In the following code fragment (Figure 4.48) we can see the opposite of what is explained. Once the player has pressed the key to interact, it hides the interaction message and proceeds to perform a Blend function that makes a smooth transition to the new camera. At the end of this, the mouse is displayed on the screen, as it will be needed next to complete the puzzle.

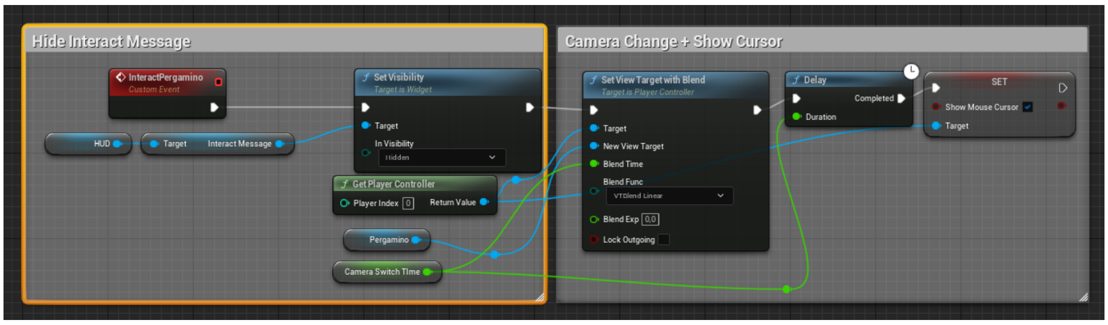


Figure 4.48: Hide message and enable camera change.

Once the above has been done, we now have a code that allows the camera to be changed. For the player's convenience, a code similar to the previous one has been implemented with another key so that the player can leave this camera position at any time.

For the programming of puzzle 1 itself, a similar system of triggers has been used. In this one, 4 triggers have been placed in specific positions on the scroll, and these detect collisions with tokens that are component child objects of the scroll object. In this picture (Figure 4.47) we can see the triggers and the children in the hierarchy on the right side. To check that it is the correct token, once a token collides with any of the triggers, it checks if the interacting actor in question is the same actor that has been previously assigned to the script manually. If all 4 triggers have the correct position for each of the tokens, this code destroys the Niagara particle system associated with the puzzle, the corresponding seal on the main door is destroyed.

Puzzle 2 : Rotating Actor on Correct Position

For this puzzle, the same Trigger system has been used as in the previous one to check whether the player can interact with the object or not. Once the player presses the key to interact, the owl that activates the code performs a rotation using the Rotation property of the object. This is calculated from a variable created in which the range of rotation of the sculptures can be modified. (Figure 4.49)

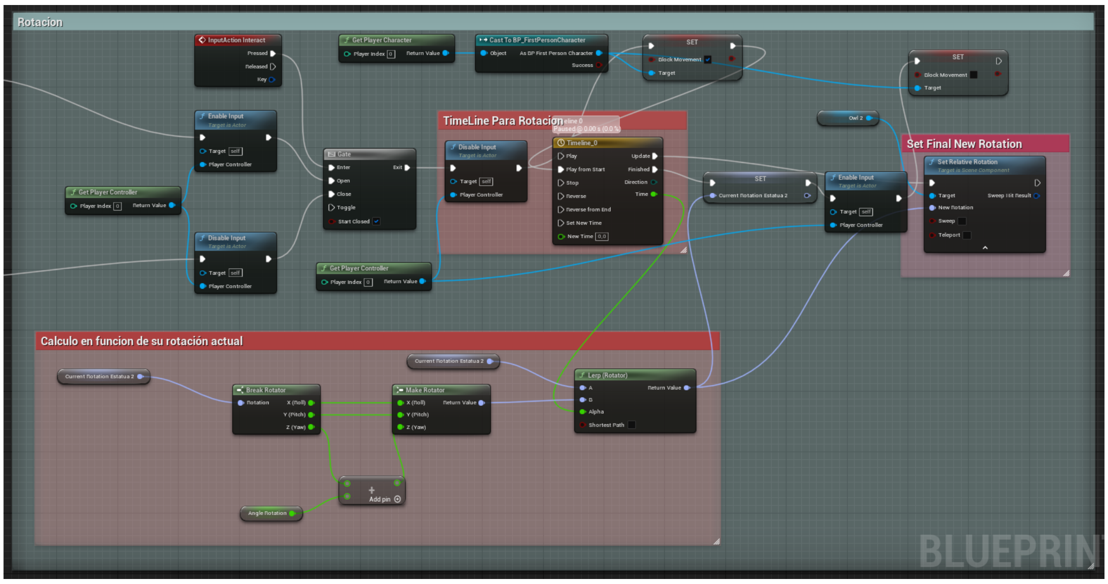


Figure 4.49: Hide message and enable camera change.

