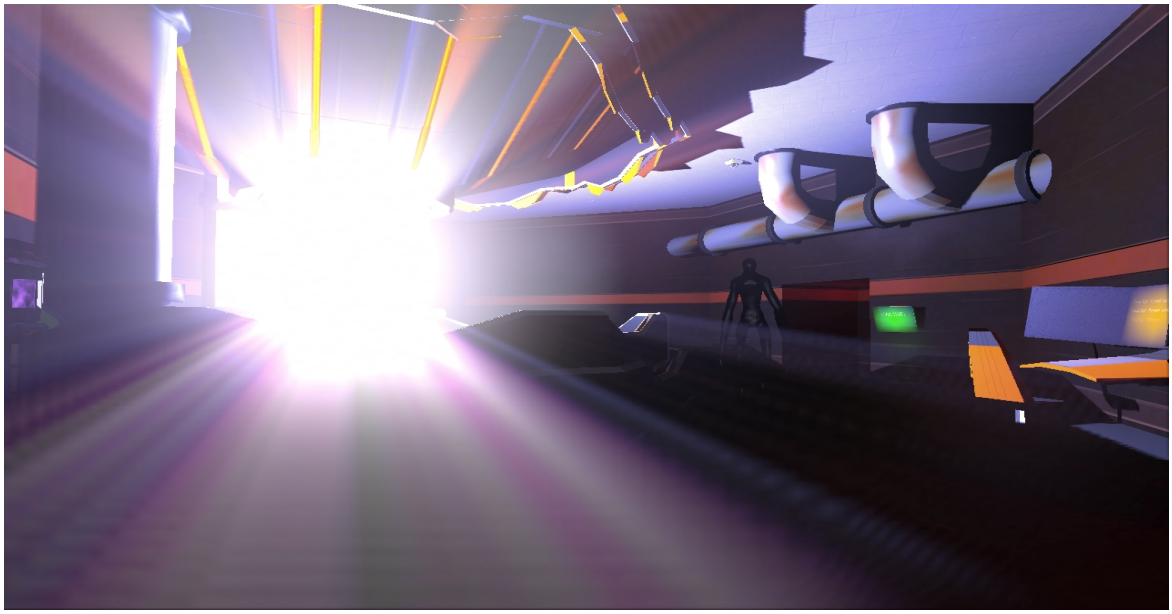
CHALMERS



Breach

A 3D horror game designed for immersion and head-mounted displays

Bachelor of Science Thesis in Computer Science & Software Engineering

BRISTAV, KARL
KAUFMANN, ROBERT
LUNDKVIST, JESPER
ODEH, SAM
SIKANDER, ERIK
TOUCHE, ROBIN

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

Breach
A 3D horror game designed for immersion and head-mounted displays

Karl H. Bristav, C. Robert Kaufmann, Jesper M. Lundkvist,
Sam R. Odeh, Erik Å. Sikander, Robin R. Touche

Examiner: Sven-Arne Andreasson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: A screenshot from the game *Breach*.

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

Abstract

For this thesis, a horror video game has been developed from concept to finished product, with the goal of being as immersive as possible by finding and highlighting techniques that provide a high level of immersion and enjoyment in a horror video game. The presented techniques address what immersion, game design, and asset creation aspects lead to an enjoyable, immersive, and coherent game. Furthermore, an evaluation of the applicability of a head-mounted display was performed to discern if such hardware enhances the game experience and the perceived level of immersion.

By performing a literature study on immersion and game design theory, techniques were found and applied to the game, which was developed using an agile methodology. In order to speed up the development process and focus on gameplay and asset creation, the game development tool *Unreal Development Kit* was used.

In addition to the video game *Breach*, the attained results are the game design and immersion techniques utilized in the game which were found enhancing the game experience, for instance *artificial intelligence*, *sound design*, *design for head-mounted displays*, and *objective design*.

The implication of this thesis is that horror games, such as *Breach*, benefit from attaining an increased level of *emotional*, *cognitive*, *spatial*, and *sensory immersion*. Contributing factors to attaining both enjoyment and immersion of these types are implementing an artificial intelligence with its own goals, implementing support for a head-mounted display, designing objectives with care, and utilizing a diegetic and partly non-linear sound design.

Keywords: Video Game, Game Design, Horror, Immersion, Unreal Development Kit, UDK, Virtual Reality, VR, Head-Mounted Display, HMD, Breach

Sammanfattning

För denna avhandling har ett skräckspel utvecklats från koncept till färdig produkt, med målet att vara så inlevelserikt som möjligt genom att hitta och lyfta fram tekniker som ger en hög nivå av inlevelse och nöje i ett skräckspel. De presenterade teknikerna behandlar vilka inlevelse-, speldesign- och innehållsskapandeaspekter som leder till en trevlig, inlevelserik, och konsekvent spelupplevelse. Dessutom har en utvärdering av tillämpningen av en head-mounted display utförts för att avgöra om sådan hårdvara förhöjer spelupplevelsen och den upplevda graden av inlevelse.

Genom att utföra en litteraturstudie av inlevelse- och speldesignteori, har tekniker identifierats och tillämpats på spelet genom agila metoder. För att förkorta utvecklingsprocessen och fokusera på spelupplägen och bandesign användes spelutvecklingverktyget *Unreal Development Kit*.

Förutom själva spelet *Breach* uppnåddes andra resultat inom bland annat speldesign- och inlevelsetekniker, vilka användes för att öka spelupplevelsen i *Breach*. Detta inkluderar t.ex. *artificiell intelligens*, *ljuddesign*, *design för head-mounted displays* och *utformning av spelarmål*.

Innebörden av denna avhandling är att skräckspel som *Breach* drar nytta av en ökning i *emotionell*, *kognitiv*, *rumslig* samt *sensorisk inlevelse*. Bidragande faktorer till att uppnå både nöje och inlevelse av dessa typer är implementerandet av en artificiell intelligens med sina egna mål, implementerandet av stöd för en head-mounted display, utformningen av mål med omsorg och utnyttjandet av en diegetisk och delvis icke-linjär ljuddesign.

Acknowledgements

We would like to thank our supervisor Ulf Assarsson for support and advice during the project, who especially gave quality feedback rapidly during the writing of this report. We also want to thank the Oculus VR team for their Kickstarter campaign, which inspired us to pursue this project. Finally, we would like to express our gratitude to our friends and family for the feedback they have given us while reading this thesis.

Contents

Chapter 1

Introduction

The video game industry exploded into life with the video game market forming approximately in the early 1970s (Perry et al., 1982), and it is now comparable, with regards to revenue, with other high-grossing forms of entertainment, such as the music and film industries. Video games have achieved saturation in more target audiences than ever before, and they can currently be found in many consumer devices ranging from personal computers to telephones, as well as in popular web services such as Facebook. Not only do video games grow in popularity, but the quality of games' assets and the complexity of games' engineering technology is following the same trend. This has the side effect that the cost and time required for a game project are growing as well (Gamasutra, 2010), thus turning developers to embrace already existing engines and tools instead of creating them themselves.

One such tool is Unreal Development Kit (UDK), which is a game development toolkit based on the game engine Unreal Engine 3 (UE3), targeted at independent game developers (Epic Games, 2013a). It is free of charge for non-commercial purposes, with special terms for commercial use (Epic Games, 2013b). It features integrated functionality frequently required to produce games, *e.g.* a level editor, scripting language, and material creation tools (Epic Games, 2013c). The functionality of UDK has the potential to greatly speed up the game development process, giving independent game developers a means to keep up with the increasing quality demands in the game industry.

UDK also has integrated support for the Oculus Rift, a Virtual Reality (VR) hardware in the form of a Head-Mounted Display (HMD). While VR technology traditionally has been used for research purposes, with few commercialization efforts, the Oculus Rift is an affordable, consumer grade HMD. The goal of using VR technology is to enhance the immersion in video games, to make the player feel absorbed by the experience of playing them. Immersion is in itself an emerging research subject which has started to be explored with multiple models to categorize the constituting parts of immersion.

1.1 Aim of report

This bachelor's project aims to create a horror video game that is as immersive as possible, developed from concept to finished product, and to find elements which

increase the perceived immersion. The purpose of this thesis is to highlight the techniques found best suited for providing a high level of immersion in a horror video game.

This is to be accomplished using a game development toolkit and virtual reality hardware (a head-mounted display), by six developers over a time period of four months (a combined 2 400 hours of labor), creating a new horror game named *Breach*.

1.2 Limitations

The project is limited to being based on existing literature on immersion, and does not present any further research on the subject. Measured immersion and quality is, unless otherwise stated, based on the project members' subjective experience, with sensory immersion only provided by audiovisual feedback using conventional audio hardware and a head-mounted display.

The developed game is limited to be runnable on the Windows 7 operating system by Microsoft, since each of the computers used for the development run operating systems compatible with the platform. Performance is considered sufficient if the game runs at a consistent 60 frames per second on the machine used for testing. Furthermore, the playable game is considered long enough when it consists of at least 15 minutes of play time for a first time user with previous experience in playing video games.

1.3 Problem

Even when utilizing a game development toolkit, the project process from concept to finished product is faced with challenges. This section states the problem associated with this thesis, such as content creation, artificial intelligence, and game design, as well as the challenge of attaining a satisfying level of immersion in the finished product.

Immersion

Game immersion can be defined in a number of ways (Brown and Cairns, 2004), and to avoid ambiguity there needs to be a clear definition made as to what immersion is in the context of this thesis.

Game Design

There are multiple game design patterns and variations which can be implemented in a video game, and with this comes the problem of finding and evaluating which combination of design choices caters to immersion and enjoyment in *Breach*.

The Head-Mounted Display (HMD) is a type of VR hardware unit which may enhance the immersion in a virtual environment. Since *Breach* consists of a virtual environment, this thesis evaluates if an HMD can be utilized, in addition to other design choices, to attain a higher level of immersion.

Artificial Intelligence

In *Breach*, an Artificial Intelligence (AI) is to be used to control a monster. This monster is the most prominent horror element within the game and therefore needs to be believable in the way that it moves, as well as challenging to the player to increase immersion. With this in mind, the problem is thus how the AI should be constructed to achieve these goals.

Content

With creating a game comes the task of creating the content and art assets, which constitute the representation of the game world. Assets can either be created by the development team or acquired from external sources. However, a consistent look and feel is important in attaining a believable environment, and to achieve this, each asset's design has to be carefully considered such that it adds to the enjoyment of playing *Breach* and is fitting to the consistently designed experience of the game. The problem regarding content is thus both choosing what external assets may be utilized in *Breach*, as well as how assets made by the development team should be created to fulfill the requirements of coherency and enjoyment.

There are multiple types of content (each having their own work flow) which need to be considered. In this thesis, the following types of assets will be discussed:

- Levels
- Models
- Textures
- Sounds
- Gameplay scripts

The problem with models, textures, and sounds is whether these assets should be acquired from external sources or created by the development team and how these should be implemented to provide a coherent look and feel in *Breach*.

Creating the levels of the game presents the problem of choosing what areas to include in *Breach*, as well as making them coherent to prevent single areas from standing out in the game world.

When creating gameplay scripts, the problem consists of what gameplay-related functionality to implement in order to create the feeling of a dynamic game world which reacts properly to the player's actions, creating a sense of enjoyment for the player.

1.4 Methodology

In this section, the organization of the project is presented together with the work flow and software tools used to create *Breach*. Furthermore, the underlying method of research is explained.

Project organization

The development process used in the project was inspired by agile and iterative methods, such as Scrum and Extreme Programming, leading to the development process being similar to that of the Rational Unified Process (Rational Software, 2001). The similarity is based on the following parts:

Iterative development: Utilized since not all requirements were known (and could not realistically be found) in the project concept stage.

Compartmentalization: Development had to be performed on distinct and independent parts.

Visual development: Where applicable, charts were drawn to illustrate and make design decisions.

Change control: Due to multiple, independent developers, synchronization and verification of changes were performed continuously.

Development was split into smaller work tasks, such as creating a certain model or level part, merge-able into the final product upon completion. These tasks were formally assigned to project members and evaluated on a weekly basis, by which progress could be quantified and new tasks assigned at a rapid pace. To provide security by means of revision history and to enable concurrent work, the version control system Perforce (Perforce Software, 1995) was utilized.

While mandatory and formal meetings took place, in which weekly goals were set and project status reported, most of the communication was done through informal meetings and discussions. Due to the team size this allowed to make rapid changes to the game while still keeping propagation of information easy.

Conclusively, our choice of development process provided resilience of changing working conditions and changes in the the direction of the project.

Immersion study

Due to the scope of this thesis there is no need to perform new research on what the constituting parts of immersion are. However, this thesis does, based on a literature study of various journals and books, utilize existing research on immersion to define the immersion concept used in the development of *Breach*.

Measuring immersion

Measuring the exact amount of immersion that a player feels when playing a game is a near impossible task. This is due to factors such as players having differing preferences of what makes them immersed, thus resulting in that not all players experience an equal amount of immersion when playing the same game.

This thesis does, however, aim to create an immersive game and therefore needs a way of measuring its immersion. In this thesis immersion is (as stated in the section Limitations) measured through the subjective opinion of the development team when playing *Breach*.

Unreal Development Kit

An important contribution to reducing the complexity of the project was to use Unreal Development Kit (UDK), a game development tool, which provides a game engine, thus eliminating the need to write a custom engine. This allowed the development team to focus on content creation for the project, such as materials, levels, and game logic scripting in UnrealScript (the scripting language used in UDK).

UDK comes with default assets which could be used as placeholders early in the development process. This had the advantage that prototypes and basic versions of areas were runnable early in the development, by which testing of gameplay could be performed, ultimately leading to a more efficient use of development time.

Chapter 2

Immersion

Video game developers strive to create immersive games since players want to be immersed when they play video games (Huiberts, 2010). There are, however, multiple definitions of immersion (Brown and Cairns, 2004) and creating an immersive video game is not a trivial task. A single mistake from the developer of the video game could make the players lose their immersion (Björk and Holopainen, 2004). That is why, during the development of a game, developers need to consider game patterns which help achieve immersion, in order to not make easily avoided mistakes.

