



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

---



# End of the Road

A Case Study in Building a Horror Game for Virtual Reality

Bachelor's thesis in Computer Science and Engineering

JOHAN BACKMAN  
ALBIN CASPARSSON  
ROBIN GRÖNBERG  
MAGNUS HAGMAR  
JONATHAN NILSFORS  
LINNÉA OTTERLIND

This page intentionally left blank.

BACHELOR'S THESIS 2015

## **End of the Road**

A Case Study in Developing a Horror Game for  
Virtual Reality

JOHAN BACKMAN  
ALBIN CASPARSSON  
ROBIN GRÖNBERG  
MAGNUS HAGMAR  
JONATHAN NILSFORS  
LINNÉA OTTERLIND



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2015

The Author grants to Chalmers University of Technology the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology store the Work electronically and make it accessible on the Internet.

### **End of the Road**

A Case Study in Developing a Horror Game for Virtual Reality

Johan Backman, Albin Casparsson, Robin Grönberg,  
Magnus Hagmar, Jonathan Nilfors, Linnéa Otterlind

© JOHAN BACKMAN, 2015.

© ALBIN CASPARSSON, 2015.

© ROBIN GRÖNBERG, 2015.

© MAGNUS HAGMAR, 2015.

© JONATHAN NILSFORS, 2015.

© LINNÉA OTTERLIND, 2015.

Supervisor: ERIK SINTORN, ULF ASSARSSON

Examiner: ARNE LINDE

Chalmers University of Technology  
Department of Computer Science and Engineering  
SE-412 96 Gothenburg  
Sweden  
Telephone +46 31 772 1000

Cover: *End of the Road* poster and logo.

Department of Computer Science and Engineering  
Gothenburg, Sweden 2015

## Abstract

This thesis explores the possibilities and limitations of virtual reality in gaming, with focus on the horror genre. One of the key features of this concept is the heightened level of immersion that can be achieved through the use of a head-mounted display. In order to investigate this, a case study in the form of developing a horror game for virtual reality has been performed. We have continuously analyzed the different development tools and game design guidelines available in order to present a suitable approach for developing such a game.

The case study has resulted in the horror game *End of the Road*. Due to the limited time frame of the project, we have been forced to exclude several game elements which we would have preferred to implement. Nonetheless, we consider the game to contain a satisfactory amount of content.

We have performed a series of user test sessions where non-project members were asked to play the game and evaluate their experience. After collecting all evaluations and compiling the data, we have concluded that the resulting game is perceived as frightening. Thus, we have fulfilled our ambition to utilize virtual reality to create a horror game. Whether or not the resulting game is considered more frightening than it would have been without using virtual reality is, however, uncertain. In order to establish this, we would require further testing, with and without the use of a head-mounted display, and a comparison of the gathered results.

Keywords: Virtual Reality, Horror Game, Oculus Rift, Immersion, Unreal Engine, Game Development, Game Design.

## Sammanfattning

Denna avhandling utforskar möjligheterna och begränsningarna för virtuell verklighet i spel med fokus på skräckgenren. Ett av karaktärsdragen för konceptet virtuell verklighet är den förhöjda inlevelsen som kan uppnås genom användningen av huvudmonterade skärmar. För att undersöka detta genomfördes en studie i form av att utveckla ett skräckspel anpassat för virtuell verklighet. De olika tillgängliga utvecklingsverktygen och speldesignsriktlinjerna har analyserats och utvärderats kontinuerligt för att presentera ett lämpligt tillvägagångssätt vid utveckling av ett sådant spel.

Studien resulterade i skräckspelet *End of the Road*. Till följd av tidsbegränsningen för projektet har flertalet önskade spelelement blivit exkluderade. Projektgruppen anser ändå att spelet, utan de exkluderade spelelementen, innefattar en tillfredsställande mängd innehåll.

Vi har utfört ett flertal användartestsessioner där oberoende personer blivit tillfrågade att spela spelet och sedan utvärdera sina upplevelser. Efter insamling och sammanställning av datan från sessionerna har vi dragit slutsatsen att spelet uppfattas som läskigt. Därmed anser vi oss ha uppfyllt vår ambition att skapa ett skräckspel med användning av virtuell verklighet. Om spelet uppfattas som läskigare än det skulle gjort utan användning av virtuell verklighet är dock oklart. För att kunna fastställa detta skulle ytterligare testning, med och utan användning av huvudmonterade skärmar behöva utföras och sedan jämföras.

## Acknowledgements

We would like to thank Erik Sintorn and Ulf Assarsson for being our supervisors and for providing invaluable feedback and advice during the course of this project. Additionally, we would like to express our gratitude to Arne Linde for his assistance in administering the necessary facilities and equipment.

We give our thanks to Guru Games, for inviting us to their office, answering our questions, and giving us helpful pointers on game development.

We would also like to thank our friends and family for their patience and willingness to be repeatedly subjected to *End of the Road* during its development. We thank them for their feedback, it is greatly appreciated. Furthermore, our thanks go to Harald Otterlind for providing us with equipment essential to our progress.

Finally, we would like to thank Jonas Sjögren for his baguettes, they were delicious!

*With love,  
Johan Backman, Albin Casparsson, Robin Grönberg, Magnus Hagmar,  
Jonathan Nilsfors, Linnéa Otterlind, Gothenburg, May 2015*

This page intentionally left blank.



# Glossary

**aliasing** within computer graphics is a phenomenon caused when a continuous line appears jagged because of low resolution, typically on a display. 23

**blend space** is an Unreal Engine feature that combines different animations in order to create a new animation. 19, 20

**central processing unit** is the primary component of a computer that processes instructions. x, 9

**destructible mesh** is an asset that determines how a mesh is divided into smaller, separate meshes. 19

**direct lighting** is light that travels in a straight line from a light source to a surface, which it illuminates. 11

**field of view** is the extent of the observable world that is seen at any given moment. x, 23

**global illumination** is a concept where realistic lighting is added to a 3D scene. 10

**graphical user interface** is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators. x, 15

**graphics processing unit** is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. x, 9

**head-mounted display** is a display device worn on the head that has a small display optic in front the eyes. x, 1

**heads-up display** is a user interface projected in front of the game world, that presents data without requiring users to look away from their usual viewpoints. x, 1

**keyframing** is the process of defining or creating an animation by placing keyframes. 17

**lattice deformer** is used to warp the space around a 3D object in order to animate or deform the object. 16

**level of detail** is a method used to change the detail of 3D objects, depending on how far the object is to the observer. x, 9

**mesh** is a collection of vertices, edges and faces, used to model a 3D object in computer graphics. 11

**polygon count** is how many polygons are used in a scene or a mesh. 12

**precomputed radiance transfer** is a method that recalculates complex light calculations offline to save time. x, 10

**rig** is the settings of an armature and how it is connected to a mesh. 16

**root bone** is the parent bone to all other bones of an armature. 17, 18

**UV-unwrap** is the process of transforming a 3D mesh into a 2D plane. 13

**virtual reality** is an artificial environment, which is experienced through stimulating senses, provided by a computer and in which one's actions can influence what happens in the environment. 1

# Acronyms

**CPU** central processing unit. 9, *Glossary*: central processing unit

**FOV** field of view. 23, *Glossary*: field of view

**GPU** graphics processing unit. 9, *Glossary*: graphics processing unit

**GUI** graphical user interface. 15, *Glossary*: graphical user interface

**HMD** head-mounted display. 1–4, 23, 30, *Glossary*: head-mounted display

**HUD** heads-up display. 1, 22, *Glossary*: heads-up display

**LOD** level of detail. 9, 10, *Glossary*: level of detail

**PRT** precomputed radiance transfer. 10, *Glossary*: precomputed radiance transfer

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Purpose . . . . .	2
1.3	Problem . . . . .	2
1.4	Scope . . . . .	3
1.5	Method . . . . .	3
1.5.1	Scrum . . . . .	3
1.5.2	Pre-study . . . . .	3
1.5.3	Validation of results . . . . .	4
1.6	Outline . . . . .	4
<b>2</b>	<b>Tools</b>	<b>5</b>
2.1	Game engine . . . . .	5
2.1.1	Game logic . . . . .	6
2.1.2	Material . . . . .	7
2.1.3	Particle system . . . . .	8
2.1.4	Performance . . . . .	9
2.1.5	Global illumination and precomputed radiance transfer . . . . .	10
2.1.6	Result and discussion . . . . .	11
2.2	3D modeling . . . . .	11
2.2.1	Polygonal modeling . . . . .	12
2.2.2	UV-mapping . . . . .	13
2.2.3	Character creation . . . . .	14
2.2.4	Game engine compatibility . . . . .	15
2.2.5	Results and discussion . . . . .	15
2.3	Animation . . . . .	16
2.3.1	Armature . . . . .	16
2.3.2	Motion capture . . . . .	17
2.3.3	Keyframing . . . . .	18
2.3.4	Simulation . . . . .	18
2.3.5	Animation in Unreal Engine . . . . .	19
2.3.6	Results and discussion . . . . .	20
<b>3</b>	<b>Game design</b>	<b>21</b>
3.1	Story . . . . .	21
3.2	Designing for virtual reality . . . . .	22
3.3	Game mechanics . . . . .	23
3.4	Frightening elements . . . . .	24
3.4.1	Jump scare . . . . .	24
3.5	Level design . . . . .	25
3.5.1	Level design process . . . . .	25
3.5.2	Level design in <i>End of the Road</i> . . . . .	26
3.5.3	Results and discussion . . . . .	26

3.6	Meshes and materials . . . . .	27
3.7	Sound . . . . .	28
<b>4</b>	<b>Result</b>	<b>29</b>
<b>5</b>	<b>Discussion</b>	<b>30</b>
5.1	Future development . . . . .	31
	<b>References</b>	<b>32</b>
	<b>Appendices</b>	<b>35</b>
<b>A</b>	<b>User testing</b>	
<b>B</b>	<b>Interview with Guru Games</b>	

# 1

## Introduction

The gaming industry is growing drastically, and is expected to grow even more in the coming years (Rob van der Meulen, 2013). For a new game to succeed, certain elements are required to capture the interest of the players. This can be done through, for example, a well defined story, intuitive controls, or a great sense of immersion. Immersion is of specific importance when creating games for one of the latest trends in the gaming industry: virtual reality. Achieving impressive immersion requires cutting edge rendering technology and beautiful art assets.

In this chapter we will first present a brief background of virtual reality and this thesis. We will describe the purpose of the project and define the problem statements. Furthermore, the limitations surrounding the thesis are presented, and finally, the last section outlines the structure of the thesis.

### 1.1 Background

Virtual reality as a concept has existed for over fifty years, and is based on trying to immerse a user in a virtual world (Payatagool, 2008). If done correctly, the user will feel like he or she is in another place than his or her physical location. Many attempts have been made to create and commercialize virtual reality hardware in the form of head-mounted displays (HMDs), the earliest being as early as 1968 (Sutherland, 1968). A more recent, but still unsuccessful, attempt was made by Nintendo in 1995 with their console *Virtual Boy* (Nintendo, 2015).

Immersion in virtual reality can be achieved in several different ways. Today it is generally accomplished by using sensors and stereoscopic displays. The displays are used to create a three dimensional image view and the sensors track the movement of the users and translate that movement into the virtual world. Further, sound through headphones is frequently used to elevate the experience even more (Baker, 2014). The stereoscopic displays and sensors are usually mounted on the head through a HMD. Companies such as *Sony*, *Oculus VR*, *Samsung*, and *HTC/Valve* are competing to make HMDs commercially available at an affordable price. According to a report from MarketsandMarkets (2014), virtual reality and augmented reality is expected to reach a market worth of \$1.08 billion by 2018.

One specific genre of gaming that depends significantly on immersion is the horror genre. According to Perron (2009, p. 20), the fear experienced is intensified when the player is fully immersed. Full immersion also minimizes the possibility of distractions from the outside world.

Developing games for virtual reality, as well as for the horror genre in general, is challenging because of the subjective nature of the game experience. Designing quality assuring tests that are objective and quantifiable can be immensely difficult. Additionally, virtual reality challenges some of the traditional concepts used when developing games, which makes it even harder to design a high-quality game. An example of such a concept is the use of a heads-up display (HUD) that many games use to display resources, such as health and ammunition. This is not favorable in virtual reality, and any required information should be incorporated into the game world (Oculus VR, 2015).

*End of the Road* is a game developed with the previously mentioned concerns in mind and is the basis of this thesis. The process and tools used to develop a horror game for Oculus Rift is discussed by presenting the development process of the game.

## 1.2 Purpose

The ambition of the project is to develop a horror game for the Oculus Rift virtual reality headset. The game should be immersive, frightening, realistic and intuitive. In addition, the game should be adapted to fit the technology of virtual reality. This thesis aims to describe how the development of such a game can be realized within a time limit of 20 weeks. It mainly concentrates on the procedure of the development and the tools that were used. Some technical aspects will be mentioned in association with the tools described. The purpose is to provide future developers with an example of a plausible approach on how to develop horror games for virtual reality headsets.

## 1.3 Problem

The overall problem is to find, implement and evaluate an approach for developing a horror game for virtual reality. This section underlines the problem statements associated with this thesis and a game development process which involves virtual reality. It includes challenges concerning tools, virtual reality, as well as level design and content creation.

Developing a realistic game in a short period of time requires a set of high-level tools. For a new developer entering the field of game development, and specifically virtual reality game development, having a complete toolbox and being sure that the tools are evaluated and working for the given purpose is crucial. This problem is narrowed down to the following questions.

- What game engine is suitable for developing a virtual reality horror game?
- What 3D modeling software is suitable to use when developing a virtual reality horror game?
- What animation software is suitable to use when developing a virtual reality horror game?