In order to check if it is in the correct position or not, the sculptures have an additional Trigger placed in front of them that rotates with them. When this trigger comes into contact with a hidden actor in the scene, it changes the value of a Boolean to indicate that it is in the correct position (Figure 4.50). To give feedback to the player that he has placed the sculpture in the correct position, I have implemented a function that changes the emission value of the material of the owl's sphere, so that it lights up for 2 seconds when the owl is in the correct position. To avoid errors, once you activate a sculpture and it starts to rotate, the interaction button is blocked until the owl finishes rotating. (Figure 4.51)

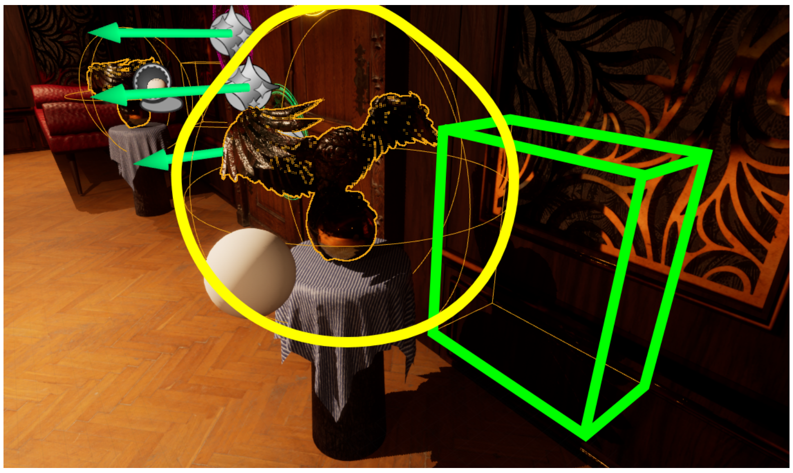


Figure 4.50: Owl component Schema. Yellow color represents the area of interaction of the player. Green color represents the Trigger area that checks the collision with the white ball for correct position.

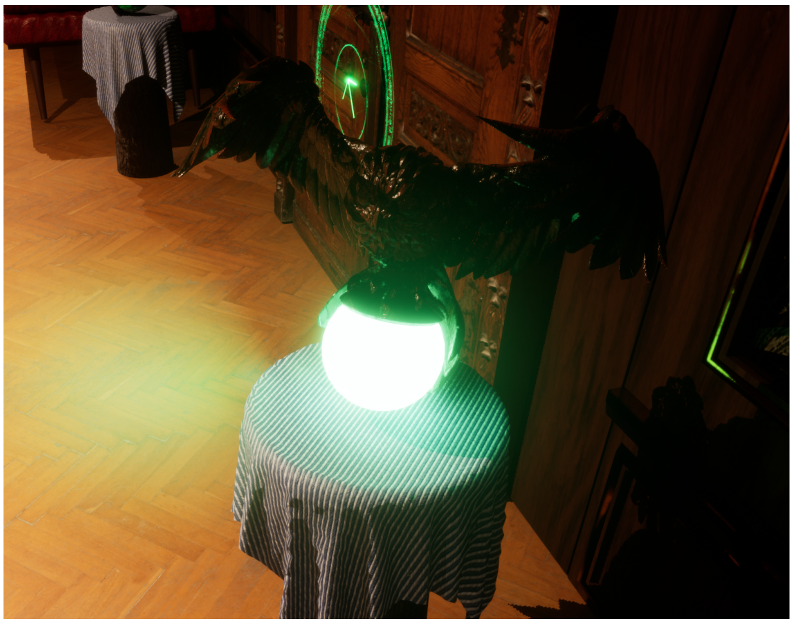


Figure 4.51: Owl in correct position.

Puzzle 3: Grab and Place Objects in the correct position

In the fourth puzzle a puzzle to grab objects and place them in the right position has been made. The same philosophy as in the previous puzzles has been used to check the collisions of these with Triggers that are components of the same object. In this case, 5 different Triggers have been placed and a table on which the 5 sculptures to be moved are positioned. To indicate to the player the order of these, a box has been implemented which, when interacting with the central table, changes the image displayed and the player has to place the Moons on the shelf in the order in which they appear in the images.

The most important points of this puzzle are the actions of taking each of the Moons. For this, a RayCast (Figure 4.52) system has been implemented, which generates a ray from the player's position to a distance. If in the path of this ray, it collides with the collider of one of the Moons, these are set as a "Grabbed Object". This works only with moons, because when the RayCast comes into contact with the moon, it is checked as a moon type object, so the player cannot grab any other object. As long as the player holds down the left mouse click, the "Grabbed Object" will change its position depending on the player's movement, setting Moon's object on the center of the player camera screen.