In this thesis, immersion is defined as the state of feeling connected with, or within, a virtual world. Immersion is, in essence, about being absorbed by the virtual environment, becoming less aware of the real world and the self, which creates a focus on the environment. This is a desired effect in video games, a medium where players generally want to experience immersion. The purpose of this chapter is to describe the different types of immersion used in this thesis.

2.1 Combining Immersion Models

There are multiple models describing different types of immersion and in this thesis, a combination of two models was created and is hereby presented.

The ECSS model

Set in a gameplay context, Björk and Holopainen (2004) state that immersion can be divided into four categories: Emotional immersion, Cognitive immersion, Spatial immersion, and Sensory-motoric immersion. This model will be referred to as the ECSS model throughout this thesis.

Emotional immersion is reached by having events within a game which unfold parts of a story or reveal new information (Björk and Holopainen, 2004), providing the player with a deeper knowledge about the game world. This is the same type of immersion that certain other forms of media provide, such as books and movies.

Cognitive immersion can be described as giving the player a sense of problem solving within the game environment (Björk and Holopainen, 2004). To

achieve this, the player is presented with a set of possible actions to perform, forcing a choice to be made by the player.

Spatial immersion is achieved by letting the player control an actor and its movement within the game environment, effectively making the player regret actions that do not work out as intended (Björk and Holopainen, 2004).

Sensory-motoric immersion can be described as actions that are repeated by the player and then processed to provide feedback to the player in an appropriate form, such as visual or aural (Björk and Holopainen, 2004). Games that strongly focus on such immersion are, for instance, fighting games such as Street Fighter (Capcom, 1987) and music games such as Guitar Hero (Harmonix Music Systems, 2005).

The SCI model

In addition to gameplay immersion, by using the human senses, *e.g.* aural or visual senses, immersion can be attained for the player. According to Ermi and Mäyrä (2005), the audiovisual quality is an important factor when achieving immersion based on sensory input, and it is an integral part in their model for describing immersion in a video game. This model, the SCI model, classifies three key parts for immersive gameplay: Sensory, Challenge-based, and Imaginative immersion. Sensory immersion is attained when the senses are stimulated more by the virtual environment than by the real world, *e.g.* by utilizing surround sound and tactile feedback. Challenge-based immersion is where the game challenges the player's motoric or mental skills, *e.g.* puzzle solving. Imaginative immersion is where the virtual world has story elements that make the player use her imagination and relate to agents in the game world.

The combination

According to Ermi and Mäyrä (2005), some of the existing game immersion models are overlapping. This thesis has combined two models: the ECSS model from Björk and Holopainen (2004) and the SCI model from Ermi and Mäyrä (2005), which covers the visual and sound aspects of a video game. These two models overlap on two occasions. The ECSS model's cognitive immersion is an overlap of the challenge-based immersion of the SCI model, and the imaginative immersion of the SCI model is an overlap of both emotional immersion and spatial immersion of the ECSS model. The ECSS model also includes sensory-motoric immersion. However, sensory-motoric immersion is not further explored in this thesis since it is not implemented in *Breach*, the game that this thesis is built upon. Thus, the immersion aspects covered in this chapter are emotional immersion, cognitive immersion, and spatial immersion, which are based on the ECSS model, as well as sensory immersion from the SCI model.

2.2 Emotional Immersion

Research, such as that performed by Jennett et al. (2008) and Brown and Cairns (2004), indicates that emotional involvement is a key factor in immersion, and it is thereby an important part of the ECSS model as well as the SCI model.

Most of the different types of patterns that constitute emotional immersion can be divided into three categories: storytelling, ownership, and emotion. Although these categories do not cover all patterns facilitating the implementation of emotional immersion, they cover the basic patterns which are discussed and relevant to this thesis.

Figure 2.1: *This image shows social interaction between the player and a Non-Player Character (NPC) in the game Dishonored (Arkane Studios, 2012). Note that the NPC response is in auditory form, not only text.*

Storytelling

Storytelling is mentioned in the introduction where it is described as the unfolding of a story. Although the unfolding is the big part of this category, there are subpatterns that further increase immersion, such as social dilemmas and betrayal (Björk and Holopainen, 2004). Such patterns often require some sort of social interaction, which in turn further enhance immersion. Figure 2.1 shows one such interaction.

Ownership

The ownership category covers patterns that involve the player having ownership of something. Example of such patterns are rewards and penalties (Björk and Holopainen, 2004). According to Salen and Zimmerman (2003), penalties are used as a punishment for failure while rewards are used as an incentive for players to achieve a game goal. These patterns involve the player receiving or losing an item, control of an area, or something more abstract such as health, and will for a short period of time give the player a feeling of emotional immersion (Björk and Holopainen, 2004). The consequence of the gain or loss needs to be of importance to the player in order for her to fully experience this category of immersion.

The ownership category also covers player created content, such as constructing a world or a character, where ownership as a goal can create an incentive to keep playing a game (Björk and Holopainen, 2004) or reward the player for the hours already spent (Salen and Zimmerman, 2003), even more so if the game world is persistent.

Emotion

The easiest method of achieving emotional immersion is to use various patterns involving emotions (Björk and Holopainen, 2004). These patterns include, but are not limited to, anticipation, tension, and surprises. Anticipation, as the name suggests, is when the player is given something to anticipate, which can be in the context of a story or a separate part of the game, such as a level system where the next level will allow the player to further advance their character. Tension can be created when the player is faced with choices or scenarios where the outcome is uncertain (Adams and Rollings, 2006) but has an impact on the player's avatar, story, or in the real world (seen in Figure 2.2)(Björk and Holopainen, 2004). Surprises are events within the game which the player did not or could not foresee (Björk and Holopainen, 2004); these events can be of either positive or negative nature. The player can predict that such an event might occur, such as having the knowledge of the possibility of being flanked in a first person shooter, or it can be a complete surprise to the player. A reason to use surprises is that it could add replayability to the game (Salen and Zimmerman, 2003).

Figure 2.2: *This figure illustrates a decision in Mass Effect (Bioware, 2007), where the repercussions of the decision are uncertain, increasing the player's tension. (My Learned Friends, 2012)*

2.3 Cognitive Immersion

Cognitive immersion is about the player's logical thinking where freedom of choice is a requirement. Providing the player with the freedom of choosing between different actions forces her to consider what the appropriate action is, immersing her in the game world. Depending on the decision made, the player could expect future outcomes to be affected by it (Adams and Rollings, 2006).

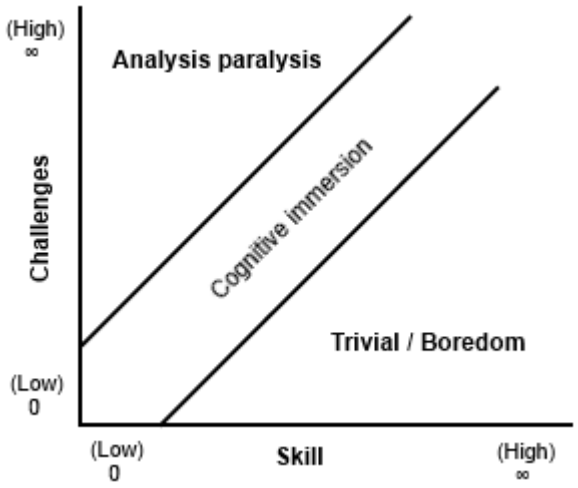


Figure 2.3: *This figure illustrates the issue of setting the correct difficulty for the player. Failing to do so will result in ether an analysis paralysis (shown above the band) or boredom (shown below the band).*

This type of immersion is difficult to maintain, and it can easily be disrupted, and therefore relies on the game providing an appropriate difficulty, illustrated in Figure 2.3. Making decisions too hard leads to the player finding herself in an analysis paralysis (Björk and Holopainen, 2004) or downtime, which means that the player over-analyzes the situation or that the player is not actively working towards the goal, respectively. Making decisions too easy will lead to the required reasoning being trivial, causing the player to be bored due to the lack of challenge (Sweetser and Wyeth, 2005), thus losing immersion (Björk and Holopainen, 2004). The difficulty of the decisions may need to be adapted during the course of the game to provide an appropriate level of challenge to the player. Successfully doing so will grant the player a feeling of satisfaction (Lazzaro, 2004).

Cognitive immersion also benefits from gameplay patterns such as *tradeoff* and *risk/reward* (Björk and Holopainen, 2004). *Tradeoff* means that there is not always a correct choice, due to each choice having both advantages and disadvantages. This allows the player to make the choice that she thinks is the most beneficial to her. With *risk/reward*, the outcome of a choice have the potential of being either advantageous or disadvantageous to the player, *i.e.* a gambling scenario.

Emotional immersion generally does not work well together with cognitive immersion according to Björk and Holopainen (2004). Game events or actions that increase one type of immersion often decrease the other. For example,

while surprises can increase emotional immersion, cognitive immersion can be affected negatively due to the disruption of the player's thought process.

2.4 Spatial Immersion

Spatial immersion is about having a game world where the player is in control of the movement of a game object, such as a car in a racing game, requiring her to be aware of the object's surrounding to make decisions about its movement and other actions. The easiest approach to achieve spatial immersion is through controlling the game object via a first-person perspective. Although it is harder to implement, other perspectives, such a third-person perspective, are able to achieve spatial immersion as well (Björk and Holopainen, 2004).

Spatial immersion is most common in real-time games, but it can also exist in turn-based games, where it is acquired though the planning of the moves of the player's units (Björk and Holopainen, 2004).

2.5 Sensory Immersion

The audiovisual experience is an important part of video game immersion according to Ermi and Mäyrä (2005), and it is through the advances in this area that people easily can recognize how games are evolving over time.

Visual

Good graphics in a video game can make the game more appealing, and having the in-game view provide relevant and pleasing views of the game world is associated with good gameplay. However, people have different opinions of what good graphics is. For example, some prefer a style similar to cartoons, while others prefer a more realistic approach. The differences between these graphic styles are illustrated in Figure 2.4. Furthermore, sensory immersion benefits greatly from having a large screen, thus covering a large area of the player's field of view (Ermi and Mäyrä, 2005).

Figure 2.4: *This figure illustrates the differences between cartoon graphics and realistic graphics. The image to the left is from Battlefield Heroes (DICE and EASY, 2009) and uses cartoon graphics, the right image is from Battlefield 3 (DICE, 2011) and uses more realistic graphics.*

Audio

Sensory immersion is increased through sound in two ways (Huiberts, 2010). The first is by enhancing the player's enjoyment by providing her with pleasing sounds. The second is by providing sounds that not only increases the player's feeling of presence in the game world, but also can improve the environment of the game world, making it feel more dynamic. This is further explained in Chapter 3.5: Sound Design.

Although audio is mainly a part of sensory immersion, it can increase other types of immersion as well (Huiberts, 2010). Emotional immersion can be increased by providing in-game characters with sound, such as speech, making them appear more living and increasing the player's empathy towards them.

2.6 Summary

In this chapter, a definition of immersion has been provided which will be used throughout this thesis, if not otherwise specified, when referring to immersion.

Immersion by itself is a vast topic, even when constrained to a video game context. To specify video game immersion, this thesis has used immersion types defined in the SCI and the ECSS model. To cover the immersion types relevant to this thesis, a combination of these two models were used since existing models overlap. This combination resulted in the following immersion types: emotional immersion, cognitive immersion, spatial immersion, and sensory immersion.

Emotional immersion was further divided into three subcategories: storytelling, ownership, and emotions. These three subcategories cover different methods of increasing emotional immersion, such as the telling of a story and providing the player with feelings by the use of anticipation or surprise.

Cognitive immersion is about providing the player with a problem or a situation that require the player to consider what options provide a solution. These problems or situations need to provide an appropriate level of difficulty, otherwise the player may find herself bored or in a state of analysis paralysis.

Spatial immersion requires the player to be in control of the movement of a game object within a game world, forcing her to navigate it in relation to other game objects.

Sensory immersion covers the basic human senses, such as visual and aural stimulus, and how these can be used to increase the perceived game immersion.

Although this is not a complete description of game immersion, it covers the needs for the analysis in this thesis.

Chapter 3

Game Design for Immersion and Enjoyment

Whether games have been intentionally designed to follow a certain pattern or not, they still consist of design choices made by the designer. Game design is, however, not trivial to define; not only is *game* hard to define but so is *design*, which's definition changes depending on the context it is used in (Findeli, 1995). The definition used in this report is coined by Salen and Zimmerman (2003), who state: "Design is the process by which a designer creates a context to be encountered by a participant, from which meaning emerges".

Since the aim of this thesis is to create a horror game, design decisions to create an immersive, believable, and logical player experience had to be made. In addition to immersion, a main goal was for the player to be able to, on her own accord, derive what tasks had to be performed for the game to proceed, thus the experience also had to be intuitive.

To achieve such immersion and accessibility while still allowing leeway for experimentation and artistic freedom, a futuristic and minimalistic space ship setting was decided upon in the concept stage of the project. This chapter covers such design choices made in *Breach* and, more importantly, why they were made. The design areas presented are: World Design, Horror Design, Objective Design, Design for Head-Mounted Displays, and Sound Design.

3.1 World Design

Every game contains a world in which the game takes place, consisting of one or multiple levels in which the player interacts with the game elements. There are multiple instances of video games containing the traditional design of having more than one level (*e.g.* Super Mario Bros 3 (Nintendo EAD, 1988)), but there are also games such as Just Cause 2 (Avalanche Studios, 2010) where the game world is a seamless experience, allowing the player to go from any location in the game to another without experiencing loading times. The latter has less interruptions and discontinuity of the world, which is an attribute that prevents the breaking of immersion in the game (Björk and Holopainen, 2004).

The implementation used in *Breach* was decided to be a single seamless world, used to enhance the immersion and prevent any frustration the player

could feel by the game loading a new level. A secondary effect of this design is that the game would not have to contain any level switching functionality. However, this design choice could potentially lead to further work required on performance optimization. Since the overall design was minimalistic and the target game time was about 15 minutes (in accordance with this thesis's limitations), the level was not required to contain a large quantity of level geometry, alleviating the previously noted performance optimization issue.

World layout

A major goal in the world design was to have the player as locationally aware as possible, while keeping a sense of confusion induced by being trapped in a closed, and somewhat confined, environment. To attain this, a logical layout of the rooms of the ship was designed, by which the player would recognize familiar locations and draw logical conclusions on where different parts of the ship are. The interior of the rooms contains indicators and screens, which inform the player on where in the ship she is, and also asymmetrically placed objects (*e.g.* furniture) for the player to use as navigational reference points in the game environment. This decision was made for the player to feel more immersed since the world follows an expected and believable layout.