There are some specific challenges to consider when developing a game for virtual reality. Wearing a HMD can sometimes cause simulator sickness, which does not occur to the same extent while playing traditional games. Another challenge that needs to be taken into consideration is how to display necessary information to the player in an intuitive way. To make certain that *End of the Road* is well suited for virtual reality, the following questions need to be answered.

- What methods reduce simulator sickness sufficiently?
- How should information be presented to the player in virtual reality?
- What methods are suitable for providing the player with an immersive experience?

Since horror games, as well as virtual reality games, rely on immersion, the game world has to be coherent and realistic. A horror game additionally has to frighten the player. Therefore, the following questions need to be answered.

- What methods are suitable to frighten a player?
- What development process is suitable for creating the game world?
- What is a suitable method to create realistic game objects and characters?

## 1.4 Scope

This thesis discusses several different tools that can be used when developing a game. No emphasis is placed on building or discussing how the tools are implemented or how they could be implemented. The thesis is limited to already finished and well-tested solutions.

The thesis presents tools and techniques that can be used to develop a game. Even though alternatives are discussed, the scope of this thesis is to present one useful approach rather than focusing on the differences between approaches.

Different HMDs and hardware can be used to achieve immersion in virtual reality. This thesis is limited to Oculus Rift.

Frights are subjective, which makes it complex to measure and benchmark such experiences and feelings. Therefore, evaluating how frightening a scenario relies solely on the subjective impression of the development team and players.

## 1.5 Method

To manage the complexity of developing *End of the Road*, some organizational structuring was required. This section explains the chosen workflow and how tasks were distributed within the team. The agile development framework Scrum has been used to structure the development of *End of the Road* (Scrum.org & Scruminc, 2014). Section 1.5.1 describes why Scrum was chosen and how it was applied.

To make sure the project focused on the right research areas, a pre-study was performed. The pre-study provided insights into game development and a starting point for the creation of *End of the Road*. This pre-study is described in Section 1.5.2. Results were validated by performing user tests, as discussed in Section 1.5.3.

### 1.5.1 Scrum

Two particular aspects were of significant importance when determining the workflow for *End of the Road*. The complete scope of the game was not fully defined at the beginning, and therefore an adaptable workflow was required. Additionally, considering that the team had little or no experience with neither game development nor the horror genre, the team considered the task complex. According to the Scrum guide, the purpose of Scrum is to simplify the development of complex and adaptive problems (Scrum.org & Scruminc, 2014). Therefore, Scrum appeared suitable.

The Scrum setup was organized as follows: the development team was formed by the entire group. The group also acted product owner. Instead of assigning a Scrum master, every team member was responsible for making sure the Scrum framework was followed as accurately as possible.

To increase accessibility, a virtual Scrum board was used. The Scrum board actions were posted to all the members through a chat client, to maintain a high level of transparency. The length of each sprint was two weeks. Every sprint was evaluated by the team during a sprint review. In addition to the sprint review, sprint planning meetings were held to plan the two upcoming weeks.

### 1.5.2 Pre-study

Because of the previously mentioned lack of experience, a research phase was conducted at the beginning of the project. This included a visit to an experienced horror game developer, and investigating if and how fear can be objectively measured. The phase also included studying the concept of virtual reality.



Before starting the development of the game, the team had the ambition to visit a developer with experience in both virtual reality and horror games. The purpose of this visit was to gain inspiration and get a reference workflow for game development. A game development studio named Guru Games was found suitable. The studio has developed a horror game, Medusa's Labyrinth, which supports Oculus Rift (Guru Games, n.d.). The team visited Guru Games' office, and an interview with three of their employees was held. A summary of the interview can be found in Appendix B.

To be able to design an immersive and well-fitted game for virtual reality, a literature study was made on virtual reality in games and the concept of immersion, as well as HMDs.

### 1.5.3 Validation of results

To make sure the game was both enjoyable and frightening, user testing was essential. By arranging sessions where users tested the game, while the development team observed their reactions, areas in need of improvement could be identified. These areas included, for example, what the user experienced as difficult, and how well the player progressed in the game.

Research was made on how to measure the level of fear experienced by a test subject. In an article written by Gera (2014), Supermassive Games' managing director Pete Samuels is cited. Samuels states that fear can be quantified by measuring the change of moisture content on people's skin. However, we did not have access to the necessary equipment to use this technique.

Another method that was considered was measuring pulse on the subject while testing. This method was discouraged by Guru Games, and was therefore not used (Ström, 2015). A test person is often slightly nervous before the test is to be performed, and therefore, has an elevated pulse. After the test person has played for a while, he or she will feel more comfortable and their pulse will drop again. Vachiratamporn, Legaspi, Moriyama, Fukui, and Numao (2015), have measured how different states of emotion can affect how frightening an event is perceived to be. They had to use expensive and complex equipment, none of which we have access to. Due to these restrictions, the measurement of fear using pulse was not used.

We decided that the game is assumed to be scary when at least 3 out of 5 test subjects considered the game frightening. Test subjects were chosen randomly at computer game events at Chalmers University of Technology. To quantify the result from the user-testing sessions, a questionnaire was produced, with quantifiable scales, which the users filled in after a play-through of the game. The feedback from the responses could then be used in order to improve parts of the game that were perceived as insufficient according to the users. The question form and the collected results can be found in Appendix A.

## 1.6 Outline

According to our problem statement, many different technologies are evaluated. These tools, their usage in the development process, and the results are what Chapter 2 will describe.

Continuing with Chapter 3, the game design, such as story, game world and game mechanics will be discussed. This chapter also shows how we used the tools in the previous chapter to achieve satisfying results with the developed game.

The final chapters include results, discussion, and conclusions that were made once the project was finished.

# 2

## Tools

Developing a game is time-consuming and requires many different technologies. By using several tools available on the market, cutting down development time, and therefore development costs, is possible (Lecky-Thompson, 2007). This chapter will consider what applications could be useful while developing a virtual reality horror game.

The game engine is the software backbone of a game, connecting the various game components. Different game engines will be compared, and the functionality that the chosen game engine provides will be explained. Continuing with 3D modeling, which is needed to create game objects perceived by the player, the section clarifies how a 3D model is created, and what tools the development team used. Following is a section introducing how animations are represented, and how these were created for *End of the Road*.

### 2.1 Game engine

There are several different types of libraries and game engines that can be used to develop a game. Since the project is focused around developing a game for virtual reality in a short period of time, a game engine with extensive features had to be chosen. We chose to investigate four extensive and popular game engines that are available either commercially or free.

Game engines:

- Unity from Unity Technologies
- CryEngine 3 from CryTek
- Source from Valve
- Unreal Engine 4 from Epic Games

For this project the game engine had to have Oculus Rift support, since the project was centered around creating an Oculus Rift game. We perform the evaluation under the assumption that all four engines will produce a high level of graphical fidelity, enough to keep the player immersed.

In Table 2.1 it is shown that all the engines provide support for Oculus Rift. All the engines except Source support some kind of scripting language. Having the alternative to prototype in a user-friendly scripting language is preferred. Comparing the platforms that the different development tools support, Unreal Engine 4 and Unity have the advantage. Moreover, the development team had to be able to run the tools on their operating systems. The team used computers running Windows as well as OS X, which excluded the use of Source and CryEngine 3.

From the aspects presented in Table 2.1, Unreal Engine 4 and Unity are preferred. To make a decision between the two, literature and articles were studied. It was shown in a survey done by DeLoura (2009) that out of a hundred game executives, 60% prefer to use Unreal Engine compared to just under 20% for Source, and under 10% for CryEngine. Unity had a notably low score. Based on the technical aspects of the game engines, in relation to the team, and the result of

	<i>Unity</i>	<i>CryEngine 3</i>	<i>Source</i>	<i>Unreal Engine 4</i>
<b>Oculus Rift</b>	Supported	Supported	Supported	Supported
<b>Platforms</b>	Windows, OS X	Windows	Windows	Windows, OS X, Linux <sup>1</sup>
<b>Language</b>	C#, UnityScript, Boo	C++, LUA	C++	C++, Blueprint

Table 2.1: Overview of game engine characteristics (indiedb, 2015).

the literature study, Unreal Engine 4 was deemed most suitable for this game project. Henceforth when discussing game engine features we refer to Unreal Engine 4.

In the following sections, five of the engine’s features are presented in more detail. The first section explains how game logic was implemented with the engine’s built in scripting language. The second section describes the basics of materials and how the engine provides help with visualizing them. The third section outlines the construction of particle systems in the game engine editor. This is followed by discussions regarding performance issues, as well as how lighting is computed within the engine. Finally, an overall discussion regarding the game engine is presented in Section 2.1.6.

### 2.1.1 Game logic

Unreal Engine 4 provides two options for implementing game logic. The engine supports interpreting code written in C++, and a graphical scripting language called *Blueprint* (Epic Games, 2015c). The possibilities of what can be achieved through the different approaches are very similar, but there are a few distinct differences. The most significant difference is that C++ consists of text-based code, while Blueprint is visualized as a graph containing nodes and edges. Furthermore, Blueprints can be made within the game engine while C++ needs a separate text editor. The Blueprint visual scripting language was chosen for this project, and the decision was based on two aspects. First, the advantage of being able to script in the game engine, which also offers the opportunity of graphically following a graph’s execution flow during gameplay. Secondly, Blueprint provides the possibility of prototyping behaviors in a much less time-consuming manner (Ström, 2015).

The fundamental parts of the Blueprint scripts are graphs, consisting of nodes and edges. A graph is similar to a class in any object-oriented language. Nodes represent behaviors and edges visualize the game’s possible execution flow. Any desired number of nodes can be placed in a graph. The nodes can then be connected with edges to achieve an intended behavior. Figure 2.1 illustrates one example of a Blueprint.

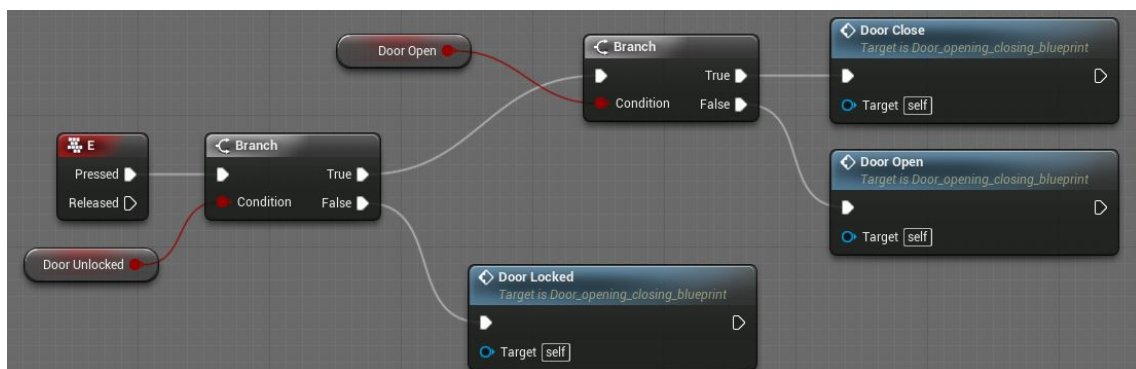


Figure 2.1: A script for opening and closing doors in the Blueprint editor.

The choice of using Blueprint was beneficial in several ways. The Blueprint system provides real-time flow visualisation. During run-time, the edges turn red when the corresponding game flow

<sup>1</sup>Experimental, and managed by members of the Unreal Engine community.

occurs. This feature makes it possible to debug any scripts. Even though this worked fairly well, problems arose when the game and the graphs increased in complexity. In a traditional debugger, it would be possible to step through the code and find the location of the error. With the visual system, there are difficulties in pinpointing the exact problem and locating in what graph the problem is contained.

The decision of using only Blueprints resulted in one significant restriction. The Blueprint scripting language consists of pre-defined behaviors and functions. These can be seen as black boxes that are closed for modification. The consequence of this abstraction is that the implementation details are hidden and cannot be controlled. Developers can thereby not optimize the components that are used in the game logic.

### 2.1.2 Material

Materials are used to provide in-game objects with textures and properties such as transparency and glossiness. There are several material editing software applications, such as Maya LT, 3ds Max and Blender. Unreal Engine also provides a built-in editor to create and edit material shaders. The possibility to create materials by writing the shaders directly using a more low-level graphics library also exists. This is not very user friendly, and the development team deemed this method to be too time-consuming. To ensure compatibility with the game engine, and avoid time-consuming methods, the material editor in Unreal Engine 4 was used to create all materials during the development.

Unreal Engine uses a physically based material system, which means that every material is defined by physical properties from the real world (Pharr & Humphreys, 2010). This system simplifies the translation of the material to its mathematical representation, in the game engine. The built-in editor ensures that all the material properties and features can be utilized.

The material editor in Unreal Engine is structured in such a way that the development team could directly apply the workflow developed for Blueprint editing on the material editing. The similarities between the systems can be seen in Figure 2.1 and Figure 2.2. The same node-based system is used with a set of functions that only work for materials. In addition, the editor has a live preview window which makes the workflow almost identical to the Blueprint scripting in terms of live feedback.