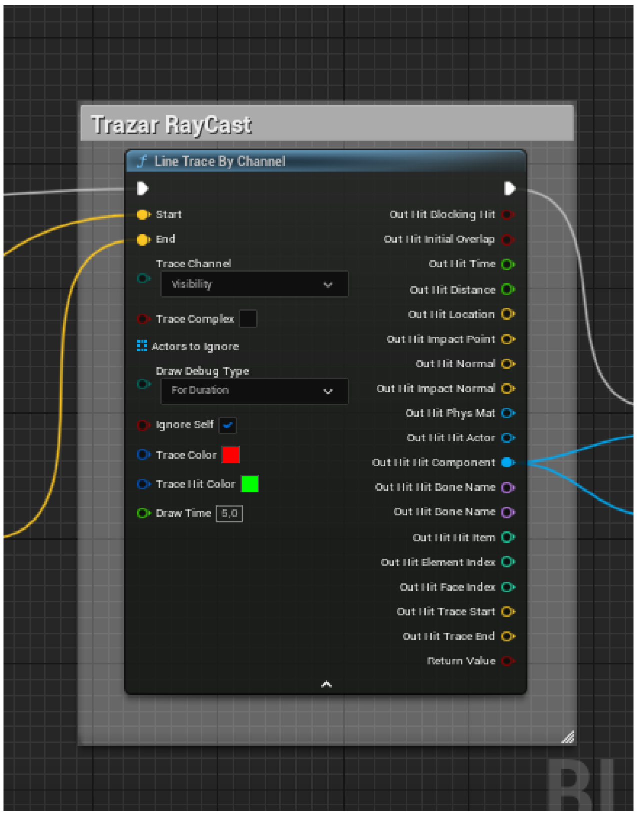


Figure 4.52: RayCast function in Unreal Engine.

Once the player moves the moon object closer to an indicated position, it will come into contact with one of the 5 triggers for each position of each moon. Upon contact it is checked to see if it is a moon object, if it is, this fixes the position of the moon object in the centre of the Trigger and the player releases the object. After it is placed, it is checked by the name of the moon type object to see if it is in the correct position. When all 5 objects meet this condition, the corresponding seal on the main door is destroyed.

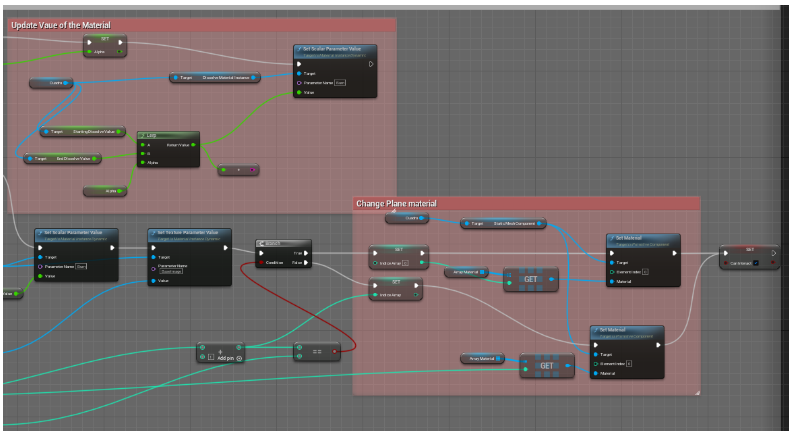

Update Vaue of the Material
Change Plane material
Set Scalar Parameter Value
Set Texture Parameter Value
Set Material
Lerp
Branch
GET
SET

Figure 4.53: Change Material function and Plane Changer function.

For the picture code, it was first necessary to duplicate the plane on which the image is displayed. This is because, the material that provides the burning effect of the frame, remains invisible as it performs its effect. Therefore, by means of code what is done is to place the next image in the plane placed 0.1 millimetres behind the front plane and when the burning animation is finished, the material of the back plane is assigned to the front plane and the next image is assigned to the back plane (Figure 4.53). As the player cannot move from the place when interacting with the orb, the instantaneous change between these 2 materials is imperceptible from the player's camera.

Puzzle 4

For this puzzle, as it is the last one to be programmed, parts of code from previous puzzles have been reused. For the sliding of the tiles between them, we have simply implemented the code used in puzzle 1 to move the tiles. In this case it was only necessary to check that there was not a piece in the position to which you wanted to move. The movements of each piece are restricted on the X and Y axes, so they cannot move diagonally.

To check the correct position of each of the pieces, the same method used in puzzle 3 has been used. It is an individual trigger for each piece that checks by means of a boolean if the piece in its position is the correct piece. Once all the pieces are in the correct position, the corresponding seal of the main door is destroyed.

Main Door

This is the end of the game, and it has a simple code that checks if all 4 seals have been destroyed. If they have, the player is allowed to interact with the door and move on to the endgame scene.