To achieve a sense of confusion and confinement for the player, corridors were added as a means of transportation between the rooms, with some rooms having multiple entry/exit points (as seen in Figure 3.1); many of which were designed to be small and confining. At multiple points, the corridors branch off in different directions, which combined with some rooms' multiple entry/exit points create multiple paths for the player, allowing her to reach a certain location in the level in many ways (this also applies to any hazardous entities present in the game).

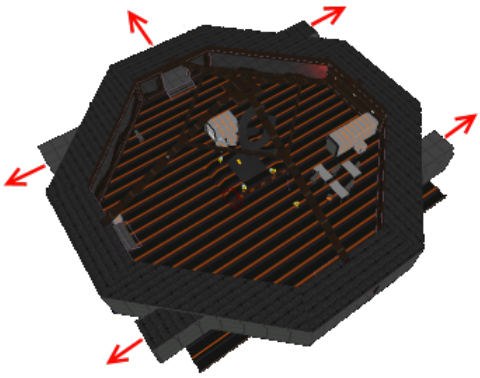


Figure 3.1: *Illustration of the multiple entry/exit points in the 'upper hub' room of the ship.*

3.2 The Horror Aspect

There are several subgenres to the horror game genre, each with different horror aspects. For the purpose of this thesis, two subgenres have been defined: The action-focused horror games such as Dead Space (EA Redwood Shores, 2008) and F.E.A.R. (Monolith Productions, 2005), and non-action-focused games such as Amnesia (Frictional Games, 2010). Both of these may feature horror elements in the form of non-playable characters that acts as hazards to the player (*e.g.* monsters and villains).

Action-focused horror games primarily have a horror aspect constituted by a scary environment and atmosphere, although this does not affect the gameplay itself. The horror elements may be counteracted by weapons provided to the player.

Non-action-focused horror games have a central horror aspect in gameplay in addition to environment and atmosphere. In contrast to the action-focused horror games, the player is given limited or non-existent means of defending herself against the horror elements, increasing the horror aspect and forcing the player to use stealth and caution.

The monster in Breach

Breach was designed to have limited means for the player to counteract the horror elements of the game, thus making it follow the design of non action-heavy horror games. The decision was made to utilize an alien monster as the main horror element (depicted in Figure 3.2), hunting the player in the game world without the player knowing why. The player is unable to defend herself against the monster, making it necessary for the player to run away or hide from the monster should it detect the player. The monster was designed to utilize an artificial intelligence (AI) which chases and hunts the player, enhancing the enjoyment of playing *Breach* since the monster's actions and movement are difficult to predict. The AI implementation of the monster and its abilities are further discussed in Chapter 4 - Artificial Intelligence.

Towards the end of the game, the player has the ability to indirectly deal with the monster by trapping it in the cargo hold of the ship and then ejecting the cargo hold into space, which destroys the monster, thus eliminating the main horror element in *Breach*.

Monster effects

As a way to further increase the horror aspect and the sensory immersion experienced by a player in *Breach*, monster related features affecting the gameplay and providing audiovisual feedback to the player are implemented.

To provide the player with some means of utilizing an additional light source, a flashlight was introduced for the player to find in the game world. This is the only means for the player to defend herself from the horror elements (by increasing visibility). However, the player becomes more easily detected when she is using the flashlight, which makes the use of the flashlight a risk/reward scenario for the player, increasing the cognitive immersion that the player will experience (as discussed in Chapter 2.3 - Cognitive Immersion).

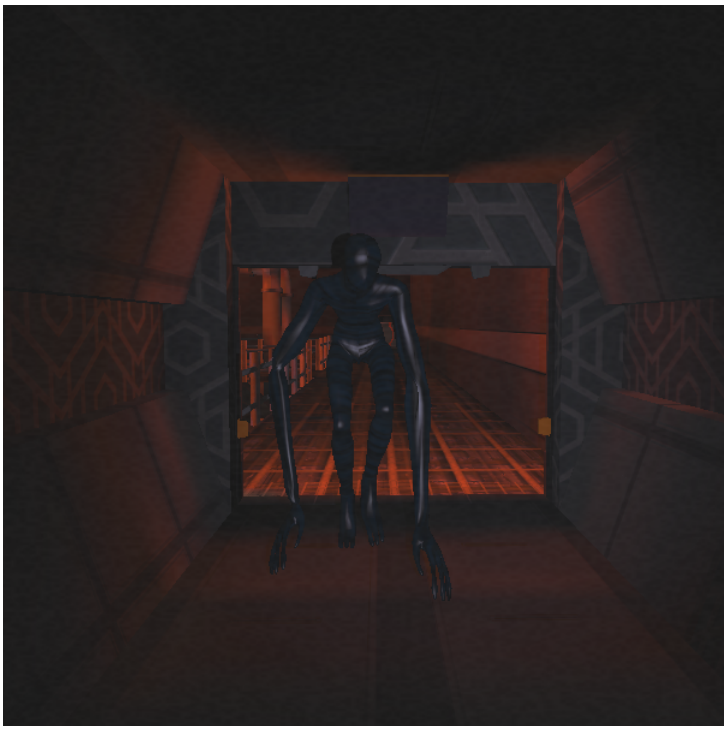


Figure 3.2: *Screenshot showing the monster, the main horror element of Breach.*

To enhance the horror aspects of the game and to provide the player with visual cues indicating that the monster is close by, three features have been implemented:

- Flashlight flickering (simulated instability in the flashlight's power supply, making the flashlight flicker and emit a fainter light)
- Full screen, visual, white noise (as seen in Figure 3.3)
- Horrifying aural noise (further explained in 3.5 - Audio design in Breach)

Each feature increases in intensity the closer the player is to the monster, with the intensity function being inversely dependent on the distance between the player and the monster. They are all provided by the game environment and not shown in a heads-up display, which has been shown to be preferred by most gamers in most genres, creating a better game experience (Jørgensen, 2012).

Lastly, the speed of the monster was designed to initially make the monster avoidable while still hazardous to the player. As stated in 3.3 - Objectives, the monster is time dependent, by which it is designed to have its speed increasing with time. Specifically, the monster becomes five percent faster with each minute, eventually becoming too fast for the player to avoid. Furthermore, when the monster chases the player it becomes faster by fifty percent, simulating that the monster is running.

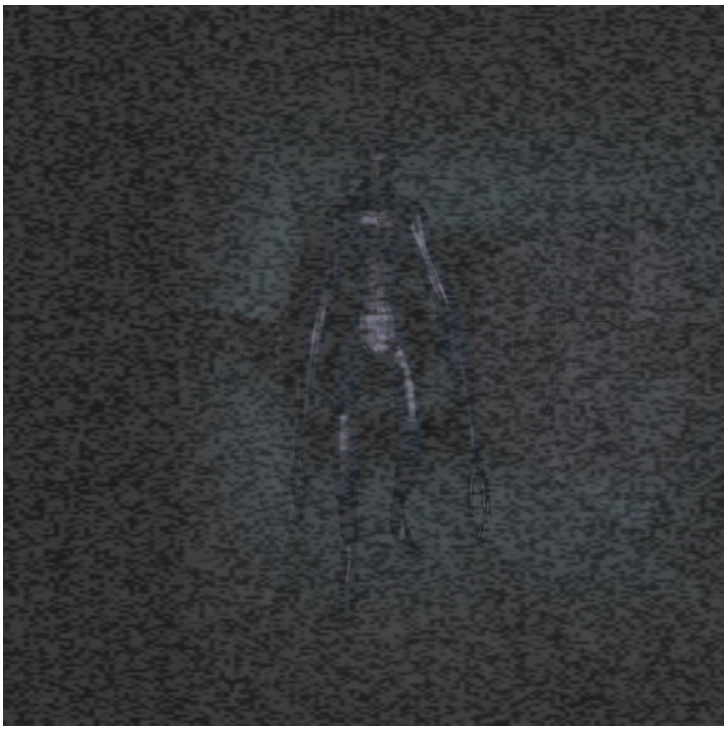


Figure 3.3: *Screenshot showing the monster being close to the player, thus filling the screen with semitransparent white noise.*

By having a monster which affects both gameplay and the perceived horror, an immersive and scary experience is reached in *Breach*, stimulating cognitive immersion and providing a hazardous and frightening element.

3.3 Objectives

All computer games feature objectives which the player will have to complete in order to advance in the game (Adams and Rollings, 2006). These objectives vary in scope and importance depending on the particular game they are in. For *Breach*, it was decided to make the amount of areas available to the player at a given time dependent on how many objectives of the game that the player has accomplished, creating a sense of progress for the player when objectives are completed.

As seen in Figure 3.5, there are two main objectives in *Breach* that are needed for the player to win the game: to divert the course of the ship, and to lure the monster into the cargo hold and then eject the cargo hold from the ship. These two main objectives correspond to the two player hazards in the game: an asteroid on a collision course with the ship (shown in Figure 3.4), and a monster that is hunting the player. Both objectives are time dependent, meaning that they have to be completed in a limited amount of time for the player to be able to win the game. This design choice was made to give the player a feeling of distress and urgency, forcing the player to take action, which is appropriate in a

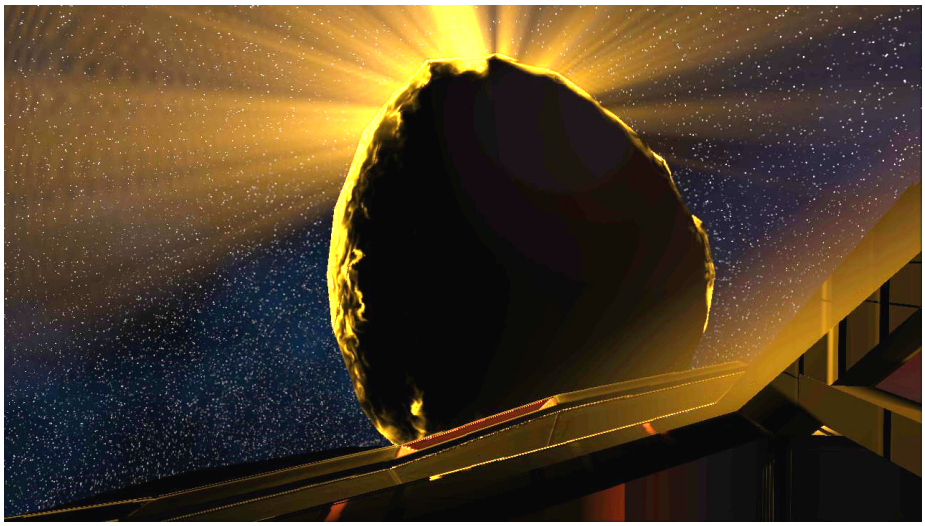


Figure 3.4: *Screenshot of Breach, showing the asteroid on collision course with the ship.*

horror game context. This section covers how to communicate objectives to the player in *Breach*, as well as the reasons for not including secondary objectives.

Communicating objectives

The traditional approach of communicating objectives to the player is to either display information on a player's Heads-Up Display (HUD) or in a separate objective log menu. However, this is not suitable for *Breach* as it would lead to a lowered level of emotional immersion and issues with displaying the game on head-mounted displays (as discussed in Section 3.4 - Movement in Virtual Reality). Since there are a limited amount of means of instructing the player what she should do next without breaking the perceived immersion, much consideration has been made to make the objectives of *Breach* logical and easily deducible. The player should, by using the information provided indirectly in the game, be able to come to a conclusion what her next objective should be. For instance, in the beginning of the game, the power of the ship is malfunctioning leading to a lot of doors being inaccessible. By exploring the areas available to the player she will find that a power core (depicted in Figure 3.6) in the reactor room is missing, enabling the player to draw the conclusion that if she finds the missing power core, brings it to the reactor, and insert it into its designated slot (as shown in Figure 3.7), power can be restored. Mental challenges such as this increase both the cognitive and emotional immersion the player experiences, since the player is presented with new information about the game world but have to use their cognitive skills to figure out the next course of action, as discussed in Chapter 2 - Immersion.

One issue with this approach is that not all players may make this logical conclusion on their own. There are three design choices made to counteract this problem in *Breach*. The first is the fact that the game environments in *Breach* are limited in their scope and the player will, by exploring the environment,

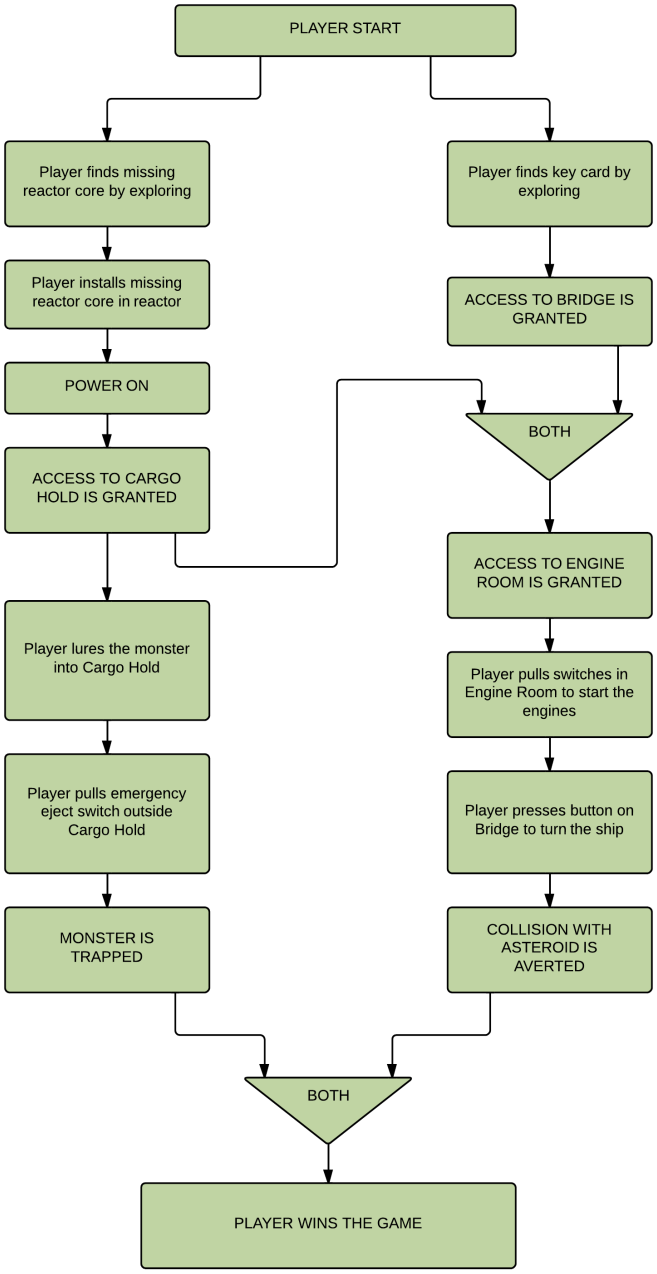


Figure 3.5: *Flow chart of the objectives and player actions in Breach, from the start of the game until the player has won the game. Text in upper case marks objectives that have been completed by the player actions leading to them.*

Figure 3.6: *Screenshot of Breach, showing the power core that the player will search for and pick up.*

eventually find the lost power core even if she has not realized that the power needs to be restored. Secondly, there are a limited number of pickup-able objects in *Breach*, and they are all emitting a faint light to make them stand out in the game world. Thus, since *Breach* have few pickup-able objects and they all stand out, the player will most likely understand the importance of the power core when she finds it. Thirdly, a helping system in the form of computer screens, which are placed in each room of the ship, was implemented to help the player better understand the current objectives. These screens show the current status of the ship, *e.g.* "Power offline" or "Collision imminent", but they do not explicitly tell the player the solution to these issues. An example of these screens are shown in Figure 3.8.