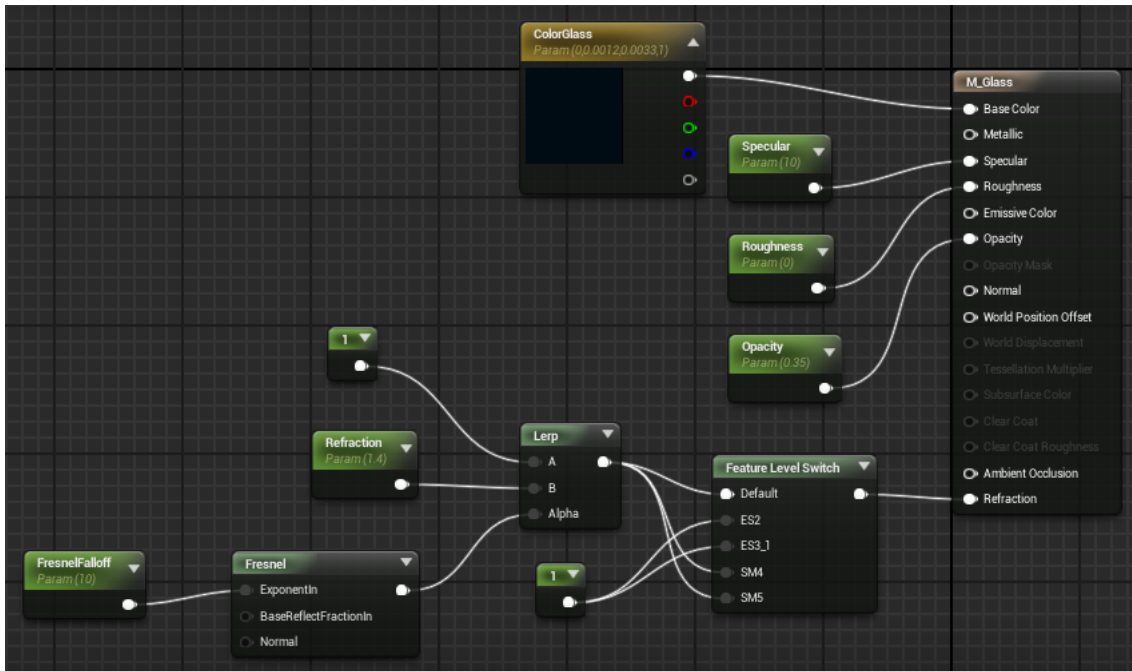


Figure 2.2: A material represented inside the material editor.

In Figure 2.2, a simple glass material is modeled. Different computations are represented by nodes. Input and output values are connected by edges. By using the material editor when developing materials, *End of the Road* achieved realistic materials suitable for the environment.

### 2.1.3 Particle system

To simulate effects like fire and dust, a particle system can be utilized. Many game engines implement their own version of a particle emitter editor, but some standalone alternatives exist, for example RigzSoft and PopcornFX. The advantage of using a built-in solution is the same as stated for materials in Section 2.1.2, that is compatibility. Unreal Engine contains a particle emitter editor and due to the advantage of compatibility, this was used to develop particle effects. The particle editor interface can be observed in Figure 2.3.

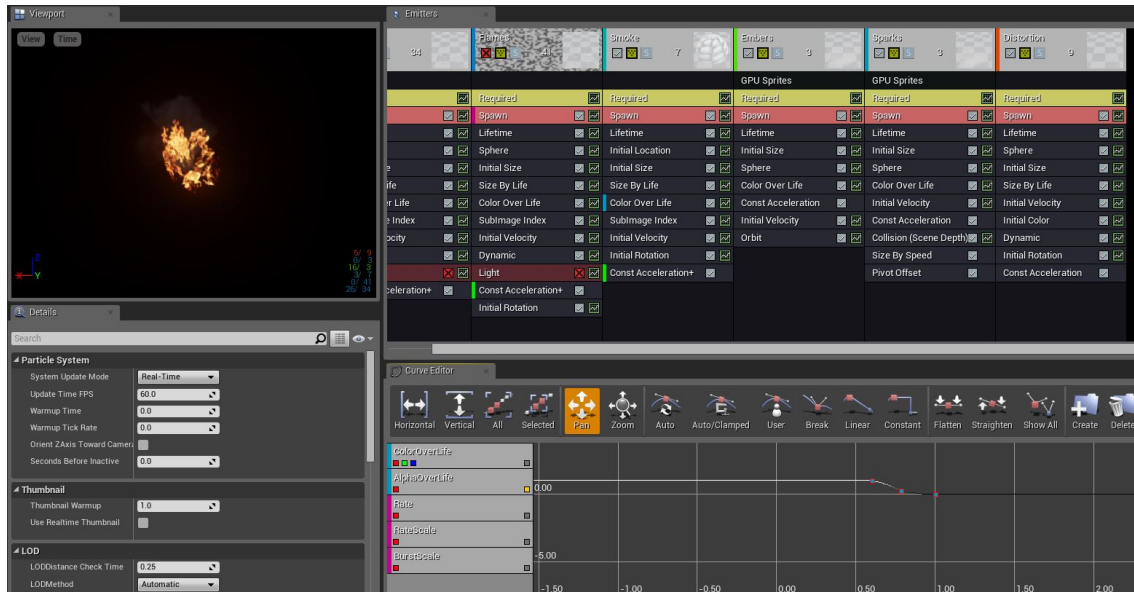


Figure 2.3: The particle editor in Unreal Engine.

The game uses a particle system to, for example, simulate rain in the game world. In a particle system, particles are randomly generated from a point or surface and move in a random speed and direction. Their movement can then be affected by physical forces, such as gravity (Chopine, 2012). Particles can have a limited life span, which is useful for creating effects such as rising smoke or fire.

An advantage of using the Unreal particle editor, besides compatibility, is the ability to have particles be rendered and buffered on the graphics processing unit (GPU) instead of doing the calculations on the central processing unit (CPU). By doing this, a game can have hundreds of thousands of particles efficiently rendered on the screen (Epic Games, 2015d).

### 2.1.4 Performance

During the development of *End of the Road*, poor performance was an issue. Therefore, several options to solve this problem were investigated. According to Epic Games, different performance problems can be reduced by these different solutions (Epic Games, 2015e), (Epic Games, 2015j).

- Using level of detail (LOD)
- Adjusting tessellation factor
- Simplifying materials
- Reducing the number of seams in the UV-maps
- Reducing the number of lights using dynamic shadows

By using the Unreal Engine 4 GPU profiler, the cause of the performance issues can be detected. According to the GPU profiler, lighting computations and a too high number of visible vertices seemed to be the most significant issues with the performance in *End of the Road*. Epic Games suggest that LOD should be used if too many vertices is the problem. If lights are the problem, the use of real-time shadows and dynamic light sources should be reduced as much as possible.

LOD reduces the number of polygons in 3D-objects distant to the viewer (Luebke, 2003). Since this problem was introduced when vegetation was added to the game, and vegetation includes a large number of polygons, implementing LOD should replace many vegetation meshes with less detailed meshes. Replacing meshes this way reduces the number of vertices, and therefore the

performance of the scene should increase. An example of an object with different LODs can be observed in Figure 2.4

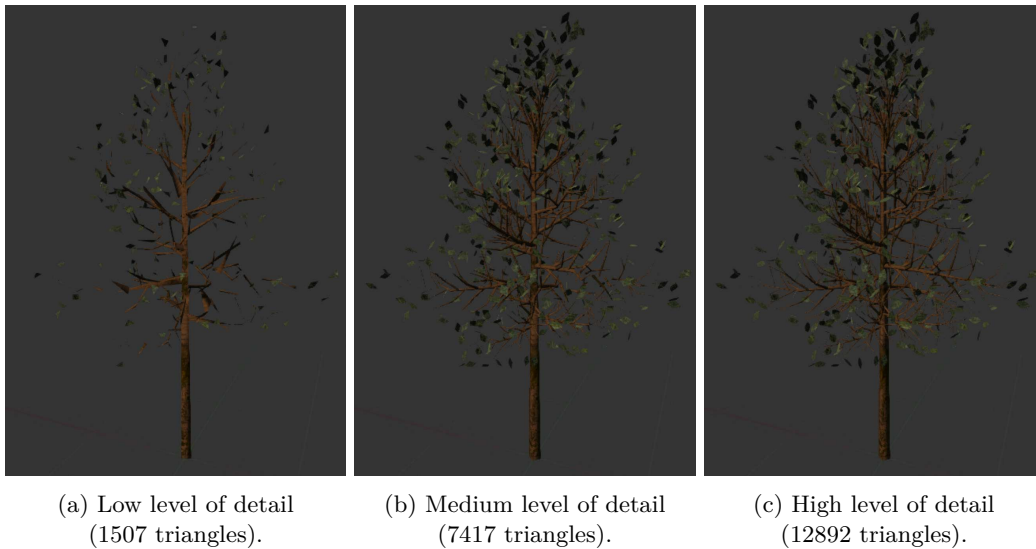


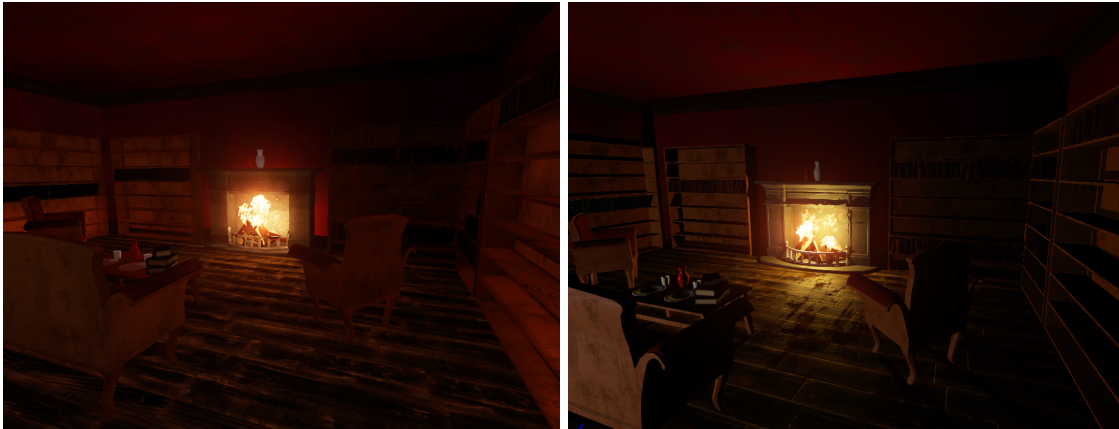
Figure 2.4: Level of detail for trees in *End of the Road*.

According to Epic Games (2015j), lights with real-time shadowing require 20 times more computing performance than lights without real-time shadowing. By using static lights, eliminating the use of real-time shadowing while still maintaining shadows is possible. Static lights use precomputed radiance transfer (PRT), which will be described in detail in Section 2.1.5.

### 2.1.5 Global illumination and precomputed radiance transfer

Global illumination is a concept for displaying indirect lighting in 3D computer graphics. For example, when light bounces to illuminate a whole room, this can be modeled with global illumination (Jensen, 1996). Simulating this can be done with, for example, a technology called *photon mapping* that traces each photon from the light source, and calculates how that photon bounces in the room, which in turn determines the brightness and color of each surface in the room. Since this method is very close to what happens in reality, an extremely realistic effect is observed. The problem with this method and many other methods is that it takes too much time to calculate indirect lighting in real-time to achieve global illumination. There are alternatives that enable dynamic global illumination which can produce realistic effects in real-time (Epic Games, 2015h). However, our game engine of choice does not fully support this technology yet.

One way to solve this problem is by computing global illumination beforehand (Epic Games, 2015g), this is called PRT. With PRT, we can have realistic indirect lighting. However, this introduces another problem, as no properties can be changed for any light without the need of recalculating the PRT. This means that only static lights can have realistic indirect lighting.



(a) Fireplace rendered with PRT.

(b) Fireplace rendered with only direct lighting.

Figure 2.5: Difference between using PRT and only direct lighting

Figure 2.5 demonstrates the effect of using PRT. Figure 2.5a has PRT, which is visible by the indirect lighting spread all over the room. Figure 2.5b however, does not have indirect lighting, and the visible light comes from direct lighting only.

### 2.1.6 Result and discussion

The choice of using Unreal Engine as a base for the development of the game granted the team a set of high-level tools with full compatibility. Since the workflow is almost identical between the different tools, the development pace could be kept high and ensured that team members could discuss solutions between different domains. While Unreal Engine was beneficial in many ways, the team also experienced many bugs. The engine crashed frequently, which was frustrating and often disturbed the workflow.

When comparing the features the engine provides to the lack of entirely finished features, the team agrees that the benefits outweigh the disadvantages. The possibility of being able to work in an abstraction level that does not demand implementation of, for example, physics and lighting, enabled the team to create a more extensive product. Therefore, Unreal Engine proved to be an extensive game engine with all required components for developing a virtual reality horror game.

## 2.2 3D modeling

In order to make the game world interesting, 3D objects are needed. Two different methods of obtaining these assets were considered. The first and simplest option is to download them from the Internet, as there are several communities devoted to creating and sharing meshes (Archive 3D, 2015)(The Free 3D Models, 2015). The second alternative is to create each object by hand, using 3D modeling software.

When downloading from the Internet, different licenses apply to the assets. These licenses may restrict the usage of the asset for certain purposes. For example, a license may allow a mesh to be used in a private project but not in a project that will be published or in other ways released publicly. Another aspect to consider when downloading assets is that the development team would have limited or no control over the mesh design. Mesh design was incredibly important since it was decided early on that *End of the Road* should be created in a certain graphical style. When relinquishing control over the appearance of assets, creating a coherent game world will be more complicated. Should the team instead choose to create all assets, a significant amount of additional



work would have been required. Considering all of the above factors, it was decided that creating our own assets would be most beneficial.

There are many programs that can be used to create 3D assets, for example Blender, Maya and 3ds Max. Maya and 3ds Max are both developed by Autodesk and are commercial applications available for a fee (Masters, 2015). Blender, on the other hand, is a completely free application. Even though Blender does not include all the features of Maya and 3ds Max, it has sufficient capabilities to create any 3D assets that might be needed to develop *End of the Road*. Additionally, according to Masters (2015), Blender is known for a modeling workflow that is intuitive and easy to understand, and there is an extensive, supporting community. After considering all of these factors, we decided to use Blender.