Interface

For the interface, a simple pre-game menu has been created, which is the one we are returned to once we complete the game. This has been created using a class provided by Unreal Engine that specialises in this. This creates a canvas in which we can add all the elements we want and, in turn, add to those elements the necessary Blue Prints to perform the actions we want.

For the In-Game HUD, as explained above, we have opted for a minimalist style, which simply consists of a white dot in the centre of the screen to help the player know exactly where the character is looking.

4.2 Results

Errors

Several errors have been detected in the programming of the entire game. The first bug detected was that, when you interacted at the beginning with any object, it automatically took you to the camera of the first puzzle. To fix this I had to identify each time the player interacts with an object, which is the object in question. It was simply to create a Branch[2] and check the name of the object.

Secondly, I had errors with the pivots of the objects. For example, in puzzle 1, when moving the pieces, they miscalculated their positions because the pivot was not in the centre of the object. This is due to the fact that when modelling in blender I was moving the object from side to side and when exporting it, it was not centred in the centre. To solve these persistent errors, the only thing I had to do was to centre the object in blender and re-import the mesh, an option that has been used repeatedly offered by unreal in which you import the mesh of the object again without having to re-assign all the materials and re instantiate all the objects of this type.

Several bugs were found in puzzle 3, which by all accounts was the most difficult puzzle of all. First of all, the picture planes did not work in the correct order. This was solved by creating 2 arrays that stored the materials in the same order + 1. That is, the position of the material [0] in the first list, which in this case belongs to the front plane, is the position [4] of the secondary plane, that is, the last position of this one.

A bug was also found after 3 days of trying to find it, which meant that the moons could not be moved out of place. This was because the Unreal module used to calculate the position initially only calculated the global position. To solve it, we only had to change the module to one that calculates the position and rotation, even if the object does not rotate in any direction

Minimal errors have been detected when rendering lights with lumen technology. This was due to the fact that, when setting up the environment, some objects had a separation of a few thousandths that was only perceptible from a certain angle and created errors with strange lights in the interior. This was solved with a little patience

[2] *Module that implements If...Else, a conditional.*

by examining and studying the origin of these strange illuminations and moving the objects that produced the error.

Planning Errors

In addition to the results I would like to emphasise the drastic change in project planning. The modelling part has taken me much more time than planned, going from the 70 hours initially planned to, approximately, almost twice as many hours. This forced me to work simultaneously starting to develop parts of the project code before finishing the modelling of the whole environment. Below (Figure 4.54) I attach an image of the time that each of the sections of the project actually took me. It has been done as a Gantt chart to compare with the initial diagram (Figure 2.1).

Graphic Results

For the final lighting an Unreal 5 actor has been added which is called Post Process Volume. This actor consists of an invisible cube that alters the way the light works. This allows us to add typical camera parameters to add more realism to the scene such as Depth Of Field, Lens Flare, Bloom, Anti-Alising, etc. After testing several ranges of parameters, we arrived at a satisfactory result which will be seen below.

In this section I will attach several figures to show the final results within the Unreal Engine editor

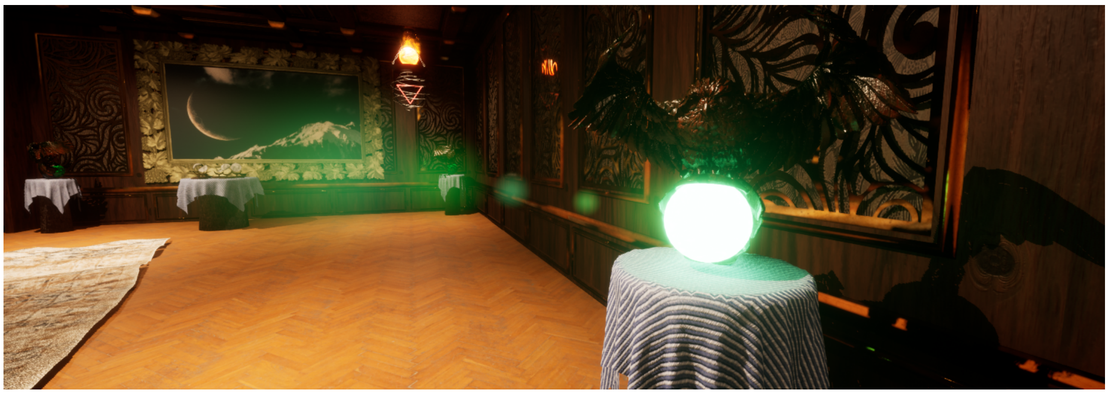


Figure 4.55: Final Render 1.