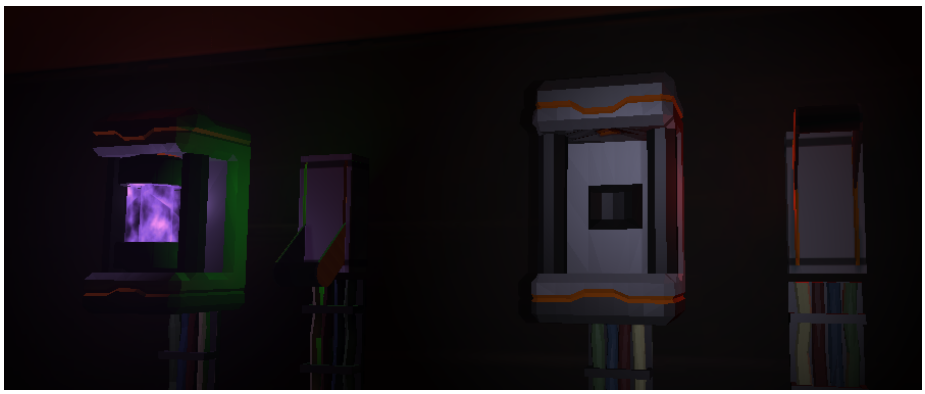


Figure 3.7: *Screenshot of Breach, showing an occupied power core slot next to the empty one.*

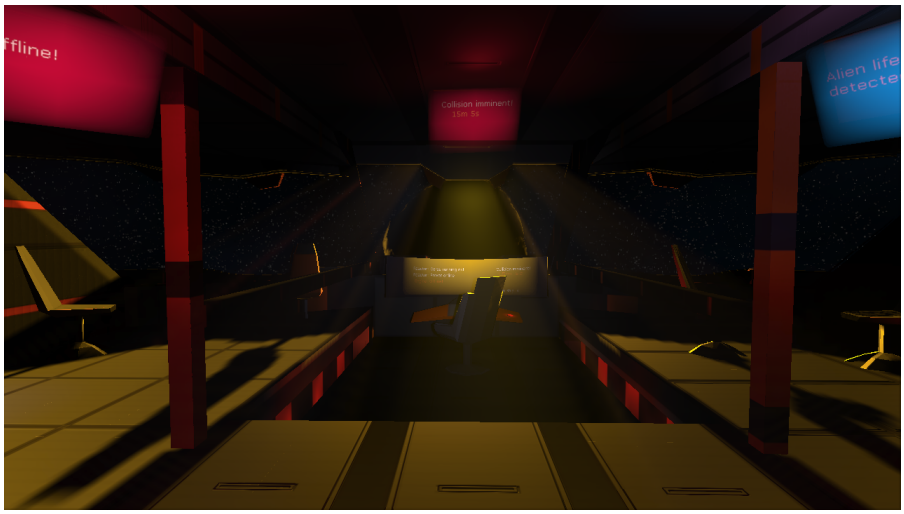


Figure 3.8: *Screenshot of Breach showing the bridge of the ship and some of the computer screens that convey information about the game state to the player.*

Another issue with this approach is that while the player may see the problems leading to the objectives (for example, the doors not opening leading to the "restore power" objective), the player may not think of the particular solution to the problem that is implemented in *Breach*. A player may for example see the monster and start looking for a gun to defend herself with instead of thinking of the solution offered by the game, which is to trap the monster in the cargo hold of the ship. This issue could partially be solved by implementing alternative solutions to the problems facing the player in the game, thus increasing the chance that the game supports the solution that the player comes up with. It is, however, impractical to implement all actions conceivable by all players, since the number of such actions is virtually infinite.

Secondary objectives

Many video games have secondary, optional, objectives that the player can pursue (Andersen et al., 2011). These objectives often exist to give the players something else than the main objective to work on, and to increase the expected playing time. However, it was opted to not have such secondary objectives in *Breach*. A study by Andersen et al. (2011) shows that secondary objectives can have a negative effect on the playing time, making players quit playing earlier than if the game would not have secondary objectives. Further, having secondary objectives in *Breach* could make the player less focused on the primary objectives of the game, leading to a loss of emotional immersion for the player.

3.4 Design for Head-Mounted Displays

Breach is designed to be used with the Oculus Rift (depicted in Figure 3.9), a Head-Mounted Display (HMD) developed by Oculus VR. The main difference

between HMDs and regular computer displays is that when using an HMD, the screen covers the entire field of view of the player, letting the player see only the virtual world and not the real world. This increases the emotional immersion of the player in a first person game, since the player will see the game world through the eyes of the player's avatar, creating a stronger emotional bond between the player and her avatar. Furthermore, the sensory immersion experienced by the player will be increased since the player will not be distracted by real world visual stimuli.

Figure 3.9: *The Oculus Rift, the HMD used during the development of Breach.*

The Oculus Rift also provides head tracking which allows the player's view in the virtual world to correlate to the movement of her head in the real world. This increases the spatial immersion of the player, due to the player using natural body movements to control the in-game view.

Displaying information on HMDs

Breach is developed targeting HMDs as the primary display technology, which has effects on how the game objectives and other game information, such as player status, are communicated to the player. In traditional computer games designed to be displayed on a monitor, game information is often displayed as a part of a Heads-Up Display (HUD) or available in a separate game menu (Adams and Rollings, 2006). This approach is not suitable in a game designed for a HMD, such as the Oculus Rift, due to two reasons.

The primary reason is that HUDs are difficult to implement correctly when displayed on an HMD, since they do not exist in the natural field of view of the player. Therefore they can decrease the perceived immersion, due to the player

being constantly reminded that she is playing a video game. This decrease in immersion is also present using regular displays but is amplified in an HMD context, since when using the HMD the player will only see the game environment and not the real world.

The secondary reason is connected to the current technological limitations of contemporary, consumer grade virtual reality HMDs (such as the Oculus Rift). Since they have a limited screen resolution, the pixel density of the view area is low compared to traditional monitors (Oculus VR, 2013). This leads to difficulties in displaying information (*e.g.* text) on the HUD since it needs to be large enough to be visible, but small enough to not interfere with the game view.

Since *Breach* does not feature quantifiable resources, such as player health or ammunition, the need for an in-game HUD is diminished. The only instance in which changes are made to the state of the player such that it would be appropriate to show them in a HUD, is when the player picks up an item. However, this happens rarely and since an added HUD would infer a lowered perceived immersion for the player, it was thus opted to not include a HUD in *Breach*. Without a HUD, the view the player sees on the screen is the same as the view of the player's avatar, leading to a stronger connection between the player and her avatar, thus increasing the player's perceived emotional immersion.

The use of the Oculus Rift was found through testing by the development team to greatly increase the percieved level of immersion due to the visual stimuli and the feedback from real-world physical movements of the player.

Movement in Virtual Reality

Developing a game that will be used with an HMD featuring head tracking creates an opportunity to implement alternative control schemes that can not be used when designing for a standard display.

One such control scheme usable in games with a first-person perspective, such as *Breach*, allows the player to use the HMD in order to look in a different direction than the direction that the player's avatar is moving in. This control scheme was tested in *Breach*, but was found to induce a sense of discomfort and dislocation for the player. Because of this, the decision was made to not use it in *Breach*. Instead, a control method more similar to the one generally used in first person games were implemented, making the rotation of the player's head in the real world correspond to the rotation of the entire body of the player's avatar in the virtual world.

3.5 Sound Design

As a prominent part of sensory immersion, sound is an important aspect of any video game (Grimshaw et al., 2008). To reason about sound in video games, several sets of terminology have been developed.

Theories and terminology

Originally conceived for the purpose of analyzing audio in movies, the terms *diegetic* and *non-diegetic* (also known as *extradiegetic*) refer to sounds that orig-

inate from inside the movie world and sounds that are heard only by the audience viewing the movie, respectively. This paradigm is not entirely applicable to sounds in video games, since sounds in games are rarely purely diegetic (Ekman, 2005) and non-diegetic sounds in games may influence the player's actions in the game, making the non-diegetic sound affect the diegetic part of the game (Jørgensen, 2007), resulting in a *diegesis feedback loop*. Jørgensen (2011) argues that the use of the terminology *diegetic* and *non-diegetic* in computer game contexts is bad practice and therefore coins the word *transdiegetic* to bridge the gap between the two.

This implies that film sound theory is not applicable to computer game sounds, and thus a separate theory is required. The *IEZA model* (also known as the *IEZA framework*), as introduced by Huiberts and van Tol (2008), is a model used for analysis of game audio, and it provides guidelines for how to design audio so that it affects the gameplay in a desired fashion.

The IEZA model

In essence, the IEZA model describes the auditory environment of a game using a two-dimensional space to categorize the audio; *diegetic* versus *non-diegetic* (sound that originates from the game world, and sound that does not, respectively) and *activity* versus *setting* (sound that corresponds to actions in the game world and sound that is played independently of actions in the game world, respectively).

These are in turn used to define four discrete categories, or domains, in which all audio in a game can be placed, as seen in Figure 3.10. These categories are:

Interface sounds are non-diegetic activity sounds. These correspond to game actions but can not be heard by the in-game avatar, such as HUD and menu sounds. There are exceptions to this; Dead Space 2 (Visceral Games, 2011), for example, features fully diegetic GUI elements due to them being rendered in the game world (as seen in Figure 3.11).

Effect sounds are diegetic activity sounds. These can be heard by the in-game avatar, and corresponds to actions in the game world. Effect sounds are for example the sound of an engine in a racing game or the sound made when firing a weapon in a first-person shooter.

Zone sounds are diegetic setting sounds. These can be heard by the in-game avatar, but does not correspond to any specific actions in the game (*e.g.* ambient wind sounds outdoors or cars and people in a city environment).

Affect sounds are non-diegetic setting sounds. They neither correspond to actions in the game, nor can be heard by the in-game avatar. An example of this is music that plays in the background.

Even though the IEZA model does not take transdiegetic properties of game audio in consideration, it is adequate for use in *Breach*. The reason for this is that *Breach* does not feature sounds from the Interface domain and thus avoids the *diegesis feedback loop* that generates transdiegetic sounds.

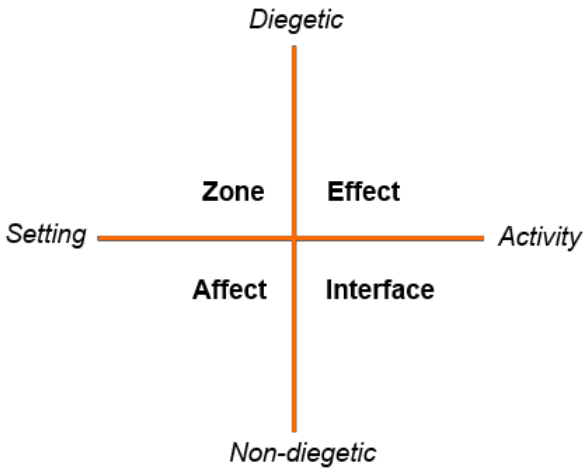


Figure 3.10: *Illustration of the IEZA dimensions and categories.*

Figure 3.11: *Screenshot of Dead space 2 (Visceral Games, 2011), showing the diegetic GUI of the game.*

Immersive audio

Huiberts (2010) shows that a significant portion of gamers regard audio as a positive factor for immersion. As described in Section 2.5 - Audio, there are several aspects of audio which can increase immersion in a game.

A study was conducted by Huiberts (2010), showing that the most prominent perceived effects of audio on immersion is the enhancement of cognitive, sensory, and emotional immersion. Cognitive immersion is enhanced as an effect of the

audio altering the tension in anticipation of upcoming events, while sensory and emotional immersion is enhanced as a result of the audio providing a feeling of presence and atmosphere in the game world.

The same study also shows that when the immersion is broken due to shortcomings in the audio, it is most likely caused by music that is inappropriate in the situation. Other reasons include sounds that are, due to the context, unrealistic, as well as sounds that do not correspond well to the actions in the game and sounds that make it too obvious that the player is merely playing a game.

Increasing immersion using audio

Since immersion can be increased by applying sound, Huiberts (2010) has, by utilizing the IEZA model, developed two design techniques: *optimization* and *dynamization*.

Optimization refers to designing audio in such a way that it helps the player by providing her with information about the game. The main contribution to *optimization* is the *Interface* domain of the IEZA model, followed by the *Effect*, *Zone*, and *Affect* domains in descending order. *Optimization* sounds can for example be GUI sounds that signal confirmation, denial, selection, and progress.