Further into this section, techniques for creating 3D models are presented and discussed. First, the subject of polygonal modeling is presented, followed by a section about UV-mapping. Thereafter the creation of characters is described. Furthermore, a section outlines the compatibility between Blender and Unreal Engine. Finally, the section ends with results and discussion of the used 3D modeling software.

### 2.2.1 Polygonal modeling

A polygonal model or mesh is a 3D model that is created using a computer and can be used in several different settings (Shene, 2010). A mesh is made up of vertices, edges and faces. Vertices are points in a 3D space, which are connected by edges to form faces, or surfaces.

There are many different workflows that could be used when creating a mesh in Blender. If the sculpting mode is applied, a form with many vertices is used as a base. Then, several different brushes are used to manipulate or sculpt it. The brushes have different results when applied to the existing mesh. They can, for example, inflate, deflate, smoothen or flatten its surface. As we are creating a game, we do not want the final meshes to be too complex in terms of polygon count (the number of triangles that make up the mesh). Sculpting mode requires a mesh rich with vertices and triangles, it is great for sculpting organic objects such as human faces and bodies, but is inappropriate for symmetrical objects like chairs. Most of the meshes in *End of the Road* are furniture, thus the sculpting mode that Blender provides did not seem suitable to create the 3D assets for the game.

Another mode that can be used in Blender is called *edit mode*, which is visualized in Figure 2.6. In this mode, a regular form such as a plane, square, circle or cylinder can be used as a base. In this base, vertices, edges and faces are then manipulated to create the desired form. They can be moved, scaled or rotated one or more at a time in the 3D space. In addition to moving, scaling and rotating, Blender contains several other ways to manipulate a mesh. The edit mode provides a straight-forward workflow that can be used to create anything that might be needed in *End of the Road*.

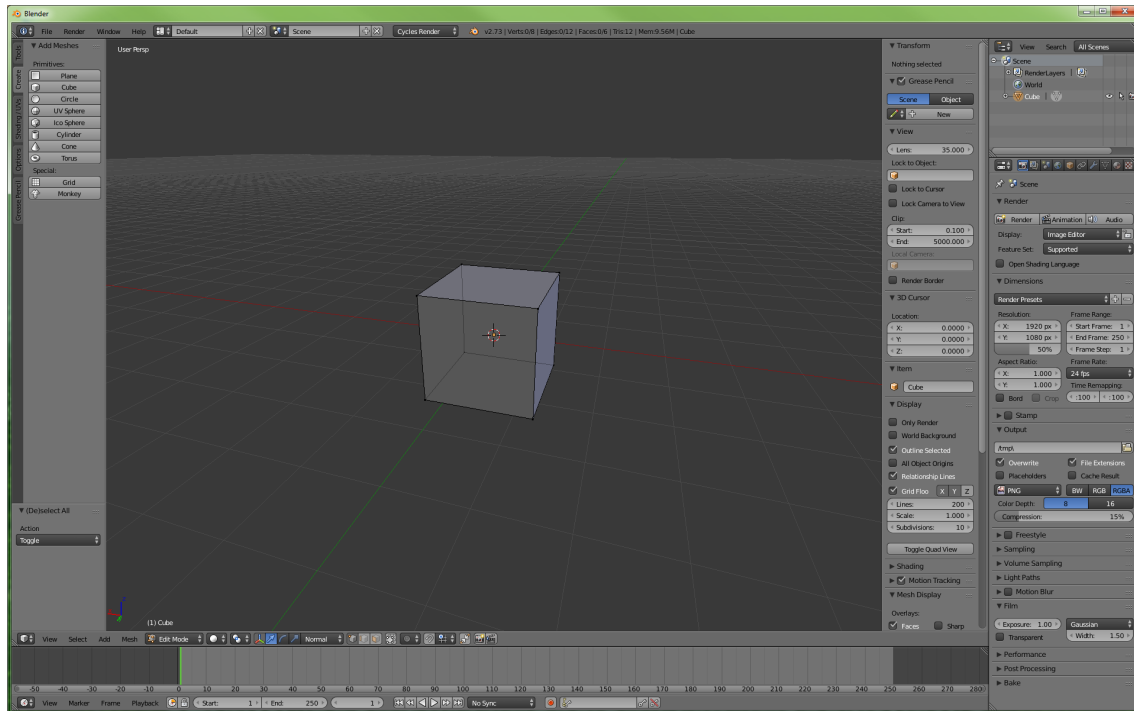


Figure 2.6: Edit mode in Blender.

After considering the above, it seemed that using the edit mode would be a more efficient process for creating 3D assets. Therefore we chose to create all of our custom-made meshes using this approach.

## 2.2.2 UV-mapping

A UV-map is essentially a 3D mesh that has been transformed into a 2D plane, in order to define how textures should be applied to its surface. There are several ways to UV-unwrap a mesh in Blender. The procedure that we have used most is where the mesh is inspected, and the appropriate edges are marked as seams. These seams tell Blender that the faces connected by that edge do not need to be connected in the UV-map. This is the procedure that has been used in Figure 2.7. On the right side we can see that the edges that are marked as seams appear in a red color.

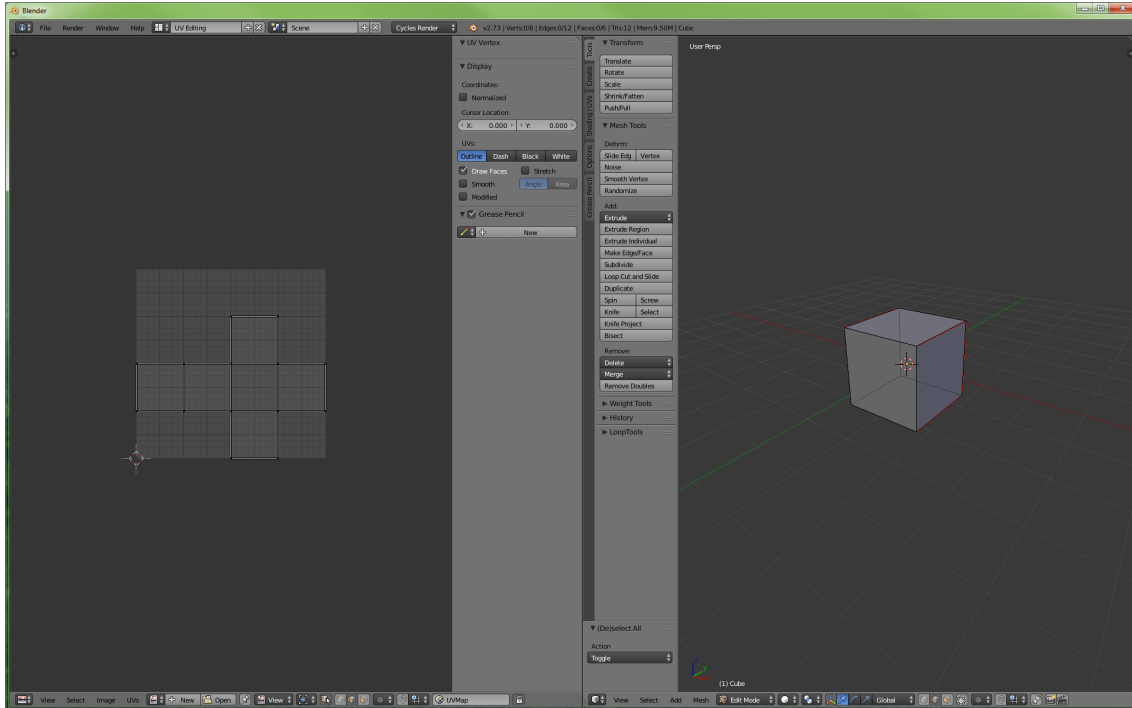


Figure 2.7: The mesh of a cube with corresponding UV-map, visualized in Blender.

Another procedure that could be used to UV-unwrap a mesh in Blender, is referred to as *Smart UV project*. In this procedure, Blender determines on its own which edges to mark as seams and unwraps the mesh from there. A third procedure called *Lightmap Pack* simply marks all edges as seams, thus making each face of the mesh a separate element in the UV-map.

### 2.2.3 Character creation

We want the player to find the game world believable, which entails that the characters need to be realistic and anatomically correct. When the character design process was researched, we found out that it is a complex task, and concluded that it does not fit into the time frame of this project (Maraffi, 2003). Therefore, a solution that could speed up this process had to be considered for *End of the Road*. Multiple such solutions are available, such as MakeHuman, Poser and DAZ Studio. Out of these three options, MakeHuman is the only alternative that is available for free. In addition, MakeHuman is open source and provides a vast set of features necessary for character creation. Thus, MakeHuman was established as the preferred solution.

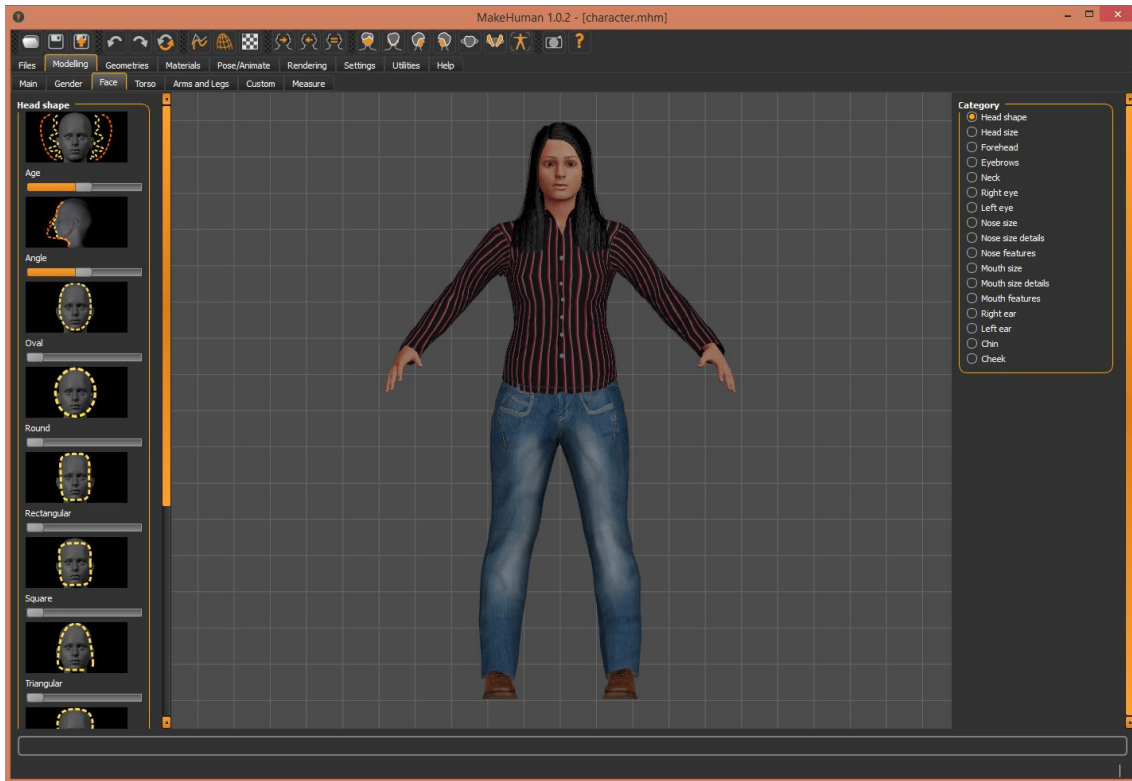


Figure 2.8: The user interface of MakeHuman.

MakeHuman has a user-friendly graphical user interface (GUI), as seen in Figure 2.8, where the user can adjust a wide selection of properties ranging from age, gender and weight to details of each body part (MakeHuman team, 2015a). There is also an option to add an armature for the character. For an explanation of armatures, see Section 2.3.1. A character created in MakeHuman can be exported to a custom file format intended for importing characters into Blender. Once imported into blender, the characters can be edited further to customize their appearance.

## 2.2.4 Game engine compatibility

When creating a mesh in Blender, the exported model will be saved in a format that is not supported by Unreal Engine. However, there is an add-on for Blender which enables exporting an object in a format supported by the game engine. By using this add-on, Blender is compatible with Unreal Engine.

Unreal Engine 4 supports Nvidia APEX PhysX, a solution for advanced physics simulations, for example cloth animations (Epic Games, 2015b). However, Blender is missing support for this plugin, which means that if a game needs realistic real-time physics-based animations of cloth, 3ds Max or Maya would be the preferable choice. For the scope of this project, however, such animations were considered too complex and were not be implemented.

## 2.2.5 Results and discussion

All custom 3D models have been created using Blender and MakeHuman, exported to a suitable format and then imported into Unreal Engine 4. In total, over 150 models have been created for use in *End of the Road*. Blender has proved to be sufficient for creating the meshes for the game, and the created meshes have turned out satisfactory. It is possible that either Maya or 3ds Max, which were mentioned previously under Section 2.2, has better compatibility with Unreal Engine 4.

However, for our purposes, Blender has provided every required features. Three different examples of finished meshes can be seen in Figure 2.9

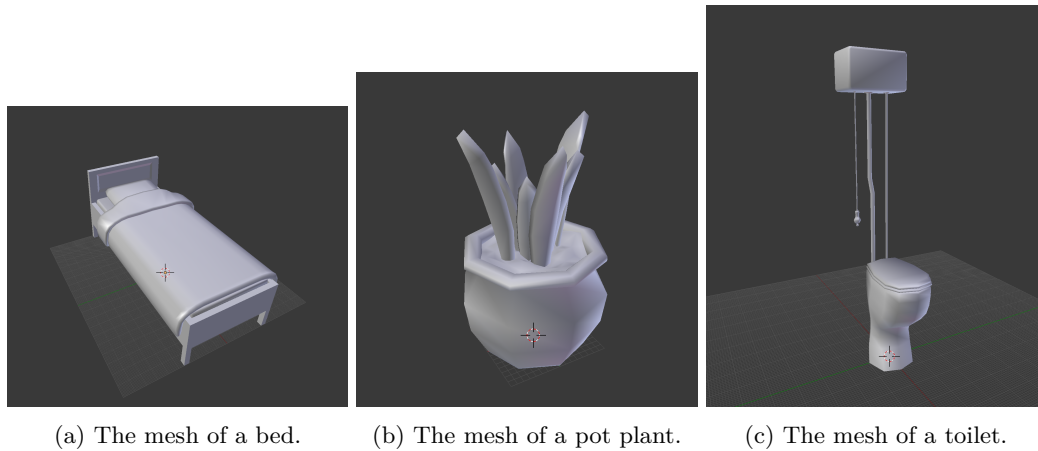


Figure 2.9: Examples of finished meshes in Blender.