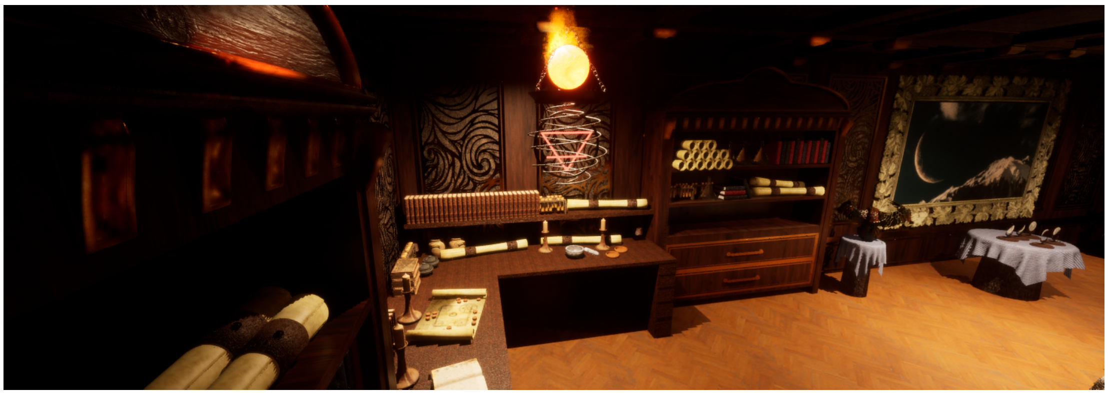


Figure 4.56: Final Render 2.

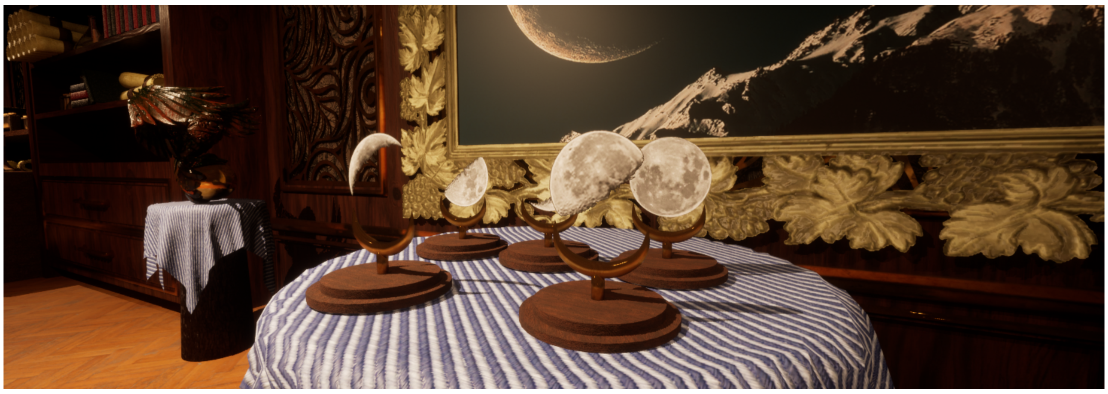


Figure 4.57: Final Render 3.

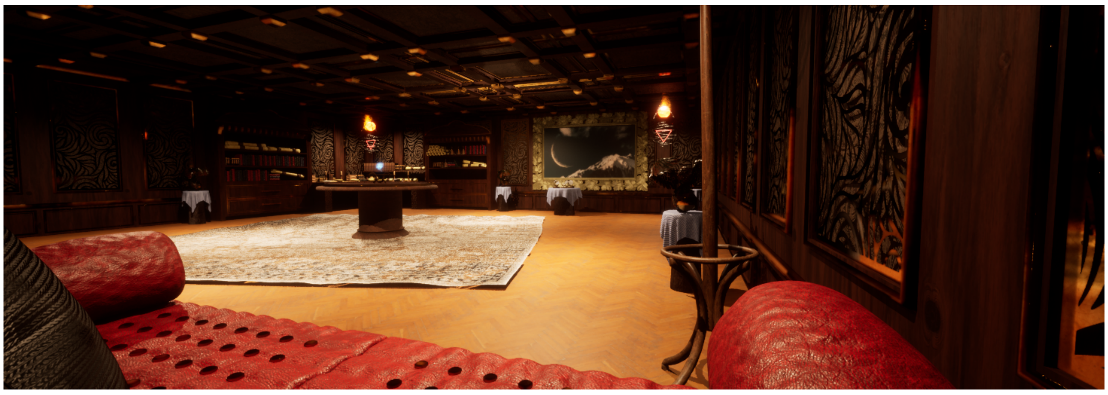


Figure 4.58: Final Render 4.