Dynamization refers to designing audio such that the gameplay is made more thrilling and intense by it. This is mainly performed by *Affect* sounds, followed by *Zone*, *Effect*, and *Interface* in descending order. Examples of *dynamization* sounds are atmospheric sounds (any kind of sound that indicates events in the game) and most music.

Optimize	Dynamize
Affect	Interface
Zone	Effect
Effect	Zone
Interface	Affect

Table 3.1: *The four domains' contributions to optimization and dynamization audio, in descending order of importance.*

The primary difference between optimization and dynamization sounds is that optimization sounds normally originate from the domains that concern *Activity*, while dynamization sounds stem from the domains that concern *Setting*. Table 3.1 displays the domain origins of optimization and dynamization sounds.

Non-linear audio

Certain types of audio may be helpful to use when trying to convey feelings of fear and horror in the listener. One such type is *Non-linear audio* which is, in essence, audio with chaotic and sometimes harsh characteristics (Blumstein et al., 2010). Examples include distortion when turning audio volume up too much and human screams (as opposed to, for example, singing). Some sources

suggest that non-linear audio used in certain horror movies may share characteristics with warning calls in primates, aiding in the creation of fear and panic feelings in the listener (Blumstein et al., 2010; Grimshaw, 2010). For instance, non-linear acoustics are present in the cries of human infants (Fitch et al., 2002), which presumably elicit a basal emotional response in an adult.

Audio design in Breach

Since the main goal of *Breach* is immersion, the game features very few non-diegetic audio elements. Furthermore, since none of *Breach*'s non-diegetic sounds are activity sounds, no trans-diegetic sounds are present. As covered in Section 3.4 - Movement in Virtual Reality, diegesis helps achieve emotional immersion by increasing the player's feeling of presence in the game world when using HMDs.

The only non-diegetic sound featured in *Breach*, as can be seen in Figure 3.12, is the droning background ambience. This has the sole purpose of conveying mood, and it is entirely within the setting area of the setting-activity spectrum (effectively making it an Affect sound). To avoid repetition of the droning mood sound, it is played back using three individually looping channels (one for each subtrack) of different lengths, ensuring that the three subtracks are not played back in a synchronized manner.

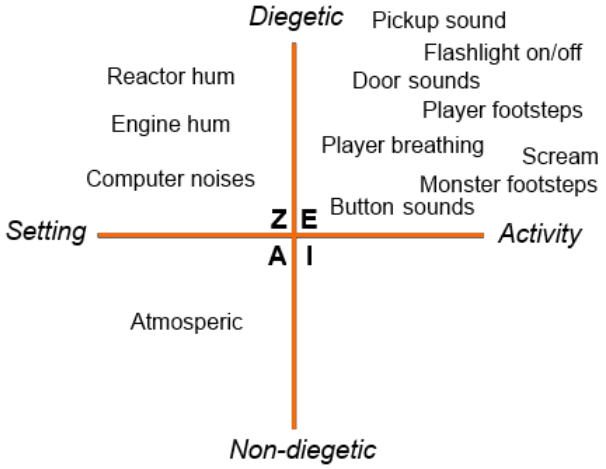


Figure 3.12: *The sound effects of Breach, categorized in their IEZA domains.*

The audio in *Breach* can generally be considered as dynamization sounds as its purpose is to convey suspense and also because no Interface domain sounds exist in the game. Some optimization audio is present, mainly in the Effect domain. As information is not conveyed by HUD elements, optimization sounds are regarded as a complement to the information being conveyed to the player by other means (as described in Section 3.3 - Communicating objectives).

To induce fear in the player, *Breach* employs sounds which display non-linear characteristics. These include a horrifying aural noise that grows in relation to

the player's proximity to the monster and the scream that is heard when the player's avatar is killed.

3.6 Summary

This chapter has been a presentation of game design and design choices made for *Breach* such that the game achieves an immersive and enjoyable experience using techniques discussed in Chapter 2 - Immersion.

To achieve a sense of confusion and confinement (inducing horror) for the player, the world in *Breach* features branching corridors and multiple entry/exit points in rooms, thus creating multiple paths for the player to reach a certain location in the level. Rooms are however logically placed and contain visual landmarks (such as the cryo pods shown in Figure 3.13) to help the player orient herself and achieve immersion.

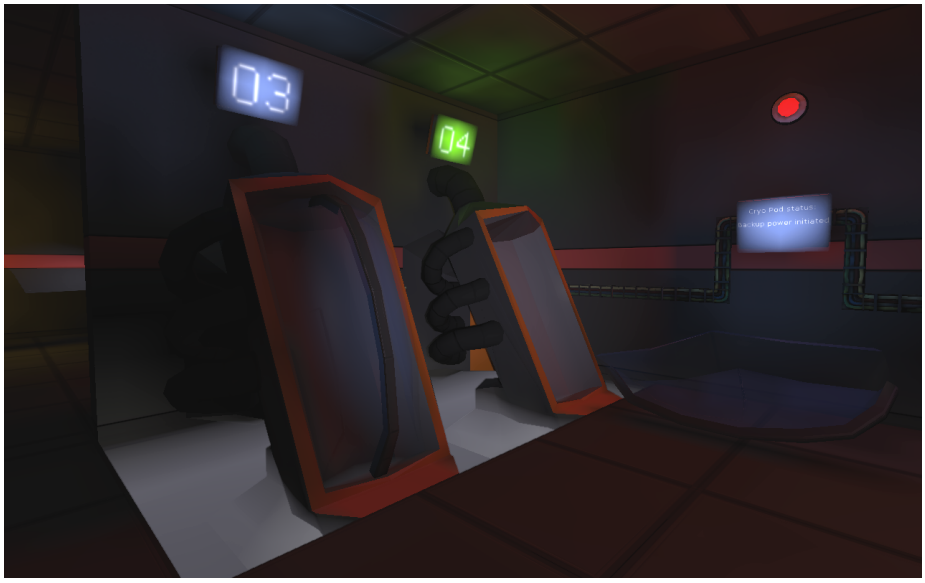


Figure 3.13: *Screenshot of Breach, showing the distinct visual landmarks of the cryo chamber of the ship.*

The gameplay of *Breach* is non-action-heavy to increase the feeling of distress and horror for the player, leading to a greater enjoyment in a horror context. The main horror element in *Breach* is a monster, which makes use of an AI to hunt the player through the game world. Although the player has no direct means of countering the monster, she is able to make use of an environment feature to defeat it. Furthermore, the player is able to use a flashlight to help her light up the game world, at the cost of her being more easily detected by the monster. To induce fear in the player, effects such as flashlight flickering and screen noise is activated when the monster closes in on the player.

Since this thesis' aim is to achieve a high level of immersion, the way information is presented to the player is important to consider (especially since the game is designed for the use of the Oculus Rift). By not explicitly informing the player about objectives (*e.g.* by not utilizing a HUD), cognitive immersion

is attained by forcing the player to use cues in the environment (*e.g* text on computer screens) to discern what the next course of action ought to be. A conscious decision was made to not include secondary objectives since they may backfire and distract the player from the main goals of the game.

The use of a head-mounted display, such as the Oculus Rift, when playing *Breach* was found to increase several types of immersion including emotional, cognitive, sensory, and spatial immersion. However, the usage of an HMD poses challenges in displaying text (such as parts of a HUD) on the screen, as well as challenges in implementing a control scheme that is enjoyable to the player. In *Breach*, these issues were solved by removing the HUD (showing game status in the game world instead), and implementing a control scheme familiar to players with previous experience in playing first person video games.

Audio can be very beneficial to the immersion of a game, and the audio of *Breach* is designed to enhance the cognitive, sensory, and emotional immersion by providing a very diegetic experience, designed with the IEZA model in mind. The audio in *Breach* possesses both optimizing and dynamizing properties in order to convey information to the player as well as to make the game more thrilling. Some sound effects have non-linear characteristics in order for them to enhance the horror experience.

Chapter 4

Artificial Intelligence

In *Breach*, there is a Non-Player Character (NPC) hunting the player, and an Artificial Intelligence (AI) is used to control the NPC. The usage of an AI gives the player the illusion of the NPC being intelligent (Buckland, 2005). While research on AI generally focuses on simulating the human mind (Rouse, 2004), AI in video games focuses on providing an enjoyable gaming experience. The quality of the AI in a video game has a significant impact on how enjoyable and challenging the game is (Rouse, 2004), and with the decision to use an NPC in *Breach*, the need for a believable AI emerged.

This chapter presents the methodology used when developing the AI in *Breach* by explaining the implemented features and their contribution to a believable, enjoyable, and immersive game experience (which is a significant part of the problem in this thesis). Particularly the navigation system is explained with its advantages and disadvantages, along with the behavioral patterns of the AI (*i.e.* patrolling and chasing the player).

4.1 Development

During the early stage of the development the team was under the impression that all AI functionality had to be written from scratch. Therefore the design for the initial AI was very simple with functionality such as being able to spawn, chase the player, and de-spawn depending on different conditions *e.g.* the player's position in the game world or elapsed time. However, due to the decision of developing the game in UDK, a more advanced AI could be created since UDK comes with integrated AI functionality (Epic Games, 2012b), such as different ways of navigating throughout the map.

However, a majority of the AI functionality included within the AI classes in UDK are based upon a first person shooter game, and they have functionality which is not utilized in *Breach*, such as vehicle control and shooting functionality. Furthermore, the AI classes are complex and not well documented, thus understanding, troubleshooting, and implementing new functionality into the AI classes is difficult. Therefore a new AI class was created, utilizing the AI features and functions in UDK that was found useful for the *Breach* AI, *e.g.* path finding for navigation.

4.2 Navigation

As the AI needs to be able to navigate throughout the game world, the navigation has to be near optimal in order to provide the player with the illusion of intelligent movement (Demyen, 2007). UDK provides functions to support the implementation of navigation.

An attempt was made to utilize an advanced function in UDK, which triggers the AI when a player moves nearby, but due to insufficient documentation, the development team decided that time required to implement this function was not worth the effort. Instead, a simple navigation was implemented, providing the AI with the ability to move towards the player. However, since simple navigation did not account for any obstacles, this could easily cause the issue of the AI getting stuck.

To address this problem a more complex navigation was needed, leading to the implementation of path finding. UDK supports two different types of path finding: waypoints and navigation meshes (Epic Games, 2012a).

Waypoints

The waypoints system, shown in Figure 4.1, works by linking together path nodes and building paths between them throughout the game world. The AI then navigates within the game world by following these paths (Epic Games, 2012f).

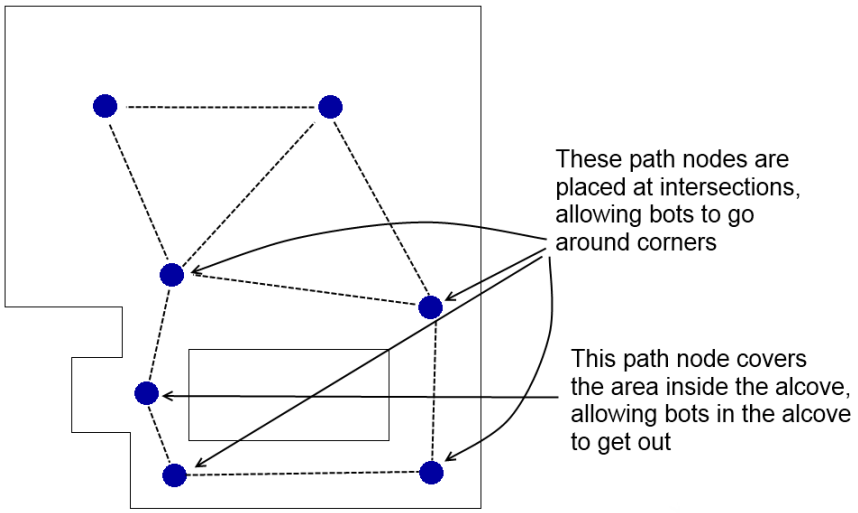


Figure 4.1: *An implementation of waypoints where path nodes (shown as blue circles) are placed on a map, enabling the AI bots to navigate between them by walking from one node to another (illustrated as the dashed lines between the nodes).*

The waypoint system is useful when the game world contains unnatural means of transportation, such as teleporters, since path nodes can be placed directly on the teleporters connecting two nodes. This enables the AI to use the unnatural means of transportation while navigating, which can not be done by

using navigation meshes.

Navigation meshes

A navigation mesh is defined as a two-dimensional region representing walkable areas in an environment (Snook, 2000), and is widely used to efficiently generate realistic paths (van Toll et al., 2011). As seen in Figure 4.2, they do not generate a path in the same way as the waypoint system does, which results in a less predictable AI (Epic Games, 2012e). For instance, if the AI is going to pass through a room it will take the shortest path instead of a predefined path when using navigation meshes.

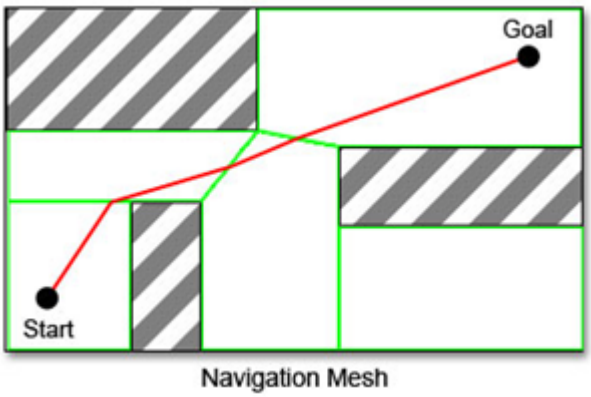


Figure 4.2: *Navigation mesh generated on a small map where grey striped areas are obstacles. Navigable areas, shown as green polygons, are used by the AI to calculate the shortest path to the goal, illustrated by a red line (Epic Games, 2012e).*

To generate this, an algorithm provided by UDK is used, filling the game world with polygons representing walkable areas, as seen in Figure 4.3. The AI then traverses the generated navigation mesh by moving through the polygons. Millington and Funge (2009) describes each polygon as a node, and that each node is connected to another node if they share an edge. The AI then navigates by walking from node to node.