Creating authentic humans in a general-purpose 3D modeling application such as Blender is a very complex task, and would likely have taken a large amount of time. MakeHuman has made this process significantly less time-consuming, and the resulting characters had an anatomically correct look. The ability to edit the characters in Blender before importing to the game engine was convenient as well, and was primarily used to correct errors in the mesh, such as the body clipping through the clothes.

## 2.3 Animation

Objects, such as characters, in games need animations in order to appear to be moving. A large number of applications are available to achieve this. A few of them are 3ds Max, Maya, Blender and Cinema 4D. In the evaluation of the available animation software products, compatibility with the characters created with MakeHuman was required. One useful method for creating realistic animations is motion capture, as described in Section 2.3.2. To enable the use of this technique, the capability of importing motion capture data and applying it to the skeleton of the character was needed. MakeHuman provides several plugins for Blender intended for editing imported characters. One of them is MakeWalk, which can be used for importing motion capture data and applying it to the armature of the character (MakeHuman team, 2015b). The plugins do not work with applications other than Blender, and therefore Blender was chosen for final editing of animations. The animation techniques and how they were used in the creation of *End of the Road* are described in the following sections.

### 2.3.1 Armature

In order to create animations, a rig is needed. A rig can consist of a skeleton, lattice deformers, blend shapes or a combination of these (Chopine, 2012). MakeHuman is capable of creating a rig for the character mesh, which can be used to create animations when the mesh is exported to external software (MakeHuman team, 2015d). Animations for the characters in *End of the Road* were created using skeletons, since this is what the rig created by MakeHuman uses (MakeHuman team, 2015c). Therefore, this section will focus on skeletons. A rig is built up as a hierarchy of bones and joints (Chopine, 2012), as can be seen in Figure 2.10. A joint marks a part of the model that can be bent or rotated, and a bone shows the connection between two joints.

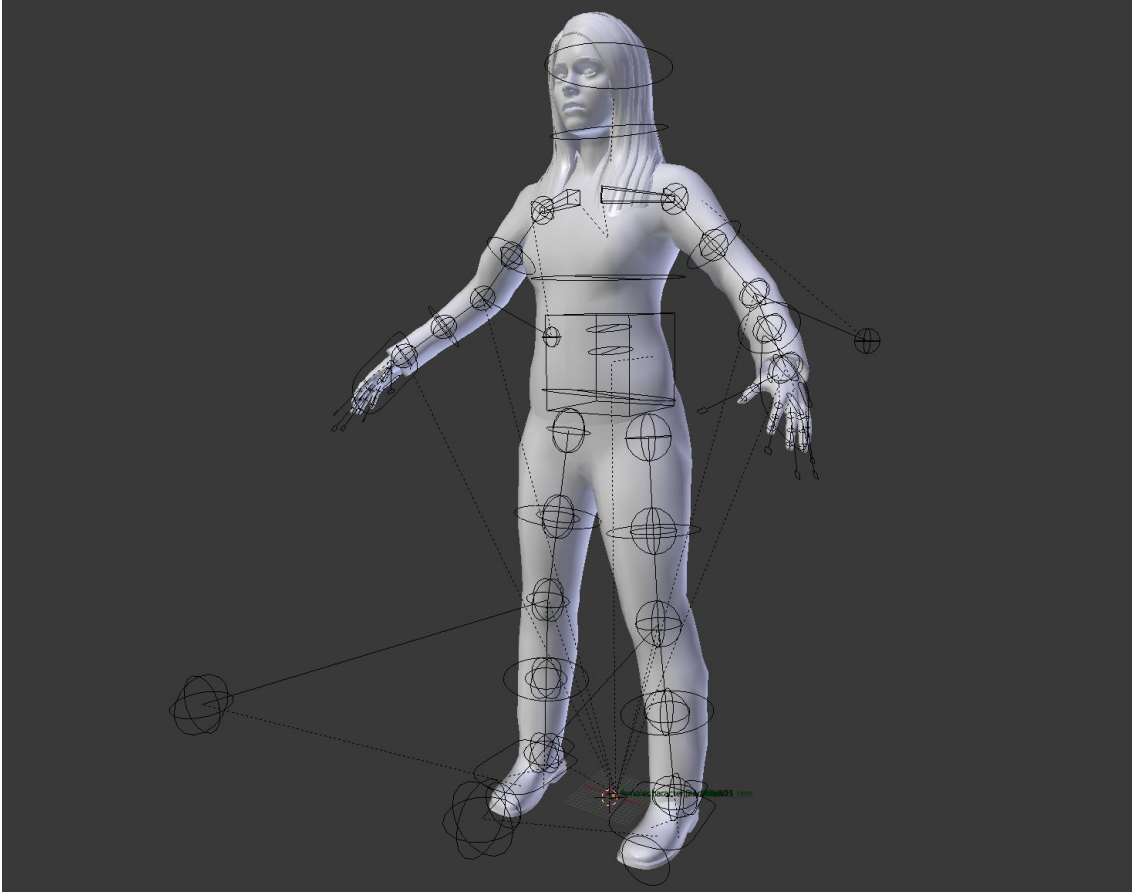


Figure 2.10: A character viewed in Blender with the armature visible.

### 2.3.2 Motion capture

Motion capture is a technique used to record the movement of a performer in order to create an animation (Chopine, 2012). One method that can be used to create motion capture data involves a performer, who is filmed while wearing a tight suit with markers on it, that can easily be distinguished by the recording software. The 3D positions of these markers are calculated and saved, and can then be used to create an animation for a computer-generated character. Using motion capture can be faster than manually keyframing the whole animation, but the technology has limitations and an animation based on raw motion capture data is not complete. The body of the performer can be different from the virtual character, and the data may contain errors. Therefore, the animator needs to perform some work to adjust and clean up the data. They may also want to adjust the animation for creative reasons, such as adjusting the poses or exaggerating certain movements.

Most animations were created based on motion capture data, in order to make them look as realistic as possible. Some initial editing of the motion capture data was done before importing it into Blender. The data used is in the bvh file format, so the application used needed to support this format. The required features were cropping of animations, removal of movement of the root bone and blending the end and beginning of an animation to make the transition smooth when the animation is looping. A tool named *Bvhacker* had the required features and was therefore used. Bvhacker is an open source tool intended for viewing and editing animation data with the bvh file format (Wooldridge, 2015).

Freely available motion capture data from the Ohio State University Advanced Computing Center for the Arts and Design (Ohio State University, 2015) and Carnegie Mellon University Motion

Capture Database (Hahne, 2015) was used. The motion capture data needed to be edited in order to be used in a game. For this purpose, the tools Bvhacker and Blender were used. To make an animation loop seamlessly, the first pose must appear as if it follows naturally after the last pose, in order to prevent a hitch when transferring between these two poses. Therefore, it is necessary to remove movement of the root bone, so that the position of the character does not change while the animation is running. These steps were done in Bvhacker. Blender also has a feature that can be used to make looping smoother, but it produced errors such as shaking and hitching movements of some body parts, and for that reason we decided not to use it.

After the initial editing in Bvhacker, the animations were imported and applied to the skeleton in Blender, using the MakeWalk plugin. The import process was found to frequently produce errors. The legs of the character were typically crossed and clipped through each other, the arms clipped through the body and some of the animations had hitches. The animations had to be manually edited to remove these problems. They were further edited to fit the way we wanted the character to move in the game. For example, we wanted the character to have a more upright posture and less movement of the head and neck in order to not move in front of the camera.

The application of the animation to the skeleton in some cases did not work at all, and for the animations that did work, the legs of the character were typically crossed and clipped through each other, the arms clipped through the body and some of the animations had hitches. The animations had to be manually edited to remove these problems. They were further edited to fit the way we wanted the character to move in the game. For example, we wanted the player character to have a more upright posture and less movement of the head and neck in order to not move in front of the camera.

### 2.3.3 Keyframing

Keyframes are used to define properties such as position, rotation and scale of objects at certain times during the animation (Chopine, 2012). In 3D modeling software, keyframes are often displayed as a graph, with time on the horizontal axis and the property being keyframed on the vertical axis. Movement between keyframes is calculated by interpolation, which can be either stepped, linear or curved. Curved interpolation was used, since it makes movement between keyframes smooth, and is therefore suitable for creating motion that looks natural.

As previously stated, most animations were created based on motion capture. For some animations involving unusual movements, we were not able to find motion capture data that fit our purposes. Therefore, a few animations were created by manually keyframing. It proved to be significantly more difficult to make these animations look sufficiently realistic. The animations where this method was applied were quite simple, so the technique was considered sufficient in these cases. However, for complex movement such as walking, which involves many parts of the body, realistic animations might not have been possible to create within the required time frame.

### 2.3.4 Simulation

Simulation is an animation technique where movement is achieved through physics-based calculations (Chopine, 2012). This technique is frequently used to animate explosions, moving cloth, hair and particles. Cloth can, for example, be simulated by using particles to deform the mesh based on physical forces.



Figure 2.11: A chandelier in *End of the Road* after crashing to the ground.

In Unreal Engine, a mesh can be animated through physical simulation using a physics asset. The physics system in the game engine was, for example, used in a scene where a chandelier falls from the ceiling, crashes into the floor and shatters into pieces, as seen in Figure 2.11. It would also have been possible to create meshes for the individual pieces and animate their movement by keyframing. However, this method would have consumed a considerable amount of time. The chandelier asset was created in Blender and a destructible mesh was created for it in Unreal Engine, in order to define how it would split up into pieces when hitting the floor. The movement of the pieces was then simulated by the physics system.

An attempt was made to use physics simulation to make a character hang from a rope and fall in a natural way. However, the simulation did not produce the expected result. Collision between different body parts made them stretch and deform unpredictably. This was solved by manually creating an animation in Blender.

### 2.3.5 Animation in Unreal Engine

The animation system in Unreal Engine 4 has a wide range of tools that can be used to control animations (Epic Games, 2015a). Animation sequences contain keyframes at different points in time and can be imported into the engine from an external application. Bones can also be controlled directly in animation Blueprints. Blend spaces are assets used to smoothly blend between animations based on one or two input values, such as speed and direction. A blend space is represented as a graph with each axis corresponding to an input value. Animation sequences are placed on the graph and the blend space automatically calculates the amount of influence the sequences should have and creates a final pose based on the result. It is also possible to directly control blending of animations through Blueprints.

Animations are controlled by an animation Blueprint. The animation Blueprint uses skeletal controls, blend spaces, and animation sequences to compute the final pose of an actor for each frame. It can also update the values used in blend spaces, and include logic for *animation notifies*. Animation notifies are special events used to define certain points in an animation sequence where an event should occur, such as a footstep sound or a particle effect.

For the player character, animations for standing still and walking forward, backward, left and



right were created based on motion capture data. An animation Blueprint and a blend space were created, using speed and direction of movement of the player character as input. The blend space was used for selecting which animation to play and to make smooth transitions between animations when these values change. It was also used for animating the character when walking in a direction other than the ones we have animations for. When the player walks in a diagonal direction, the blend space computes an animation where the two closest animations have equal influence. After adjusting the animations to consist of the same amount of steps and use approximately the same start position, this method yielded the intended result of smooth animation in all directions.

The Matinee tool can be used to animate an actor over time, for the purpose of creating cinematic sequences, called *matinees* (Epic Games, 2015i). Matinees have been used to implement some of the scripted events in *End of the Road*, such as an event intended to frighten the player where the antagonist stands at the end of a corridor, and then quickly moves in the direction of the player and screams. Scripting directly through Blueprints was considered, but the animation tracks in the matinee editor made it possible to get a better overview over the timing of animations and sounds, and therefore matinees were used.

For the antagonist, blend spaces and animation Blueprints were not needed, since we defined his movements and player input did not need to be taken into account. Movements were instead scripted through Blueprints and matinees. This method allowed us to control exactly how the antagonist would move.

### 2.3.6 Results and discussion

A possible drawback of blend spaces is that they are limited to two input values. If movement in the vertical dimension is needed in addition the horizontal plane, the use of blend spaces would have been considerably more complex. Animations for vertical movement, such as jumping or falling, were not used in *End of the Road*, however it could have caused problems if these animations were introduced later during development.

The use of blend spaces for the player character was chosen over controlling animations directly through Blueprints, since a large amount of logic would be required and therefore this approach was considered unnecessarily complicated.

A problem that was encountered when testing the animation was that parts of the character's head could be seen in front of the camera when looking around. The cause was identified to be that the character's head did not follow the movement of the camera. To solve this problem, a script was created that reads the current rotation of the Oculus Rift and applies it to the head bone of the character. In this way, it could be guaranteed that the head of the character was never visible for the player.

Bvhacker was found to be useful and simplify some parts of the process, that would otherwise have required more time to do in Blender. A major drawback of the MakeWalk plugin for Blender was that the application of motion capture data to a skeleton produced errors, which required a significant amount of work to correct. The animation interface in Blender was otherwise experienced as being sufficient for editing animations and fixing errors produced in the importing process.