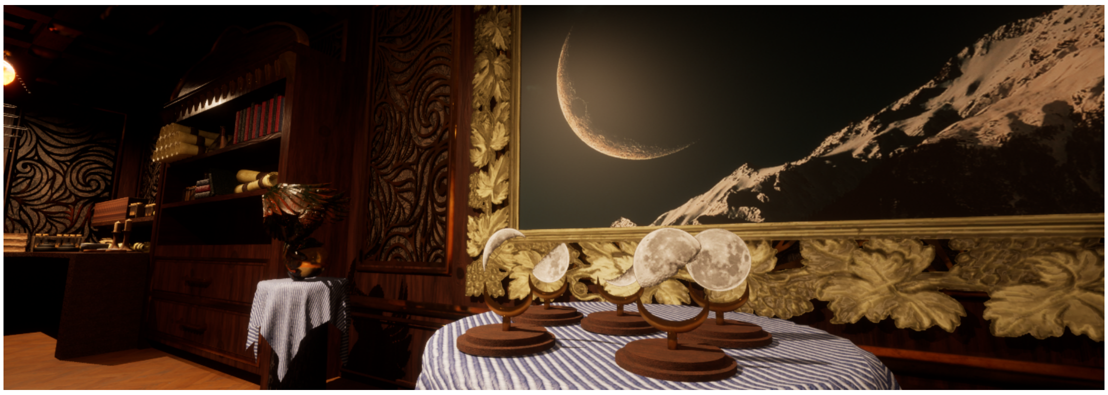


Figure 4.59: Final Render 5.

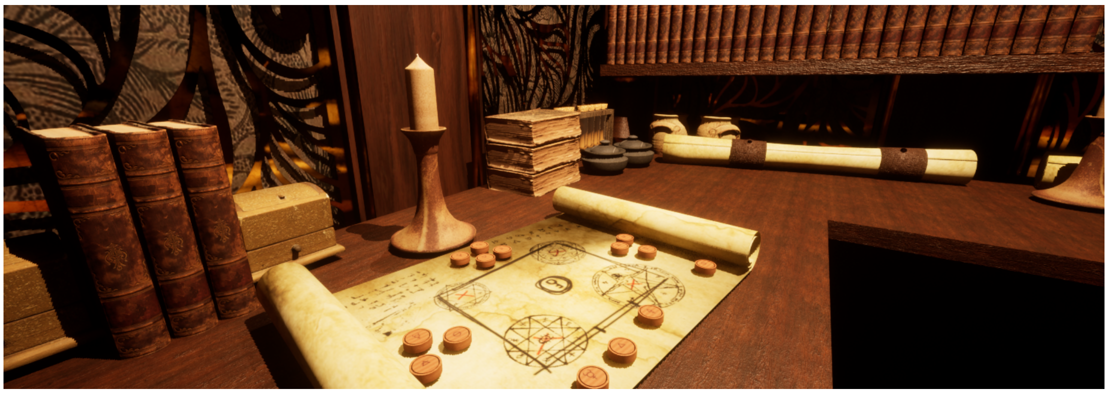


Figure 4.60: Final Render 6.

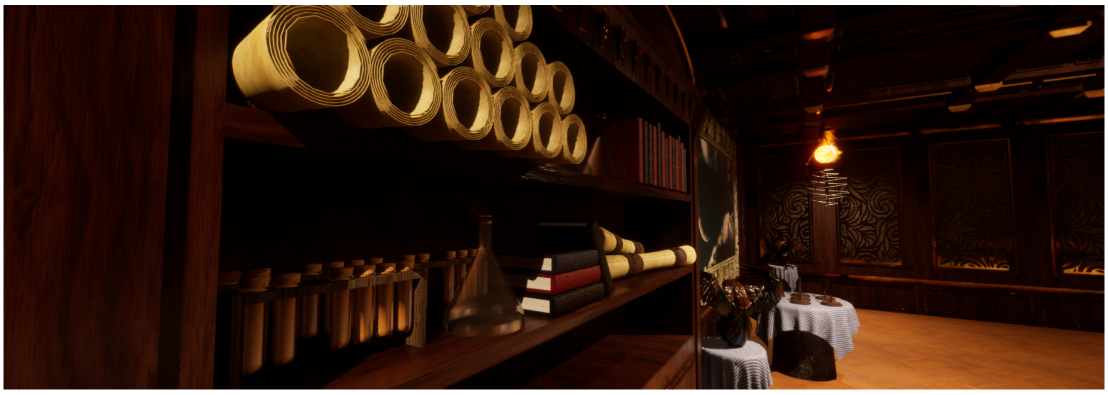


Figure 4.61: Final Render 7.