Navigation lock

A major issue with both the waypoint system and the navigation mesh in *Breach* was discovered, with the issue being that the AI becomes locked in its path finding, meaning that when the AI is navigating throughout the map, no other AI code may be executed. The cause of this originated from the usage of two latent functions provided by UDK (Epic Games, 2012d). As a latent function is executed, the code will block until that function returns, and not execute anything else during that time. Due to the navigation functions provided by UDK being latent, the navigation functions do not return until the AI reaches the destination it is moving towards, causing the AI to be unable to do anything, for instance checking if the player is nearby as illustrated in Figure 4.4.

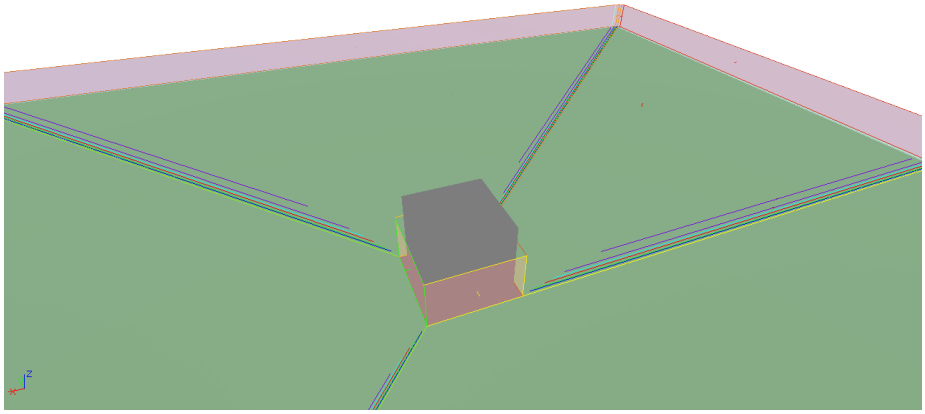


Figure 4.3: *A fully generated navigation mesh, where obstacles are enveloped by a red innavigable coating, and path-able areas are green. The lines on the floor illustrates the generated navigation mesh.*

The issue was solved by the use of timers with the ability to disrupt the latent function the AI is currently running (Beyond Unreal, 2008), which allowed the AI to execute code once again. However, this solution is not optimal since it depends on how often the timer is triggered; the more often the timer is triggered, the more often other code may be executed. For instance, the scenario in Figure 4.4 could be avoided by having a timer triggering code that reevaluates the path to the player. The higher the triggering frequency of the timer, the more rapidly the AI will start chasing the player directly.

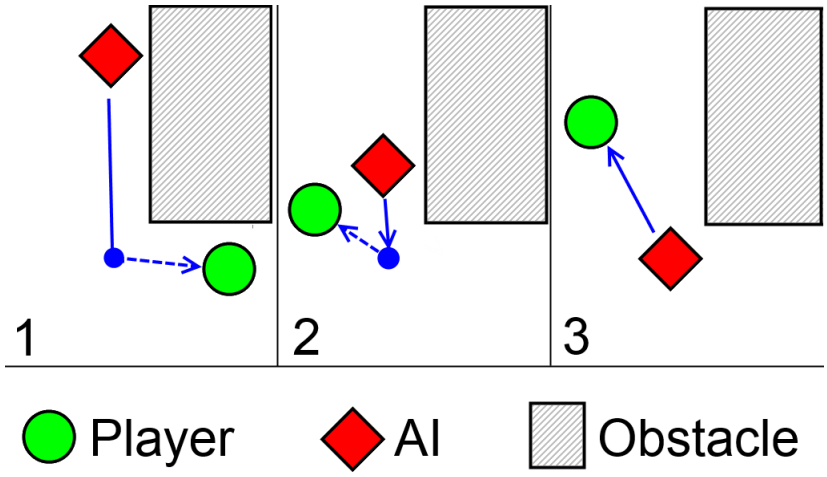


Figure 4.4: *This figure shows how the AI following the player get locked, being unable to recognize that the player has changed location until it arrives at the next node.*

Navigation in Breach

Breach needed to implement some sort of navigation for the AI, and as UDK provides two methods for AI navigation, waypoint navigation or a navigation mesh, the choice had to be one of these. Ultimately, a navigation mesh was chosen since the waypoint navigation provides no advantages (due to the lack of unnatural transportation points in *Breach*) and the navigation mesh is less predictable (thus providing a more realistic navigation).

Issues with navigation

The utilization of navigation meshes in *Breach* led to a number of issues with the AI navigation, such as small hallways being unnavigable and the AI becoming stuck in stairs.

To solve the issue with small, unnavigable areas, parameters used by the algorithm generating the navigation mesh were tweaked, *e.g.* by decreasing the AI's size, which the algorithm takes into account when generating the navigation mesh. Blocking volumes (invisible volumes in the game world which can block entities and navigation generation) to physically block the AI and its pathing were used to solve the issue of the AI getting stuck in stairs. This was primarily accomplished by adding a blocking volume on top of the stairs, as seen in Figure 4.5, making the navigation algorithm see the stairs as a slope instead of as a number of steps.

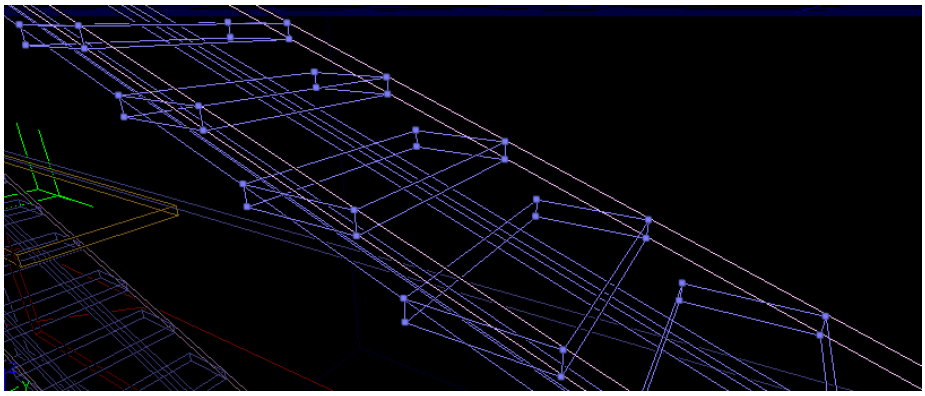


Figure 4.5: *The blocking volume (pink) is placed along the top of the stair steps (blue), effectively making the stairs a gentle slope.*

4.3 Behavior

In order for the AI to provide immersion and gameplay challenge for the player, it has to have a set of behaviors which makes the AI seem intelligent. The different AI behaviors are based on corresponding states. As seen in Figure 4.6 the AI in *Breach* has three different states implemented, excluding the idle state wich is the initial state. In this section, the AI behaviors of *Breach* (patrolling and chasing the player) are explained along with how they contribute to a better game experience.

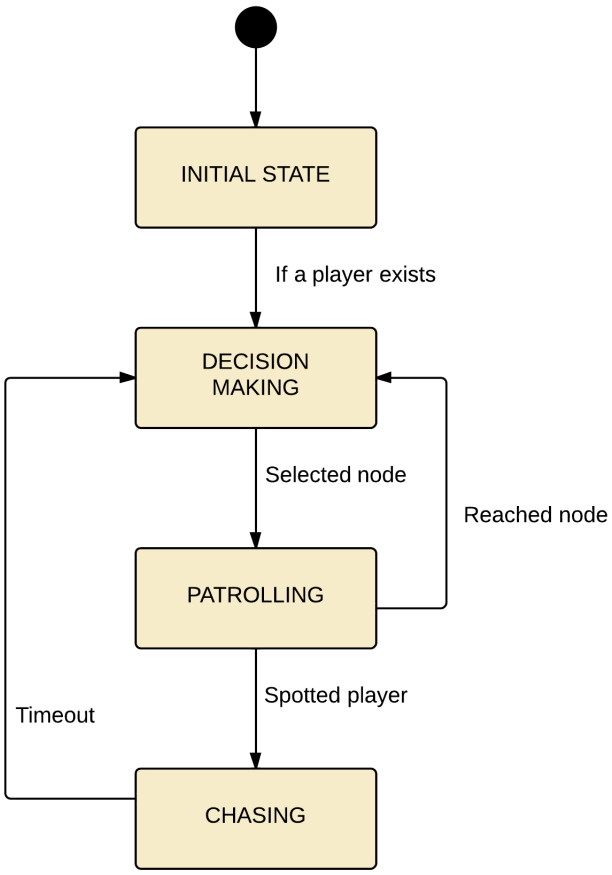


Figure 4.6: *Basic overview of the different AI states represented by rectangles, and the transition between states represented by arrows.*

Patrolling

Since the AI was supposed to be a hazard to the player, it had to chase the player. However, if it was chasing the player constantly with perfect information on the location of the player it would be too aggressive, and it would also not provide the player with the ability to outsmart it, thus neither providing the desired cognitive immersion nor enjoyment in *Breach*. The solution to this problem was to not give the AI information on where the player is at all times. However, it still had to be present at many places in the game such that the player was at constant risk of encountering the AI. Therefore, a patrolling behavior was implemented for the AI.

With the patrolling behavior, the AI is navigating the game world on its own accord, walking to random locations in the environment without having

the player as a target. This increases emotional immersion, by providing a more realistic and believable AI (the AI is seen as if it is out hunting for the player), as well as spatial immersion by further forcing the player to avoid the AI (as discussed in Chapter 2 - Immersion).

To create this patrolling behavior, the AI randomly chooses its destination from a set of pre-determined locations within the game world. To ensure that the AI will travel to all destinations regularly, the location selection process is not completely random, but uses a weighting system that lowers the weight of a location (and therefore the likelihood of it being selected as a target) when it has been chosen. The probability of a location being selected as a target approaches zero as the occurrences of it being chosen increases.

Chasing the player

When the AI is patrolling, it will eventually make contact with the player. In such a situation, the desired response is for the AI to start chasing her and ultimately catching up with her, ending the game.

For this, the monster needs the ability to detect the player and switch to the player-chasing state. Instead of merely using a simple detection radius, detecting the player when she is within a given distance from the AI, a visibility check was added. The visibility check determines if the player is "seen" by checking if there is a straight, unobstructed path between the AI and the player, which prevents the AI from detecting the player through walls.

In addition to the visibility check, a dynamic detection radius is implemented where the radius depends on two parameters: whether the player has a flashlight on, and if the player is sprinting. If either or both of these are true the radius will be greater, simulating the player being easier to detect when engaging in such activities.

When chasing the player, the monster is sprinting and receives a speed boost. Furthermore, a timeout is triggered when the AI is chasing the player but has lost visual contact with her for a specified amount of time; effectively simulating a behavior of the monster giving up the chase and regarding the player as lost.

With this behavioral pattern of the AI, the player can hide to avoid it, plan ahead to reduce the probability of detection, and escape the monster by evasion. This increases cognitive stimuli of the player which provides cognitive immersion. Furthermore, hiding and planning are gameplay elements which the player can choose to engage in instead of just fleeing from the monster.

4.4 Summary

In this chapter the subject of an Artificial Intelligence (AI) has been discussed in a video game context, focused on the implementation in *Breach* and the different issues that emerged during development.

Two types of UDK supported navigation, waypoint navigation and navigation mesh, are presented, particularly the navigation mesh since it is the system utilized in *Breach*. The navigation mesh approach was chosen due to it providing a less predictable experience, thus better resembling human navigation resulting in a believable experience.

In addition to navigation, the behavioural pattern used by the AI in *Breach* further improves the experience (both gameplay and immersion) by providing believable features simulating sensory input and decision making by the AI, such as visibility detection and dynamic detection radius.

Chapter 5

Content

All game projects, regardless of their nature, need content, such as art assets or game scripts, constituting the representation of the game world. When creating *Breach*, decisions had to be made regarding what content to include, and how the assets constituting this content should be created, to provide a sense of coherency and enjoyment for the player.

This chapter covers art based content (sound, models, levels, and textures) and gameplay scripts, used in *Breach* to provide a consistent look-and-feel and a believable setting.

5.1 Levels

To help decide the general layout of the spaceship that would form the setting of the game, multiple proposals of layouts of rooms were assembled. Afterwards, these layouts were discussed and a new collaboratively made room layout was agreed upon. This room layout consisted of nine different rooms explorable by the player: Bridge, Upper Hub, Cryochamber, Engine Room, Cargo Hold, Lower Hub, Reactor Room, Storage Room, and Workshop. Locked doors were added to some of the rooms and corridors, indicating that the spaceship may consist of other areas (such as living quarters) that are not accessible to the player when playing *Breach*. This creates the illusion of a bigger game world, while constraining the playable area to a reasonable size.

5.2 Models

Almost all of the models used in *Breach* were created using modeling software such as Blender (Blender Foundation, 1995) and Autodesk 3DSmax (Autodesk Media and Entertainment, 2009).

As the models were created by several people, and subsequently looked slightly different depending on the creator of each model, it presented a challenge to make them fit together in the setting of the game. To create a sense of coherency the same models could be used in multiple parts of the ship. For instance, a pipe model created for the engine room was used in the cargo hold as well. This enhanced the impression of the ship as a single unit instead of simply separate rooms connected by pathways.

The monster

The monster is the only model in *Breach* that is animated. It has three different walking animations to switch between depending on the height of the ceiling, since the monster is too tall to stand upright in most of the areas of the game world: one for walking upright, where the ceiling is higher than usual; one for walking in a corridor or other space with the standardized ceiling height; and one for walking through doors, which are lower than the standardized ceiling height. Figure 5.1 shows a comparison of a single frame from each animation.

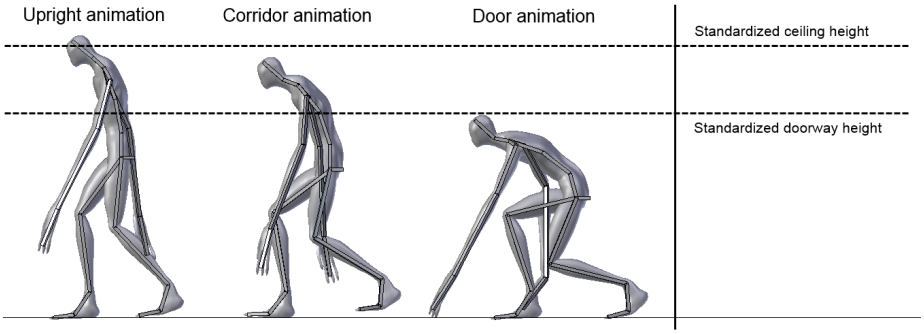


Figure 5.1: *The three animations of the monster side by side, each 10% into the animation cycle.*

5.3 Textures

In UDK, textures are not directly utilized in a game, but rather sampled by a material (a combination of textures and shader instructions) which then is applied to the desired geometry.