# 3

## Game design

The desire is that *End of the Road* should be appealing to players, to achieve this we decided to have a story, simple game mechanics, and sound effects. Since *End of the Road* makes use of virtual reality, the interaction with the game differs from that of a traditional game, and these differences must be taken into consideration during development. This chapters presents game design techniques, which includes story, designing for virtual reality, game mechanics, frightening elements, level design, meshes and materials, and sound.

### 3.1 Story

For a game to succeed in both catching and maintaining a player's interest, there must be some factors compelling the player to continue playing. In some cases, this achieved by simple mechanics and addictive gameplay, as in the immensely popular *Flappy Bird* (Castillo, 2014). In other cases, a captivating and well-written story that grips the player emotionally can provide the player with enough motivation to continue playing (Lecky-Thompson, 2007, p. 43-45). With such a story, the player will be unable to abandon the game, feeling an urge to continue playing in order to discover what happens.

We decided that *End of the Road* should have a rather simple story, and that it would be strengthened by utilizing the properties of virtual reality. The immersion players feel when using the Oculus Rift causes a sense of actually being in the game world. By placing the character in a situation where there seems to be an imminent threat to the character's life, the player will feel as if he or she is threatened as well.

The final story of *End of the Road* revolves around a woman who enters a haunted mansion in search of help. The mansion can be observed in Figure 3.1. After entering, the door locks behind her and she starts to explore the building in search for a way out. She becomes aware of another person who is somewhere inside, and encounters with this person make it apparent that he is hostile and potentially poses a danger to her life. The player assumes the role of this woman, and must find a key in order to escape.



Figure 3.1: The mansion to explore in *End of the Road*.

## 3.2 Designing for virtual reality

Virtual reality challenges some of the traditional methods for designing a game. To create a truly immersive experience for the player, some modifications in the game design are required. Traditionally, a HUD in front of the game world shows information such as the remaining amount of ammunition or the player's health. Since our virtual reality game aims to make the player feel immersed in a 3D world, projecting a HUD would not be desirable. Instead, this kind of information could be incorporated into the game world. An example of how this problem was solved in *End of the Road*, is that rather than representing the currently equipped item as an icon, the player can look down at the character's hand to see the item they are holding, as shown in Figure 3.2.

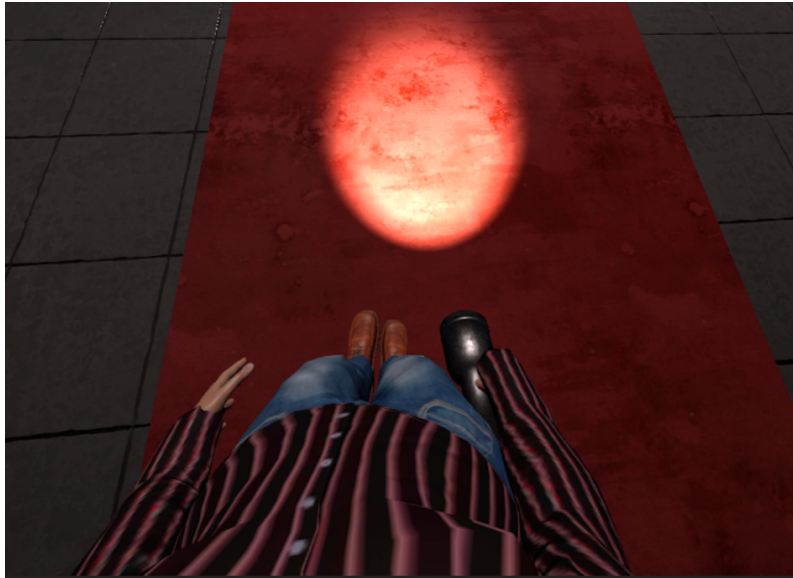


Figure 3.2: Players can see the equipped flashlight in the character's hand.

In addition to game design concerns, virtual reality comes with technical obstacles that must be considered when developing a virtual reality game. According to Vlachos (2015), rendering for a HMD reduces performance. A virtual reality headset requires a field of view (FOV) of about 110 degrees. This is a larger FOV compared to in applications where virtual reality is not used, meaning that more objects in the game world need to be rendered and therefore performance is reduced. Greater FOV results in fewer pixels per degree, which means that more of the world is rendered on the same amount of pixels. This means that a larger amount of texture data is sampled into each pixel. Therefore aliasing becomes a more significant problem and the user might notice a sparkling effect. This effect might reduce the immersion of the user. To reduce the sparkling effect, less detailed materials should be used when designing a virtual reality game.

Another aspect that must be taken into consideration is that games played with virtual reality cannot be as fast paced as traditional games might be, as it would most likely cause nausea to some users as they play (Earnshaw, 2014). Therefore we decided to make the gameplay slow. Despite the challenges of virtual reality, the immersion it enables is a great asset when designing a game. For a horror game such as *End of the Road*, this is especially useful since it might strengthen the player's reaction to the horror elements of the game.

### 3.3 Game mechanics

Immersion is an important aspect of *End of the Road*, and therefore the game mechanics should be intuitive and not distract the player. Additionally, they should resemble actions that are possible in real life. As most of the immersion is experienced by exploring the building, features such as opening doors or navigating are essential to the game. Picking up objects or tools is another important feature, since the goal of the game is to find the key to the entrance door.

By implementing a way to pick up items, we can extend the game mechanics by adding new items, which would be useful if new mechanics were needed. Allowing only one equipped item at a time can be used as a factor contributing to the player's experienced fear, by forcing the player to let go of one item in order to equip another. For example, by including a locked door with a matching key, the player will be required to drop his or her flashlight, losing his or her only source of light. We decided that this feature, together with exploration, would be the gameplay backbone of *End of the Road*.

During development, support for sprinting and jumping existed. However, those features were removed since we wanted the player to explore the environment with caution. Furthermore, these actions would speed up the gameplay, which is undesirable in a virtual reality game, as described in Section 3.2.

## 3.4 Frightening elements

Considering that we are developing a horror game, the game must strive to frighten the player as much as possible. Naturally, in order to accomplish this, we require frightening elements in the game. A person subjected to a scare will react differently depending on their current state of mind (Vachiratamporn et al., 2015). When in a condition of suspense, a frightening event reaches its maximum effectiveness, resulting in a significantly higher level of fear in the player. To utilize this, *End of the Road* attempts to sustain a constant degree of suspense throughout the game by distributing frightening elements appropriately.

There are a number of different methods of instilling fear in the player, they all differ in how the scare is performed, and result in different reactions from the player. Because of this, the decision was made to divide the scares into three different levels, depending on their type and amount of fear caused. This simplifies the process of spreading frightening elements evenly in the world.

The first level is the ambiance in the game. This level consists of the overall feeling, where for example background audio and the environment cause the player to feel uncomfortable and uneasy. To strengthen the sensation that there is an immediate and seemingly supernatural threat, an attempt was made to make the player feel as if the mansion emits an evil and unnerving aura. One step to achieve this is the ambient sound mentioned in Section 3.7. The ambient sound track contains a series of low-pitched sounds which are discomfoting to listen to. This audio track is played with a low volume in the background throughout the entire game, resulting in the player feeling as if the threat is never far off. Another method used is to maintain a consistently dark environment, providing the player with a gloomy and unwelcoming sensation.

However menacing the ambiance appears, the player will not remain frightened unless events occur which strengthen the impression that he or she is in peril. These events could consist of anything between a barrel falling over and a sudden, terrifying jump scare, as described in Section 3.4.1. The second and third levels of scares are comprised of these events. Minor scares such as a glass suddenly shattering pose no immediate threat to the player, but rather startle him or her and raise their nervousity. By doing this, a greater effect can be obtained from the third-level scares, extremely frightening elements such as jump scares. An attempt was made to implement as many such scares as possible, as well as placing them moderately uniformly throughout the mansion. A goal was set to include at least one scary element in each room of the mansion, in order to retain the player's fear. By scattering the frightening aspects in this way, there will be tension in the player, which will strengthen the reaction from the scares (Perron, 2004).

### 3.4.1 Jump scare

A *jump scare* is a technique used to scare the player of a game or the viewer of a movie. The scare is performed by surprising the victim with a sudden event when it is least expected. With the combination of visuals and audio, for example lowering the volume and then suddenly raising it when the jump scare occurs, the reaction from the victim can be strengthened, causing the experience to be more frightening (Muir, 2013). A jump scare is generally created by first hinting at danger, causing the victims to be alarmed, then making them feel safe again. When they feel safe and assume that the danger has passed, the jump scare occurs (Bishop, 2012).

## 3.5 Level design

Between different games, level design approaches can differ. For example, in a platform game the main objective is to traverse the level. In horror games such as *End of the Road*, level design should result in a game world that poses a threat to the player. The level was created with the level editor in Unreal Engine, observable in Figure 3.3. The following sections will explain the level design process, the various problems that were considered and the resulting design in *End of the Road*.



Figure 3.3: The level editor in Unreal Engine

### 3.5.1 Level design process

Designing a level can be done using different methods. Our development team had no previous experience designing levels for a game. Epic Games, creators of Unreal Engine 4, has published guidelines for building levels (Epic Games, 2015f). Their guidelines describe a sequential design process. The reason this approach was chosen over an iterative approach is that a significant amount of work is required to change a level once it is completed. Therefore we found Epic Games guidelines for building levels suitable.

In the level design process described by Epic Games, the first phase is the *prototype pass*, where the basic layout is created using primitive objects such as rectangles. By doing this, developers can observe the layout of the level. At this stage, gameplay mechanics and layout of the level can be tested. It is crucial that the gameplay is tested during this phase, since changes in the basic design require more work once the level is finished. The next phase is the *meshing pass*. In this phase, meshes are added to the level. The rectangles and other simple geometry from the *prototype pass* are replaced with the assets which are to be used. Thereafter, in the *lighting pass*, lights are positioned and tweaked. Additionally, particle effects and post processing can be applied in this pass if desired. Finally, the *polish pass* is the wrap-up phase of the level design process. Here, any final adjustments can be made, such as repositioning and modifying meshes or lighting. These different design phases are recommended and used by Epic Games themselves (Epic Games, 2014), and were utilized during the development of *End of the Road*.

### 3.5.2 Level design in *End of the Road*

The purpose and design of the level must be established before starting to build it. Otherwise the designer will create a level with no intention or idea to support it. When designing the level for *End of the Road* a floor plan was created before the mansion was built in the game engine. Each floor was designed considering potential locations for different jump scares.

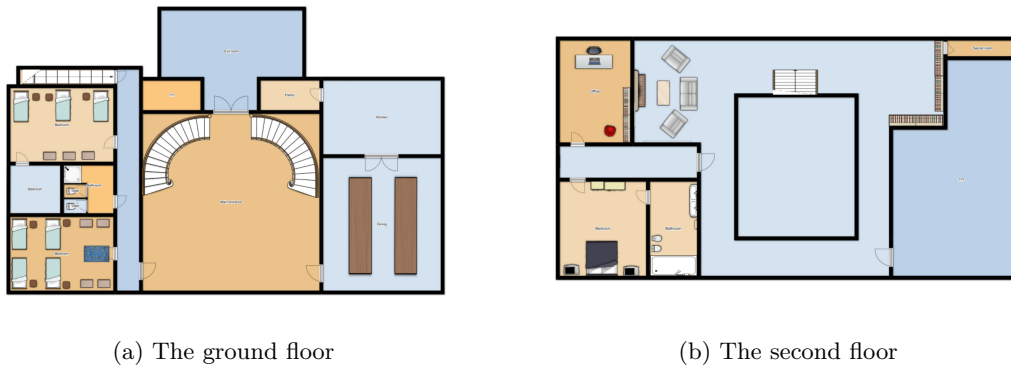


Figure 3.4: The floor-plan of the mansion

As seen in Figure 3.4a, the idea was to have a large hall that would be the first room the player enters. The hall should give the player the impression of feeling small and inferior. We also wanted this area to be the center of the house, and be what connects all other rooms. Another aspect of the floor-plan is the narrow corridors, which are perfect locations for jump-scars or other horrifying events. The mansion has one corridor on each floor, as seen in Figure 3.4.

In *End of the Road*, the player is allowed to explore the game world freely. Therefore, we cannot assume that the player will traverse the environment in a specific pattern. This makes it impossible to control when and in what order scares should occur. Our solution to this problem is to track the amount of scary events the player has seen while exploring, and reveal the key to the entrance door when the player has experienced a fixed number of scares. Another approach would be to use a more linear level design, where the player can only move in a pattern predefined by the level designer. This approach is used by other games such as *Affected*, but was not deemed suitable for *End of the Road* since it would decrease the realism of the game (*Affected, A Virtual Reality Horror Experience*, 2014).

### 3.5.3 Results and discussion

Despite the fact that the development team possessed no knowledge about designing structures prior to the project, the level design in *End of the Road* was considered sufficiently realistic according to the development team. During development, conflicts with the development process arose. The team used Scrum, which differs from the sequential level design process suggested by Epic Games. According to Scrum, changes to the project might occur during the development process, and in these cases a significant amount of work might be required to change the implemented level design. A solution would be to define the game goal and design from the start and not change that during development, which violates the idea of Scrum and iterative processes. Another way might be to change the level design process to fit an iterative workflow. An example would be to design complete individual modules acting like game world building blocks, and place these modules to create the level. The modules can easily be relocated which means that this would fit an iterative process more easily.

Even though the game was designed as a free roaming game, some scares were designed to occur in a specific order. Therefore, some events to guide the players were implemented. However, these appeared to have rather the opposite effect. Players assumed that the events intended to scare

them and therefore chose another path. This caused some scares to appear out of context, while others went unnoticed completely, confusing several players and impairing their game experience.