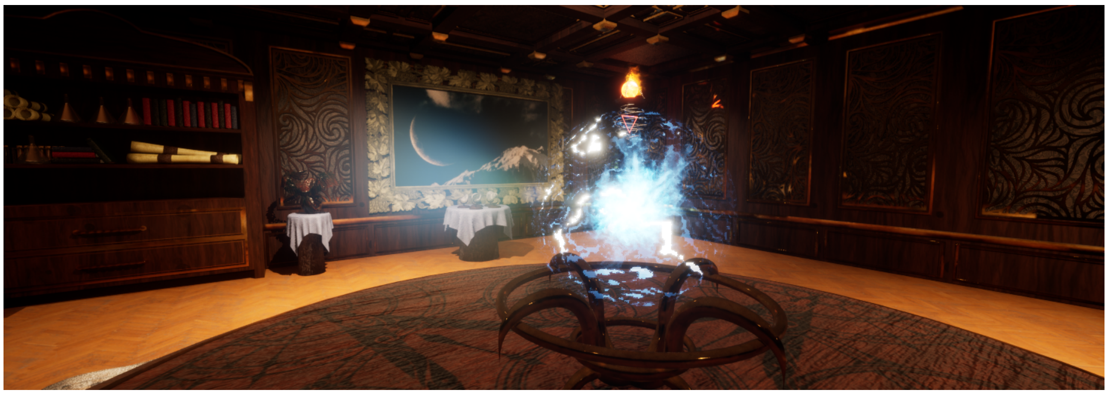


Figure 4.62: Final Render 8.

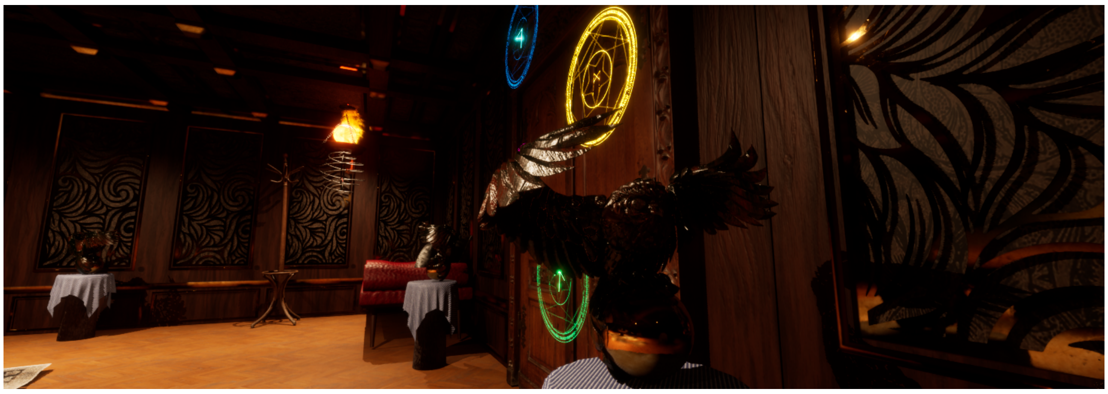


Figure 4.63: Final Render 9.

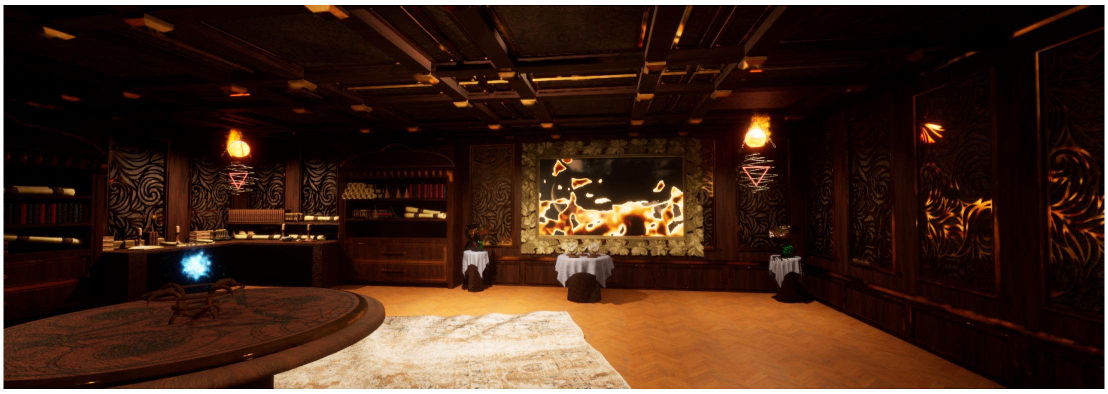


Figure 4.64: Final Render 10.

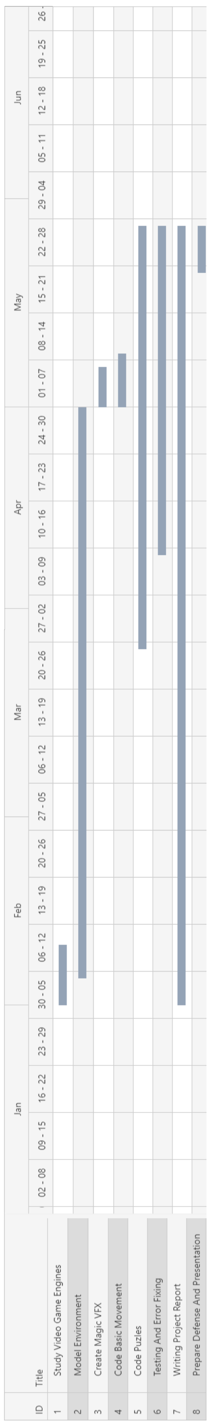


Figure 4.54: Actual time spent over the months to complete the project.