This section briefly covers the texture and material development process to highlight the adopted workflow, and the overarching design utilized when creating the visual presentation of *Breach*.

Texture creation

A few of the UDK-supplied textures were used in *Breach*. However, most of the textures in the game were created specifically for this project by the use of external image processing software. The difference in quality between project-created textures and the ones provided by UDK made it difficult to achieve a feeling of coherency, resulting in most of the supplied textures eventually being replaced by textured created by the development team.

When creating a texture, its intended usage was of great importance. While a generic texture for a wall or ceiling requires little work, a texture for a specific model is more complex. Models are UV unwrapped such that different parts of the texture corresponds to a specific location on the model (as seen in Figure 5.2). Thus, when texturing a model, the designer has to consider the unwrapping made by the modeler.

Furthermore, the material using the texture may need additional textures, *e.g.* normal maps (textures that modify the surface normals of an object during

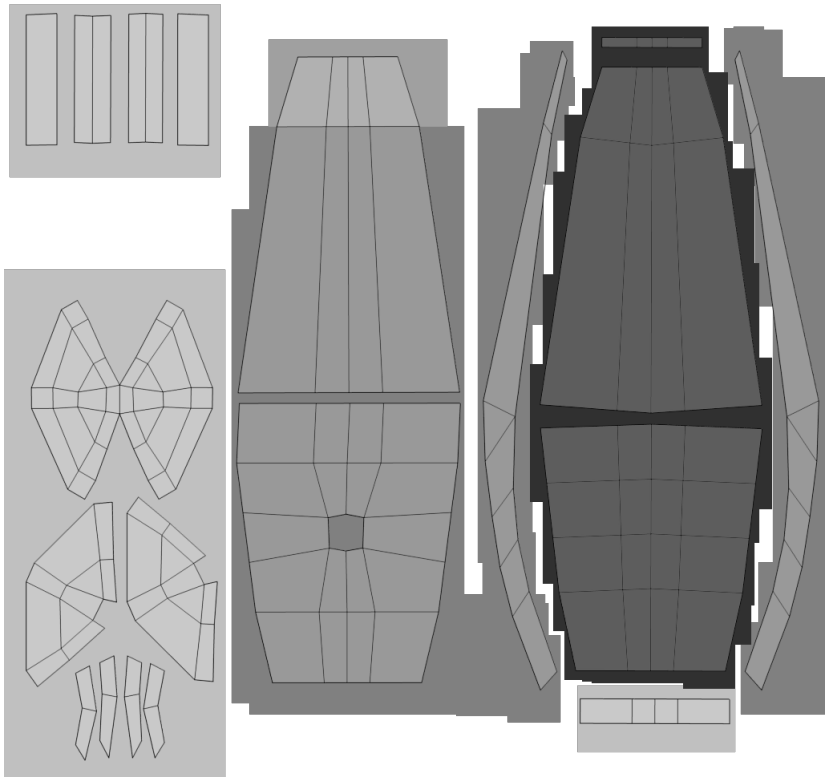


Figure 5.2: *Image depicting one of the textures for a chair model in Breach. The black lines constitutes the UV mapping used to specify what parts of the texture should be applied to a given part of the chair.*

lighting calculation to simulate bumps and wrinkles not modeled using polygons) and specular maps (Textures that specify the specularity, shininess, of a surface), used to increase the complexity and realism of the material.

Once all textures for a material have been made, they are imported into the UDK project and inserted into the new material (using the built-in material creation tool of UDK) which can then be applied to any geometry in the game.

Color theme

When designing the textures for *Breach*, the coherence of the colors used was a great factor of the final visual appearance. Most materials are therefore based on metallic gray, rubbery black, and matte orange colors, which provides a homogeneous and futuristic appearance (as seen in Figure 5.3).

5.4 Sounds

An important part of a video game is the sound effects, especially for a horror game, as discussed in Section 3.5 - Sound Design. While the initial plan was to

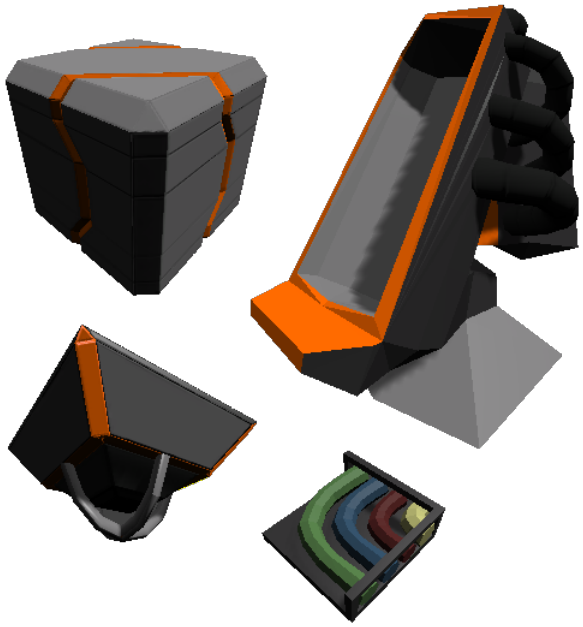


Figure 5.3: *Illustration of finished models in Breach, textured using the overarching color scheme.*

record all sound effects to be used in *Breach* during the development process, the desired sound quality required better recording equipment than was available. Therefore, sound effects for *Breach* were acquired from various free online sources.

The use of audio from different sources posed a problem, however, since coherence in style is desirable, and audio from different sources may vary in quality and characteristics. The sound effects were therefore processed using audio processing software to fit the mood of the game and the specific game situation where each sound effect was used.

5.5 Gameplay Scripts

Game scripts were implemented in *Breach* to increase the immersion and enjoyment of playing the game, by providing a dynamic game experience. The programming of the gameplay in *Breach* was done through two scripting languages, UnrealScript and Kismet, both of which are parts of UDK.

UnrealScript

Epic designed UnrealScript with gameplay programming in mind (Epic Games, 2012c), and it is through UnrealScript that the largest amount of gameplay has been implemented in *Breach*. Some of these implementations are of great importance to the game, such as the interaction functionality and the flashlight flickering.

Interaction functionality

Breach features the functionality to interact with the game world, such as picking up an object or activating a button. To interact, the player needs to look at an object and at the same time press a designated key on the keyboard. If the object can be interacted with and the criteria of an interaction is fulfilled, the interaction occurs. These criteria consist of the distance to the object being within a specific range, as well as the angle between the player's view direction and the vector between the player's view position and the object's position being within a specified angle. A visualization of the interaction's constrains are shown in Figure 5.4.

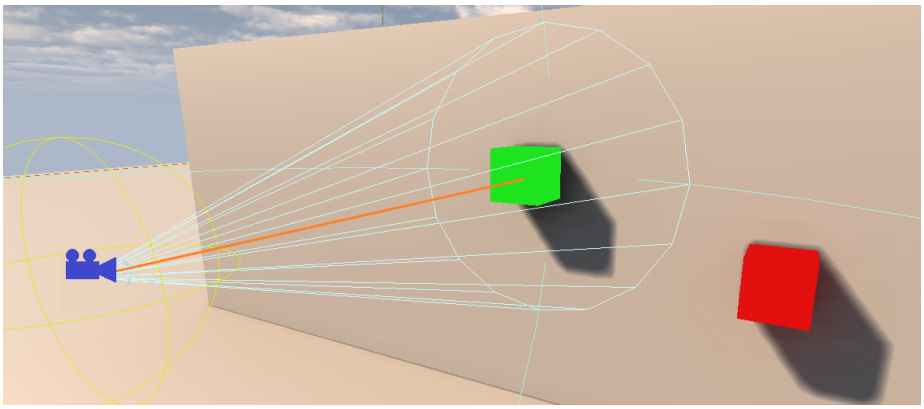


Figure 5.4: *Illustration of the interaction constraints in Breach. The player's view is illustrated with a blue camera, and is rotated in the direction of the green interact-able object. The light blue lines that emerge from the camera represent the boundary angles within which the player can interact with an object. In this image the green object is within the interaction boundaries while the red object is not.*

Flashlight flickering

A flashlight usable by the player in *Breach* was implemented by utilizing the lighting functionality provided by UDK. To provide the player with visual feedback indicating that the monster is nearby, the flashlight's light source was made to flicker, giving the player the impression that the flashlight is malfunctioning. Two implementations of the flickering were evaluated: predictable flickering and random flickering.

Predictable flickering used a sine function, where the light intensity varied over time from a fixed upper limit to a lower limit that depended on the distance to the monster. The graph to the left in Figure 5.5 illustrates how the light intensity would change over time.

The second implementation, random flickering, used a random function. It generated a random value within a range of possible light intensities, where (like predictable flickering) the lower limit depended on the distance to the monster. The light intensity of the flashlight would then approach this randomly generated value at a fixed rate. When the intensity reaches the generated value,

a new value is generated, repeating the process. This change of light intensity is illustrated in the right graph of Figure 5.5.

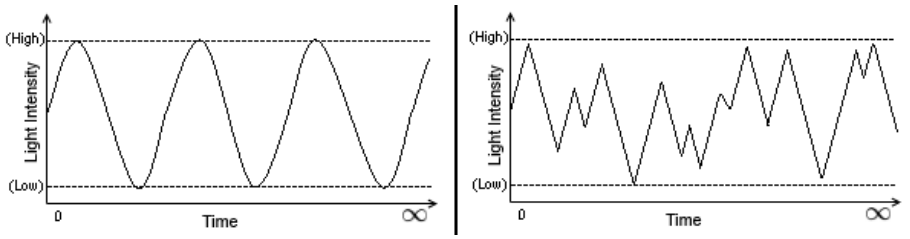


Figure 5.5: *The graph to the left shows how the light intensity changes over time when using predictable flickering, while the graph to the right illustrates the same thing, but with random flickering instead.*

Although both of these implementations worked as intended, the random flickering gave a stronger feeling of something being wrong with the flashlight, providing a higher level of immersion, when tested by the development team. This resulted in the decision to use random flickering instead of predictable flickering.

Kismet

Kismet is a visual scripting language providing developers with functions to control core elements of UDK, and is used to implement level specific functionality (Cordone, 2011). However, new features in *Breach* required additional Kismet functions (to be written in UnrealScript) not included with UDK.

One of the main features of Kismet is that it allows scripts to be triggered on different events, such as a player entering a certain area within a level. This could, for instance, be used to create automatic doors or trigger different monster animations.

Doors

Doors are used in *Breach* to give the player the impression of a dynamic environment by automatically opening when the player approaches. However, the doors needed more functionality than just opening and closing them. Some doors needed to be conditionally locked, *i.e.* locked until the player finds a keycard allowing her through, or the power being turned on. These extra conditions needed to be created as Kismet script functions in UnrealScript. These are then used in the visual Kismet environment to create scripts such as the one shown in Figure 5.6 which illustrates a door requiring the power being on in order to open.

Monster animation

As discussed in Section 5.2, the monster in *Breach* needs to switch between three walking animations in different areas of the ship. This was accomplished by adding trigger volumes, which are volumes in the level that triggers a Kismet event when an actor, such as the monster, enters or leaves it. This event can

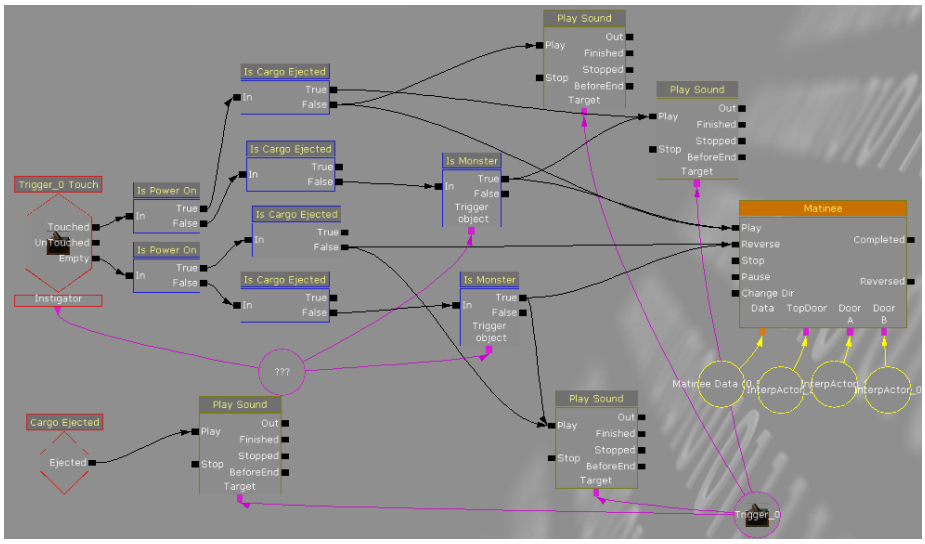


Figure 5.6: *An example of the structure of a Kismet door script which is dependent on the power being on.*

then be processed by Kismet to change the walking animation of the monster to a more appropriate one for that area.

5.6 Summary

This chapter has discussed the different ways content and assets were created to be used in *Breach*, and how they contribute to an immersive and enjoyable game experience.

Models, textures, and levels were created by different members of the development team. To acquire a coherent visual style and feel of the game world, models and textures created for usage in a particular room were also designed to be usable in other rooms throughout the game world. Furthermore, a futuristic and minimalistic visual style was adopted, minimizing the differences between assets created by different members of the development team. Thus, by sharing assets throughout the game world, and choosing a minimalistic style, a general look and feel of *Breach* was established, making the game world feel like a single, coherent unit.

Sound effects were taken from external sources and processed to fit the setting of the game and to create an aural coherency.

To increase the enjoyment and immersion the player of *Breach* will experience, gameplay scripts to make the game world more dynamic were implemented. These include flashlight flickering and interaction functionality to *e.g.* pick up objects or press buttons, as well as area specific events such as doors reacting to the player or the monster switching between different animations depending on its location.