### 3.6 Meshes and materials

There are several different art styles that could be used when creating meshes and materials. The meshes can for example be made to look like they are part of a cartoon, or be made to look more realistic. The materials used could be very colorful, use a gray scale or simply be black and white. The materials can also reflect the level of realism that the designers are trying to achieve. Since the purpose was to create a realistic game, we chose to use a realistic style when designing meshes and materials. With a more realistic environment, it is easier for the player to feel truly immersed in the game experience.

Since *End of the Road* is a horror game and its story is set in an old mansion, it is important that the custom meshes and materials reflect this theme. Any meshes that are to be used must feel like they belong inside the mansion and together with other meshes. To verify this, images of an antique or old fashioned version of the object to be modeled was often used as reference.

It is important that the meshes look like they belong together in their form and design, but it is equally important to use the right materials. Through the use of materials, an object can for example be made to look new or old, so both the meshes and materials need to conform to the theme.

A few examples of finished meshes with materials applied can be seen in Figure 3.5. The meshes themselves are made to look older in the design, and the materials used are simple wood, metal and glass materials. Additionally, a separate material and texture is used for the wall behind the meshes, and it is visible that this texture looks worn and old.



Figure 3.5: A scene of an office in *End of the Road*.



## 3.7 Sound

An important aspect of video games is the sound (Lecky-Thompson, 2007). As well as contributing greatly to the overall experience of playing a game, a captivating audio track is imperative for a player to feel immersed in the game. Since player immersion is even more important in a game designed for virtual reality, this is a crucial part of the project. Some eerie ambient background sound has an enormous impact on the player's impression. In addition to this, sound effects when performing actions add a whole new level of realism. When the player drops an item, opens a door or encounters a character in the game, the player expects auditory feedback, as would be expected in real life. These effects, even small ones such as footsteps, all work towards providing the player with an as immersive and convincing environment as possible.

To further enhance the involvement in a three dimensional game, one can make sure that the audio is perceived as coming from a certain direction. This is especially important in a virtual reality game, as the objective is to make the player feel as if he or she is actually in the game. In addition, this can be used to manipulate the player, tempting them look at and move towards certain points. For example, by correctly positioning audio, one can make it sound as if something is happening behind the player, hopefully causing them to turn around and examine the source of the sound.

# 4

## Result

*End of the Road* is played with an Oculus Rift virtual reality headset. Thus, the level of immersion exclusive to virtual reality is present throughout the entire experience. Care has been taken to optimize and expedite rendering of the different scenes, as described in Section 2.1.4. In order to succeed in not only frightening, but maintaining a constant, high level of fear in the player, the different techniques outlined in Section 3.4 were implemented.

A game engine, Unreal Engine 4, was used to create the game, simplifying the process of constructing the various game elements via the multitude of included tools described in Section 2.1. The 3D modeling program Blender was used to create a series of 3D models, which were utilized to populate the game world with content. Two different characters were designed: a main character, as well as a malevolent antagonist. Necessary skeletons and animations were produced in order to make possible the various sequences throughout the game. A more detailed description of the tools utilized can be found in Chapter 2.

A short and concise story follows the main character as she is locked inside, and forced to explore, a ghastly mansion. Mechanics such as picking up and employing certain items enrich the overall gameplay experience. Design choices were made such that the resulting game is adapted for virtual reality, resulting in an immersive and enveloping game. The game world consists of a large mansion and the surrounding yard. The game world is discussed in greater detail in Section 3.5. The implemented soundtrack consists of carefully chosen audio with both the intention of contributing to the perceived immersion, as well as strengthening the experienced sensation of fear. To further expand on the eerie audio, scares have been implemented in order to further frighten the player. More details about the sound and the frightening elements of the game can be found in Sections 3.7 and 3.4, respectively.

From the several sessions of user testing, a substantial amount of feedback containing opinions on the game was collected, shown in Appendix A, together with the questionnaire used. One of the questions requested the player to rate how frightening the game was on a scale from one to ten. We received varying responses to this question, yet there is a distinct inclination towards considering the game to be frightening. Thus, we have achieved the designated task of creating a horror game which is perceived as frightening. Additionally, several replies mentioned that the experienced immersion was impressive and significantly more engaging than what they had tried previously and were accustomed to.

# 5

## Discussion

Once the decision had been made to create *End of the Road* using Unreal Engine, and development had started, we were restricted to continuing our task with Unreal Engine. Fortunately, Unreal Engine provided practically every functionality which we required. In the scarce scenarios where no support existed for a certain element which we wished to implement, there was invariably an alternative solution. In the case that a critical component would have been impossible to create in Unreal Engine, it would have forced us to involuntarily change game engine to another alternative. This process is both complicated and time-consuming, ceasing all production for a considerable amount of time before work could be resumed.

In retrospect, additional consideration could undeniably have been given to the choice of game engine. Since this is such a vital decision, it is imperative that a suitable engine is chosen. Ideally, several game engines, perhaps even alternative options to the ones discussed in Section 2.1, would have been personally tested by us, rather than simply basing our decision on the opinion of others. However, due to the limited time frame of the project, such an investigation was not possible. Rather, enough information was gathered to elect a reasonable game engine, acknowledging that complications may emerge and that these would have to be circumvented.

As development progressed, the team's lack of knowledge about architecture became increasingly apparent. Certain design choices during the creation of the in-game mansion resulted in a series of somewhat peculiar features. For example, the corridors appeared to be too narrow and certain rooms were perceived as abnormally large. In order for the mansion to resemble a real-world building, a certain degree of symmetry was required. To achieve this, the windows were positioned in a fixed pattern from the outside. Unfortunately, since we had forgotten to consider the placement of windows in our floor plan, as seen in Figure 3.4, the outside positioning of the windows did not always conform to the inside layout. This means that from the inside, the window placement sometimes feels odd. A slight insight into architecture would have helped us in designing a more realistic game world.

As stated in Section 4, the greater part of our test subjects perceived *End of the Road* as frightening. According to the determined purpose, one of our objectives was to make the game as frightening as possible, while making use of virtual reality. Clearly, the game is frightening, although it is impossible, from our gathered data, to determine whether this is due to the aspect of virtual reality. Ideally, *End of the Road* would be tested on a significantly larger set of individuals, where half of the experiments are performed with and half without the HMD. By collecting and comparing the responses to the questionnaire in this way, it would be feasible to conclude whether or not the game is increasingly frightening or enjoyable when played in virtual reality. Currently we have neither the time nor resources to amass the required amount of people for testing of this character and magnitude.

*End of the Road* is an exploration based horror game, giving the player the option to roam freely in and around the mansion. We realised too late that we were not able to direct the player as effectively as we would have liked. As mentioned in Section 3.5.3, we frequently observed players taking a different path than expected. In hindsight, the game would have benefited if we had designed the level in such a way that the player's interest was captured by the areas and objects which we intended him or her to inspect. For example, additional use of position-based audio, as

described in Section 3.7, could have directed the player's attention to our intention.

## 5.1 Future development

The game development process presented in this thesis is not complete when it comes to accurately measuring fear and immersion. By introducing ways to measure and compare test results in a more accurate manner, the development of a game in general, as well as a virtual reality game, could become even more rapid since inadequate control schemes, 3D models, scenarios and mechanics could be eliminated or reworked earlier on.

# References

- Affected, a virtual reality horror experience* [Video Game]. (2014). Fallen Planet Studios. Retrieved 2015-04-27, from <http://www.fallenplanetstudios.com/affected-2/>
- Archive 3D. (2015). *Archive 3d*. Retrieved from <http://archive3d.net/>
- Baker, P. M. (2014). *Virtual reality* [Web Page]. AccessScience. McGraw-Hill Education. Retrieved 2015-05-12, from <http://www.accessscience.com/content/virtual-reality/757461>
- Bishop, B. (2012). 'Why won't you die?!' *The art of the jump scare*. Retrieved 2015-04-15, from <http://www.theverge.com/2012/10/31/3574592/art-of-the-jump-scare-horror-movies>
- Castillo, M. (2014). *What makes games like Flappy Bird so addictive?* Retrieved 2015-05-01, from <http://theweek.com/articles/450939/what-makes-games-like-flappy-bird-addictive>
- Chopine, A. (2012). *3d art essentials*. Focal Press.
- DeLoura, M. (2009). *The engine survey: General results* [Web Page]. Gamasutra. Retrieved 2015-05-12, from [http://www.gamasutra.com/blogs/MarkDeLoura/20090302/83321/The\\_Engine\\_Survey\\_General\\_results.php](http://www.gamasutra.com/blogs/MarkDeLoura/20090302/83321/The_Engine_Survey_General_results.php)
- Earnshaw, R. (2014). *Virtual reality systems*. Elsevier Science. Retrieved from <https://books.google.se/books?id=gEOjBQAAQBAJ>
- Epic Games. (2014). *Intro to Level Design* [Video]. Epic Games. Retrieved 2015-04-28, from [https://wiki.unrealengine.com/Intro\\_to\\_Level\\_Design](https://wiki.unrealengine.com/Intro_to_Level_Design)
- Epic Games. (2015a). *Animation system overview* [Web Page]. Epic Games. Retrieved 2015-05-12, from <https://docs.unrealengine.com/latest/INT/Engine/Animation/Overview/index.html>
- Epic Games. (2015b). *Apex* [Web Page]. Epic Games. Retrieved 2015-06-02, from <https://docs.unrealengine.com/latest/INT/Engine/Physics/Apex/index.html>
- Epic Games. (2015c). *C++ and blueprint* [Web Page]. Epic Games. Retrieved 2015-05-01, from <https://docs.unrealengine.com/latest/INT/Gameplay/ClassCreation/CodeAndBlueprints/index.html>
- Epic Games. (2015d). *Cpu and gpu sprite particles comparison* [Web Page]. Epic Games. Retrieved 2015-05-18, from [https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/EffectsGallery/1\\_A/index.html](https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/EffectsGallery/1_A/index.html)
- Epic Games. (2015e). *Gpu profiling*. Epic Games. Retrieved from <https://docs.unrealengine.com/latest/INT/Engine/Performance/GPU/>
- Epic Games. (2015f). *Level design content examples* [Web Page]. Epic Games. Retrieved 2015-04-30, from <https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/LevelDesign/index.html>
- Epic Games. (2015g). *Lightmass global illumination* [Web Page]. Epic Games. Retrieved 2015-04-28, from <https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/Lightmass/index.html>
- Epic Games. (2015h). *Light propagation volumes* [Web Page]. Epic Games. Retrieved 2015-05-18, from <https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightPropagationVolumes/index.html>
- Epic Games. (2015i). *Matinee & cinematics* [Web Page]. Epic Games. Retrieved 2015-05-18, from <https://docs.unrealengine.com/latest/INT/Engine/Matinee/index.html>

- Epic Games. (2015j). *Stationary lights* [Web Page]. Epic Games. Retrieved 2015-05-17, from <https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightMobility/StationaryLights/index.html>
- Gera, E. (2014). *The science of fear: How to measure scares in gaming* [Newspaper Article]. Retrieved from <http://www.polygon.com/2014/10/29/7081075/horror-games-testing-until-dawn-amnesia>
- Guru Games. (n.d.). *Medusa's labyrinth*. Retrieved from [https://www.kickstarter.com/projects/425879580/medusas-labyrinth?ref=nav\\_search](https://www.kickstarter.com/projects/425879580/medusas-labyrinth?ref=nav_search)
- Hahne, B. (2015). *The motionbuilder-friendly bvh conversion release of cmu's motion capture database* [Web Page]. Author. Retrieved 2015-06-01, from <https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/cmu-bvh-conversion>
- indiedb. (2015). *100 most popular engines today* [Web Page]. Author. Retrieved 2015-05-13, from <http://www.indiedb.com/engines/top>
- Jensen, H. W. (1996). *Global illumination using photon maps* [Report]. The Technical University of Denmark. Retrieved 2015-05-18, from [http://graphics.ucsd.edu/~henrik/papers/photon\\_map/global\\_illumination\\_using\\_photon\\_maps\\_egwr96.pdf](http://graphics.ucsd.edu/~henrik/papers/photon_map/global_illumination_using_photon_maps_egwr96.pdf)
- Lecky-Thompson, G. W. (2007). *Video game design revealed*. Boston, Massachusetts: Course Technology.
- Luebke, D. P. (2003). *Level of detail for 3d graphics*. Morgan Kaufmann.
- MakeHuman team. (2015a). *Makehuman and its purpose* [Web Page]. MakeHuman team. Retrieved 2015-05-10, from [http://www.makehuman.org/doc/node/makehuman\\_and\\_its\\_purpose.html](http://www.makehuman.org/doc/node/makehuman_and_its_purpose.html)
- MakeHuman team. (2015b). *Makewalk basic workflow* [Web Page]. MakeHuman team. Retrieved 2015-05-10, from [http://www.makehuman.org/doc/node/mhblendertools\\_makewalk\\_basic\\_workflow.html](http://www.makehuman.org/doc/node/mhblendertools_makewalk_basic_workflow.html)
- MakeHuman team. (2015c). *Mhblendertools: Makewalk basic workflow* [Web Page]. MakeHuman team. Retrieved 2015-05-18, from [http://www.makehuman.org/doc/node/mhblendertools\\_makewalk\\_basic\\_workflow.html](http://www.makehuman.org/doc/node/mhblendertools_makewalk_basic_workflow.html)
- MakeHuman team. (2015d). *Mhblendertools: Mhx other rigging systems* [Web Page]. MakeHuman team. Retrieved 2015-05-18, from [http://www.makehuman.org/doc/node/mhblendertools\\_mhx\\_other\\_rigging\\_systems.html](http://www.makehuman.org/doc/node/mhblendertools_mhx_other_rigging_systems.html)
- Maraffi, C. (2003). *Maya character creation: Modeling and animation controls*. Pearson Education. Retrieved from <https://books.google.se/books?id=1GE6YbntJrQC>
- MarketsandMarkets. (2014). *Augmented reality & virtual reality market by technology types, sensors (accelerometer, gyroscope, haptics), components (camera, controller, gloves, hmd), applications (automotive, education, medical, gaming, military) & by geography - global forecast and analysis to 2013 - 2018* [Report]. Retrieved 2015-05-18, from <http://www.marketsandmarkets.com/Market-Reports/augmented-reality-virtual-reality-market-1185.html>
- Masters, M. (2015). *3ds max, maya lt or blender - which 3d software should i choose for asset creation?* [Web page]. Digital-Tutors. Retrieved 2015-05-01, from <http://blog.digitaltutors.com/3ds-max-maya-lt-blender-3d-software-choose-asset-creation/>
- Muir, J. K. (2013). *Horror Films FAQ: All That's Left to Know About Slashers, Vampires, Zombies, Aliens, and More*. Applause Theatre & Cinema Book Publishers.
- Nintendo. (2015). *Virtual boy* [Web Page]. Retrieved 2015-05-10, from [http://nintendo.wikia.com/wiki/Virtual\\_Boy](http://nintendo.wikia.com/wiki/Virtual_Boy)
- Oculus VR. (2015). *Virtual reality* [Web Page]. Oculus VR. Retrieved 2015-05-14, from [http://static.oculus.com/sdk-downloads/documents/Oculus\\_Best\\_Practices\\_Guide.pdf](http://static.oculus.com/sdk-downloads/documents/Oculus_Best_Practices_Guide.pdf)
- Ohio State University. (2015). *Open motion project* [Web Page]. Ohio State University. Retrieved 2015-06-01, from [http://accad.osu.edu/research/mocap/mocap\\_data.htm](http://accad.osu.edu/research/mocap/mocap_data.htm)
- Payatagool, C. (2008). Theory and research in hci: Morton heilig, pioneer in virtual reality research. Retrieved 2015-05-29, from [http://www.telepresenceoptions.com/2008/09/theory\\_and\\_research\\_in\\_hci\\_mor/](http://www.telepresenceoptions.com/2008/09/theory_and_research_in_hci_mor/)
- Perron, B. (2004). *Sign of a Threat: The Effects of Warning Systems in Survival Horror Games*. COSIGN 2004 Proceedings. Retrieved from [http://www.cosignconference.org/downloads/papers/perron\\_cosign\\_2004.pdf](http://www.cosignconference.org/downloads/papers/perron_cosign_2004.pdf)