Chapter 5

Conclusions and Future Work

Contents

5.1 Conclusions

This project has served to learn about the development of video games with high realistic graphics. On the one hand, the curiosity to create a game with such graphic quality was present, since throughout the degree, it is true that we have carried out several video game projects, but not one focused as such on graphic quality. It's true that in some subjects it's rewarded to get a nice aesthetic but it doesn't give way to a much deeper development.

First of all, modelling skills have been developed a lot, as we have had to investigate several different modelling techniques on some occasions. For example, the most remarkable of all would be the work on textile modelling. The carpet that has been placed as decoration in the centre of the room was intended to give a natural look and not just a stretched plane. To achieve this, it was necessary to find ways to create wrinkles in order to create a more natural effect on the carpet. The tablecloths, which also followed a similar philosophy, used the same Blender modifier as the carpet. This modifier is the Cloth modifier, which in conjunction with weight and gravity variables allows us to simulate how a real cloth would act.

Secondly, the part in which most knowledge has been acquired and that I am most happy to have researched and learned is the Unreal Engine environment. This game engine is an engine that is not taught in the degree course, as Unity is taught in the degree course, an engine that has a much simpler learning curve than Unreal. Once you have some knowledge of how a game engine works internally and the methodology of work when developing a video game, which in my personal opinion, I am very grateful to the subject of engines of the degree, you can start to take the step towards Unreal. Thanks to all this, I have acquired a lot of knowledge in the development of video games in an engine that is currently in great demand in the industry.

To conclude, I believe that the final degree project itself, in addition to having acquired many skills in different fields, is also a project that I can show to the world to demonstrate these skills. Throughout the degree, several small video games are created with a goal in mind and almost always in a group. This project is something more personal, with a theme that I have chosen myself and I am very happy with the result, being proud to have this project to show to the public.

5.2 Future work

As future work in the future, certain aspects of the project are to be refined. First of all and I think the first step in the future is to create a short cinematic of the video game with the tools of Unreal Engine because, together with these tools, you can create a much more realistic image post-processing. On the other hand I would like to expand the concept from the living room to a whole house and distribute more puzzles around it, and as you progress, unlocking rooms inside the house.

I would like to study the steps and requirements of the platforms to be able to publish this project on them. First of all, look at the Steam publishing conditions and then publish it. This could be interesting in the future to be able to know in advance the requirements of a videogame before its release.

Bibliography

[1] Blender. Blender Sowftware Official Web Page. https://www.blender.org/.

[2] Escape Room by Wikipedia. Escape Room definition by Wikipedia. https://es.wikipedia.org/wiki/Escape_room.

[3] Unity Engine. Unity engine Official Web Page. https://unity.com/es.

[4] Unreal Engine. Unreal Engine Software Official Web Page. https://www.unrealengine.com/en-US/.

[5] Unreal Games Example. List of games developed with different Unreal Engine versions. https://es.wikipedia.org/wiki/Anexo:Videojuegos_que_usan_Unreal_Engine.

[6] Adobe Illustrator. Adobe Illustrator Official Web Page. https://www.adobe.com/illustrator.

[7] Autodesk 3DS Max. Autodesk 3DS Max Software Official Web Page. https://www.autodesk.es/products/3ds-max/overview?term=1-YEAR&tab=subscription.

[8] Adobe Photoshop. Adobe Photoshop Official Web Page. https://www.adobe.com/photoshop.

[9] Fire ProofGames. The Room. https://www.fireproofgames.com/games/the-room.

[10] QuoteFancy. Winston Churchill Quote. https://quotefancy.com/quote/940233/Winston-Churchill-Those-who-plan-do-better-than-those-who-do-not-plan-even-should-they, 2008. [Online; accessed 19-July-2008].

[11] Stevi. Decor owl ball N021118 - 3D model. https://archibaseplanet.com/download/5c2cd5e8.html.

[12] NiagaraParticle System. Unreal engine Niagara Particle System. https : / / docs . unrealengine . com / 5 . 0 / en-US / creating-visual-effects-in-niagara-for-unreal-engine/.

[13] Inc. Thekla. The Witness. https://es.wikipedia.org/wiki/The_Witness_(videojuego).

[14] ZBrush. ZBrish Official Web Page. https://pixologic.com/.