Chapter 6

Conclusion and Discussion

This chapter covers the results in this thesis, connecting the achieved results with the problems described in the introduction. It also contains a discussion covering the development team's own thoughts about the project as well as a section dedicated to future directions of research considering immersion-giving design choices that can be utilized in a horror game context.

6.1 Result

Since the goal of this thesis was to develop an immersive horror game experience, a clear, non-ambiguous definition of immersion usable in the context of *Breach* was established. The definition is based on a literature study of previous research, is adapted to fit *Breach*, and is defined as the state of feeling connected with, or within, a virtual environment. Further elaboration is found in Chapter 2 - Immersion.

During the literature study, a number of game design patterns was identified (presented in Chapter 3 - Game Design for Immersion and Enjoyment), which when implemented would provide a heightened sense of immersion. However, the combination of patterns does not always contribute to an overall increase in immersion and care has to be taken when deciding what design choices could benefit a horror game. The covered areas providing an increase in immersion and enjoyment in *Breach* are: *objectives design*, *sound design*, *horror design*, *world design*, and *design for head-mounted displays*. It is thus clear that when considered, the overall experience of a horror game can be enhanced by design choices in these areas, which enable the player to immerse herself in a game world. Especially important are the design decisions made when targeting Head-Mounted Displays (HMDs), since both traditional information presentation and control schemes of the game have to be re-evaluated for fitness in the context of such display technologies.

By evaluating the effectiveness of achieving immersion in *Breach*, a number of particular immersion types were found catering to the horror experience in *Breach*: Emotional Immersion, Cognitive Immersion, and Sensory Immersion. Furthermore, by utilizing the Oculus Rift, a great increase in immersion was noted due to the visual stimuli and the feedback from real-world physical movements of the player. It is thus evident that head-mounted displays augment

the virtual reality experience such that enjoyability and immersion is heightened, leading to the conclusion that head-mounted displays do, in fact, enable a further level of immersion.

In *Breach*, an AI was implemented to provide the main horror element, the monster, with a believable behavior (further discussed in Chapter 4 - Artificial Intelligence). The most important part of the AI in *Breach* was the implementation of navigation. By evaluating the different navigation systems provided by UDK, the Navigation mesh system was chosen due to it providing a less predictable behavior, thus better resembling human ambulation, resulting in a more believable AI implementation.

In addition to navigation, the behavioral pattern used by the AI in *Breach* further improved the perceived believability and the challenge provided to the player. In particular, providing the AI with its own goals (by patrolling) and simulated senses to detect the player are both prime examples of features which greatly augments the AI behavior in such a way that the AI is perceived as more real.

Since *Breach* features a futuristic theme designed specifically for the game, a majority of the utilized content had to be custom-made with a big emphasis on a consistent look and feel, and with all content providing a homogeneous atmosphere during the horror game experience. The created content consists of meshes, levels, sounds, textures, and scripts. Further information can be found under Chapter 5 - Content.

6.2 Discussion

Creating *Breach* was not an easy task, and required the development team to learn a great deal on the road to the finished product.

This thesis divides game immersion into four types of immersion: Emotional, Cognitive, Spatial, and Sensory immersion. These immersion types were decided upon based on a literature study. However, there are many studies that cover immersion, and other studies may have different findings and may draw different conclusions. This thesis required a clear definition on where to draw the line on the subject of immersion, which is why only two studies had their models presented and combined. This could be argued to be flawed when other studies, with different outcomes, have been performed on these areas. Based on our literature study, however, we found that most existing definitions of immersion cover the same areas, but using different terminology. Thus we deemed it was enough to use the combined terminology of two of them, creating a model covering the important immersion aspects of *Breach*.

Unreal Development Kit provided an engine and a platform to base the game upon, saving a lot of time by providing tools in almost all areas needed to create a game. However, much work had to be put into the actual creation of the game.

Making all models for the game was hard work and took time, especially since none of the collaborators had prior experience with modeling. Creating textures was also very time-consuming due to the decision to use as few UDK-supplied textures as possible which resulted in many textures having to be created.

Not having to spend time and resources on developing an engine made it possible to put a much larger focus on the content of the game and actually have a completed product by the end of the project, although further improvements

(such as better communication of objectives and a greater quantity of assets) could still be made.

UDK offers two ways to create a new project. The first one offers the user to build their game upon an existing game, Unreal Tournament. With this option developers can modify Unreal Tournament to create their game. The second option is to create an empty project. With this option the user only gets classes provided by the engine. This was the option chosen to create *Breach*. Choosing this option provided some advantages, such as a better understanding of how the interior of the game logic works. However, there were also disadvantages; almost all online help available regarding specific issues provided solutions which required the project to be built upon Unreal Tournament. Since *Breach* does not build upon Unreal Tournament, the solutions to most of our issues could not be found on Internet forums, requiring us to spend more time researching and testing in order to solve the issues at hand.

An issue encountered in this thesis was trying to evaluate the achieved level of immersion and enjoyment in *Breach*. Since both aspects are subjective, quantifying and measuring them objectively is difficult, which led to exclusive use of the project members' subjective opinions when designing and evaluating the game. Due to this, the sample of opinions was small; thus the result may not be indicative of a general trend in any population other than the project team. However, since the development team have a lot of video game experience and since the primary decisions made are backed by previous work, the decisions made ought to be applicable even outside of this sample group.

Initially, the expected result when evaluating control methods for HMDs was that a free camera rotation not tied to the player avatar direction (as opposed to having the player walk in the direction she is looking) would provide the most immersion. By this method, the player would be able to, for instance, run away from the monster while looking back at it. Unexpectedly however, this control scheme turned out to be difficult and unintuitive, with the particular reason presumably being a lack of a orientation reference, such as the body of the player's avatar. However, while a certain control scheme was preferable in *Breach*, it may not be the best control scheme in all HMD situations, but may be advantageous in other games lacking a point of reference indicating what direction in relation the movement direction the view of the player is looking.

6.3 Future Directions

Although immersion-providing design decisions are presented in this thesis, the underlying data set is rather small. Therefore, a more comprehensive study should be made on the proposed designs by gathering data from test participants playing scenarios with different settings (*e.g.* different control schemes and objectives presentation). This would provide more concrete evidence for the immersion and enjoyment providing aspects of the designs laid out in this report.

An additional recommendation on future research is to add another control method, for instance a motion- and orientation-sensing game controller, for the navigation in a horror game, evaluating if such hardware is a contribution to the immersion of the game and if it is desirable to utilize it along with an HMD.

Bibliography

Adams, E. and Rollings, A. (2006). *Fundamentals of Game Design*. Pearson Education, Inc., New Jersey.

Andersen, E., Liu, Y.-E., Snider, R., Szeto, R., Cooper, S., and Popović, Z. (2011). On the harmfulness of secondary game objectives. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, FDG '11, pages 30–37, New York, NY, USA. ACM.

Arkane Studios (2012). Dishonored. [DVD, Online, Blue-ray].

Autodesk Media and Entertainment (2009). Autodesk 3ds Max. [Software].

Avalanche Studios (2010). Just Cause 2. [DVD, Online, Blue-ray].

Beyond Unreal (2008). Stop latent execution. http://wiki.beyondunreal.com/UE3:Controller_%28UT3%29#StopLatentExecution. Retrieved 2013-04-19.

Bioware (2007). Mass Effect. [DVD, Blu-Ray, Online].

Björk, S. and Holopainen, J. (2004). *Patterns in Game Design*. Charles River Media, INC., Hingham, Massachusetts.

Blender Foundation (1995). Blender. [Software].

Blumstein, D. T., Davitian, R., and Kaye, P. D. (2010). Do film soundtracks contain nonlinear analogues to influence emotion? *Biology Letters*, 6:751–754.

Brown, E. and Cairns, P. (2004). A grounded investigation of game immersion. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, pages 1297–1300, New York, NY, USA. ACM.

Buckland, M. (2005). *Programming Game AI By Example*. Wordware game developer's library. WORDWARE Publishing Company.

Capcom (1987). Street Fighter. [Arcade, various].

Cordone, R. (2011). *Unreal Development Kit Game Programming with UnrealScript*. Packt Publishing Ltd., Birmingham, UK.

Demyen, D. J. (2007). Efficient triangulation-based pathfinding. Department of Computing Science, University of Alberta.

DICE (2011). Battlefield 3. [DVD, Blu-Ray, Online].

DICE and EASY (2009). Battlefield Heroes. [Online].

EA Redwood Shores (2008). Dead space. [DVD, Online, Blue-ray].

Ekman, I. (2005). Meaningful noise: Understanding sound effects in computer games. In *Digital Arts and Cultures 2005*, Copenhagen, Denmark.

Epic Games (2012a). Ai & navigation. http://udn.epicgames.com/Three/AIAndNavigationHome.html. Retrieved 2013-04-15.

Epic Games (2012b). Ai overview. http://udn.epicgames.com/Three/AIOverview.html. Retrieved 2013-04-12.

Epic Games (2012c). Getting started: programming. http://udn.epicgames.com/Three/GettingStartedProgramming.html. Retrieved 2013-05-14.

Epic Games (2012d). Latent functions. http://udn.epicgames.com/Three/MasteringUnrealScriptStates.html #LATENT FUNCTIONS. Retrived 2013-04-19.

Epic Games (2012e). Navigation mesh reference. http://udn.epicgames.com/Three/NavigationMeshReference.html. Retrieved 2013-04-19.

Epic Games (2012f). Using waypoints. http://udn.epicgames.com/Three/UsingWaypoints.html. Retrieved 2013-04-15.

Epic Games (2013a). Udk. http://www.unrealengine.com/en/udk/. Retrieved 2013-03-01.

Epic Games (2013b). Udk licensing. http://www.unrealengine.com/en/udk/licensing/. Retrieved 2013-03-01.

Epic Games (2013c). Udk licensing. http://udn.epicgames.com/Three/ DevelopmentKitGettingStarted/. Retrieved 2013-04-04.

Ermi, L. and Mäyrä, F. (2005). Fundamental components of the gameplay experience: Analysing immersion. In *In DIGRA. DIGRA*.

Findeli, A. (1995). Moholy-nagy's design pedagogy in chicago, 1937-46. In Margolin, V. and Buchanan, R., editors, *The Idea of Design, A Design Issues Reader*, page 29. The MIT Press.

Fitch, W., Neubauer, J., and Herzel, H. (2002). Calls out of chaos: the adaptive significance of nonlinear phenomena in mammalian vocal production. *Animal Behaviour*, 63(3):407 – 418.

Frictional Games (2010). Amnesia: the dark descent. [DVD, Online].

Gamasutra (2010). Report: Wii Holds 47 Percent Of Global Console Revenue. http://www.gamasutra.com/view/news/27932/ Report_Wii_Holds_47_Percent_Of_Global_Console_Revenue.php. Retrieved 2013-05-08.

Grimshaw, M. (2010). Player relationships as mediated through sound in immersive multi-player computer games/relaciones mediadas por el sonido entre jugadores en el entorno de juegos multijugador. *Comunicar*, 17(34):73–80.

Grimshaw, M., Lindley, C., and Nacke, L. (2008). Sound and immersion in the first-person shooter: Mixed measurement of the player's sonic experience. In *Audio Mostly: a conference on interaction with sound*.

Harmonix Music Systems (2005). Guitar Hero. [DVD].

Huiberts, S. (2010). *CAPTIVATING SOUND: THE ROLE OF AUDIO FOR IMMERSION IN COMPUTER GAMES*. PhD thesis, Utrecht School of the Arts (HKU), The Netherlands: Utrecht.

Huiberts, S. and van Tol, R. (2008). Ieza: A framework for game audio. http://www.gamasutra.com/view/feature/131915/ieza_a_framework_for_game_audio.php. Retrieved May 1, 2013.

Jennett, C., Cox, A. L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T., and Walton, A. (2008). Measuring and defining the experience of immersion in games. *Int. J. Hum.-Comput. Stud.*, 66(9):641–661.

Jørgensen, K. (2007). On transdiegetic sounds in computer games. *Northern Lights: Film and Media Studies Yearbook*, 5(1):105–117.

Jørgensen, K. (2011). Time for new terminology?: Diegetic and non-diegetic sounds in computer games revisited. *Game Sound Technology and Player Interaction: Concepts and Developments*, pages 78–97.

Jørgensen, K. (2012). Between the game system and the fictional world: a study of computer game interfaces. *Games and Culture: A Journal of Interactive Media*, pages 1–22.

Lazzaro, N. (2004). Why we play games: Four keys to more emotion without story. http://www.xeodesign.com/whyweplaygames/xeodesign_whyweplaygames.pdf. Retrieved 19 May 2013.

Millington, I. and Funge, J. (2009). *Artificial Intelligence for games (second edition)*. Morgan Kaufmann/Elsevier.

Monolith Productions (2005). F.e.a.r. [DVD, Online, Blue-ray].

My Learned Friends (2012). Mass effect. http://www.mylearnedfriends.com/2012/04/mass-effect-3-tribute-to-amazing.html. Retrieved 2013-05-20.

Nintendo EAD (1988). Super Mario Bros 3. [Cartridge].

Oculus VR (2013). Frequently asked questions. http://www.oculusvr.com/faq/. Retrieved 2013-04-19.

Perforce Software (1995). Perforce. [Software].

Perry, T., Truxal, C., and Wallich, P. (1982). Consumer electronics: Video games: The electronic big bang: The video-games industry, the design of game hardware and software, and the psychology of games are examined in this special three-part report. *Spectrum, IEEE*, 19(12):20–21.

Rational Software (2001). Rational unified process: Best practices for software development teams [white paper].

Rouse, R. (2004). *Game design: theory & practice, Second Edition*. Wordware Publishing, Plano, Texas.

Salen, K. and Zimmerman, E. (2003). *Rules of Play: Game Design Fundamentals*. The MIT Press.

Snook, G. (2000). Simplified 3d movement and pathfinding using navigation meshes. In DeLoura, M., editor, *Game Programming Gems*, pages 288–304. Charles River Media.

Sweetser, P. and Wyeth, P. (2005). Gameflow: A model for evaluating player enjoyment in games. *ACM Computers in Entertainment*, 3(3).

van Toll, W., Cook., A. F., and Geraerts, R. (2011). Navigation meshes for realistic multi-layered environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3526–3532.

Visceral Games (2011). Dead Space 2. [Online, DVD, Blu-ray].