- Perron, B. (2009). *Horror video games: Essays on the fusion of fear and play* [E-book]. McFarland & Company Inc.
- Pharr, M., & Humphreys, G. (2010). *Physically based rendering* [Book]. Elsevier Inc. Retrieved from [https://books.google.se/books?hl=sv&lr&id=9nJBAJhT8C&oi=fnd&pg=PP2&dq=physically+based+rendering&ots=d\\_SSNgVqxv&sig=T5FliCpGAfFIq6mxWYJ0vIS5dJI&redir\\_esc=y#v=onepage&q=physically%20based%20rendering&f=false](https://books.google.se/books?hl=sv&lr&id=9nJBAJhT8C&oi=fnd&pg=PP2&dq=physically+based+rendering&ots=d_SSNgVqxv&sig=T5FliCpGAfFIq6mxWYJ0vIS5dJI&redir_esc=y#v=onepage&q=physically%20based%20rendering&f=false)
- Rob van der Meulen, J. R. (2013). *Forecast: Video game ecosystem, worldwide, 4q13* [Report]. Gartner Inc. Retrieved 2015-05-11, from <http://www.gartner.com/resId=2606315>
- Scrum.org, & Scruminc. (2014). *The scrum guide* (Vol. 2015) (Web Page No. 8/2). ScrumGuides.org. Retrieved from <http://www.scrumguides.org/scrum-guide.html>
- Shene, D. C.-K. (2010). *Mesh basics* [Powerpoint]. Michigan Technological University.
- Ström, D. (2015). *Guru games*.
- Sutherland, I. E. (1968). *A head-mounted three dimensional display*. The University of Utah.
- The Free 3D Models. (2015). *Tf3dm*. Retrieved from <http://tf3dm.com/>
- Vachiratamporn, V., Legaspi, R., Moriyama, K., Fukui, K.-i., & Numao, M. (2015). An analysis of player affect transitions in survival horror games. *Journal on Multimodal User Interfaces*, 9(1), 43-54. Retrieved from <http://dx.doi.org/10.1007/s12193-014-0153-4> doi: 10.1007/s12193-014-0153-4
- Vlachos, A. (2015). *Advanced vr rendering* [Presentation]. Valve. Retrieved 2015-05-18, from [http://media.steampowered.com/apps/valve/2015/Alex\\_Vlachos\\_Advanced\\_VR\\_Rendering\\_GDC2015.pdf](http://media.steampowered.com/apps/valve/2015/Alex_Vlachos_Advanced_VR_Rendering_GDC2015.pdf)
- Wooldridge, D. (2015). *bvhacker* [Web Page]. Author. Retrieved 2015-05-11, from <http://davedub.co.uk/bvhacker/>

# Appendices



# A

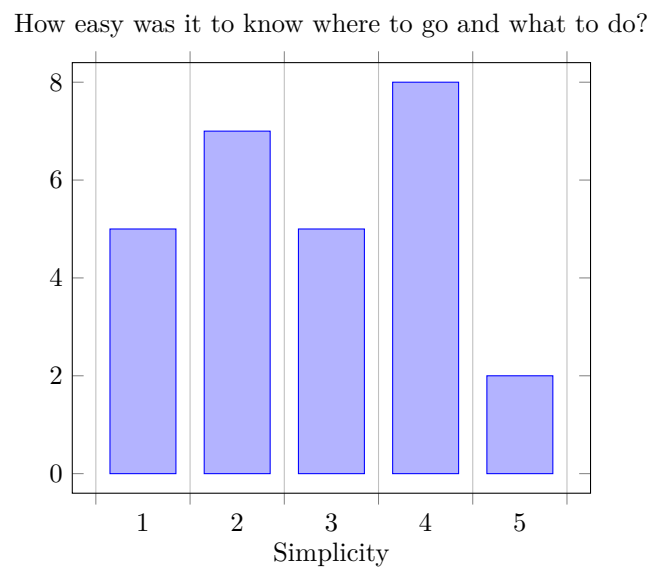
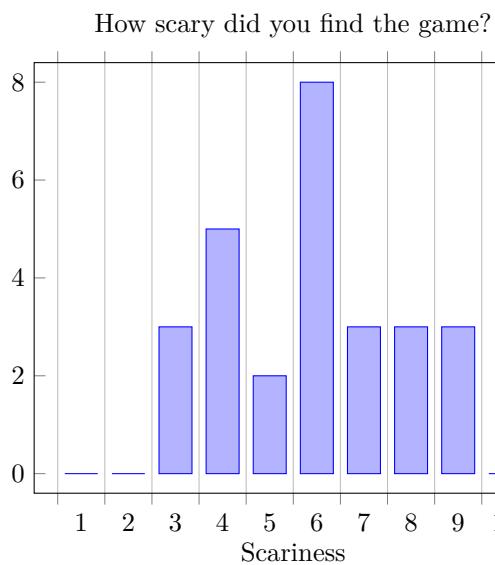
## User testing

In order to gather suitable feedback from test subjects, we formulated a questionnaire which they were required to answer. This questionnaire is presented below, along with the accumulated responses.

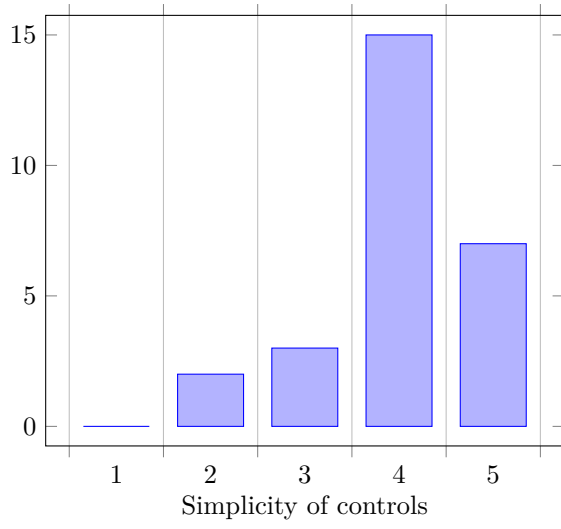
### A.1 Questionnaire

1. What was your general impression of the game?
2. How scary did you find the game? Comments?
3. How intuitive were the controls? Comments?
4. How easy was it to know where to go and what to do? Comments?
5. What did you think of the lighting in the game? Comments?
6. Did you experience any nausea? If so, how much and when/why?
7. Do you have any suggestions for improvement?
8. Other comments?

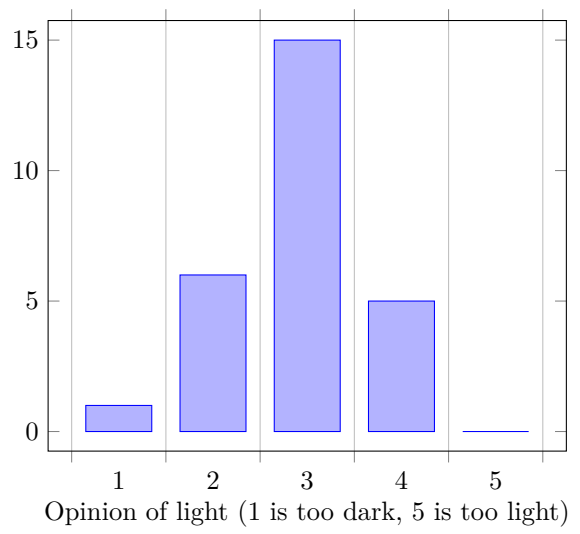
### A.2 Quantitative results



How intuitive were the controls?



What did you think of the lighting in the game?



# B

## Interview with Guru Games

As a part of the pre-study described in section 1.5.2, an interview was held with *Daniel Ström*, *Petter Henriksson* and *Johannes Jonsson* from *Guru Games*. This appendix contains the asked questions and the given responses. At the time of the interview, the studio was working on two games: *Magnetic: Cage Closed* and *Medusa's labyrinth*.

1. Vad använder ni för versionshantering?

*Separera i olika levels som ligger som lager, det vill säga flera levels som ligger på samma plats. Till exempel props i ett eget lager så att de kan committas separat.*

2. Hur ser ert workflow ut, vad görs först (level design, features, etcetera)?

*Förproduktionsfas där mekanik, design och art style prototypas.*

*Jobba modulärt.*

*Testa varje vecka. Hela teamet går igenom, får feedback.*

*Svårare att prototypa skräckspel utan att ha ljudeffekter och grafik.*

*Prioritera olika uppgifter/features och jobba sig nedåt.*

*Körde trello för Magnetic. En nackdel med det är att vissa saker såsom ljusbakning aldrig kan betraktas som färdiga.*

3. Hur många jobbar på en level, hur sköter versionshanteringen det?

*I teorin hur många som helst, om det är vettigt strukturerat.*

4. Blueprint och C++, vad använder ni?

*Medusa är mestadels skrivet i blueprint.*

*Stora/avancerade komponenter görs inte i blueprint.*

*Prototypning och enklare events fungerar bra i blueprint.*

*Debug av blueprint är svårt.*

*Blueprints är oftast snabbare att göra.*

5. Vilka problem har ni stött på?

*Det är inte ekonomiskt försvarbart att ha en heltidsanställd ljudläggare.*

6. Hur optimerar och mäter ni prestanda i Unreal Engine?

*Hålla koll på tris i modeller, storleksordningen 1-2 miljoner tris på skärmen samtidigt är rimligt.*

*Få saker som skapar skuggor.*

*Nästan allt ljusbakat i Magnetic, endast det som rör sig är dynamiskt.*

*LOD-nivåer.*

*View distance, hur långt bort skuggor renderas till exempel.*

*Unreals profiler.*

*Speglar är svårt och riktigt tungdrivet.*

*Huvudleveln/persistent level borde vara tom, ladda endast in saker som behövs.*

*Kolla på streamvolymen.*

*BSP används endast för prototypning.*

7. Hur jobbar ni med att minimera illamående som vissa personer upplever?

*Framerate.*

*Aldrig låsa kameran.*

*Inga cutscenes.*

*Svårt om man hoppar och rör sig snabbt.*

*Fokuspunkt, något att fästa blicken på, jämförbart med att titta på vägen när man åker bil.*

8. Vad fokuserar ni på för att maximera inlevelse i miljön?

*Minimera UI-element. Ingen ammo counter, titta på kogret istället. Diskret sikte. Inventory-system.*

*Mörkerseende.*

*3D-positionerat ljud. Ljudet ändras mellan olika miljöer, tonas in när man närmar sig miljön. ex. mer eko i katakomberna.*

*Oerfarna spelare rör inte på huvudet. Få spelare att vilja röra på huvudet av olika anledningar så att de lär sig.*

*Använd head tracking (böja sig runt hörn till exempel).*

9. Hur definierar ni ett läskigt spel?

*Få miljön att kännas som en del av hotet.*

*Saker som händer/ändras när spelaren inte tittar.*

*Titta mycket på andra skräckspel, mycket inspiration från Amnesia, läs på Gamasutra.*

10. Hur och hur mycket testar ni spelet på utomstående personer under utvecklingen?

*Svårt att veta när man jobbar på det om det är läskigt.*

*Medusa - dra in folk, locka med kakor.*

*Ute på events.*

*Samla in feedback, frågor till spelare, titta på när spelare spelar, låt spelaren testa själv.*

*Pulsmätning är inte ett effektivt sätt att mäta rädsla på. I ett test var folk mer nervösa innan, på grund av själva testprocessen. Bättre med frågeformulär.*