Assessing the Validity of Experience-Driven Procedural Generation in Horror Games

Exploring the Impact of Experience Driven Procedural Generation

University of Lincoln
Jez Stuart Edward Horton
School of Computer Science

BSc (Hons) Games Computing

April 2019

UNIVERSITY OF
LINCOLN

Acknowledgments

This project wouldn't have been possible without the consistent support throughout the years and countless sleepless nights accompanied by my friends Harry, Joe, Toby and Connor. I would like to extend a thank you to my partner Annalise, and my family (including all my pets, without whom I would have been a very different person) for their emotional support.

A special thanks to Kevin Jacques for enabling me to reach this point and Mark Doughty my project supervisor for helping in the lead up to hand in.

Abstract

This project presents a summary of users' feedback, based on the interpretation of their own actions. It analyses which design of gameplay in video games the users' find the most compelling. That being the crafted aspect of a carefully designed level or the procedurally generated level tailored to them to create their own experience. This project aims to explore the idea of procedurally generated content being used in place of designed methods, addressing and relieving some of the common problems found in the games industry in general.

This study also looks at the enjoyment the player felt in comparison to the generated levels or the hand-crafted levels. Showing whether a procedurally generated level even has a valid position in a successful horror game.

The chapters within the study will not only explore some of the research already conducted within the games industry and psychology but will facilitate any future research in procedural generation application in horror games and provide valuable player feedback based on a comparative artefact.

Contents

1.0 Introduction

1.1 Brief

Procedural Content Generation (PCG) in video games refers to the creation of game content that is automatically generated using algorithms, either simplistic or complex. However, "it would be futile to hope to come up with a definition of procedural content generation in games that everybody agrees on." (Togelius, et al., 2011). Some of the early iterations of PCG in video games were in the early 1980's, a time when the computational performance and limited storage meant PCG was paramount for roguelike style games such as *Beneath Apple Manor (Don Worth, 1978)*. PCG is increasingly used in modern games as "procedural content generation techniques can help to reduce the costs associated with content creation" (Carli, et al., 2011).

One of the other major advantages of PCG is diversity, this engages the player more thoroughly. You can diversify the level itself as seen in games such as *Rogue (A.I Design, 198)0* or even the collectables as seen in the *Borderlands (Gearbox Software)* series. Without the massive amount of replayability in the level generation in *Rogue* or the almost infinite weapon generation algorithms in *Borderlands series* the content would have become stale for the players long ago.

While there has been a lot of research into the varying and most effective methods of PCG in video games and we can clearly see the effect of a well implemented PCG element on the replayability of a game. One of the areas PCG has almost left behind is the horror-genre, the focus of this investigation will be to determine if PCG has a valid element in horror-based games.

1.2 Main Aims and Objectives

The main aim of the project is to determine whether PCG is a valid component in a successful horror game. PCG elements are often used in a variety of games and different implementations. They're used for a range of reasons, from small team sizes and limited development time to massively expansive content and keeping the content fresh.

Examples of this are clearly seen in games such as *Moon Hunter (Kitfox Games, 2016)* where the development team only consisted of four people – three programmers and a designer. The game developed by Kitfox games became popular for its endless replayability and the intelligent procedural development system used in the level creation. The massively expansive content can be seen in games such as *No Mans Sky (Hello Games, 2016)* who initially had a poor reception, but the procedural generation algorithms still held up to generate an entire star system and potentially billions of different planets.

The model being used to assess the validity is the comparison of a crafted and designed level compared to a level generated by the players reactions in certain scenarios and semi-scripted events. To do this, three separate objectives need to be outlined, these are outlined below;

- Creating a subjective 'scary' level, which would be the initial test basis. This level would be crafted by a developer with level design thoughtfully implemented along with lighting, sound design and monster design.

Attempting to make use of the key features of a horror game outlined in section 2.1.2. This level would have two uses; (1) a basis for a measurement and (2) recording the players reactions during certain events to generate the next level to tailor the players experiences specifically to them.

- Creating a procedurally generated level, which would include varying the monsters and sound design to adhere to the measurements set out by the player in the original level.
- Assessing the validity of the Procedural Level by giving the players a questionnaire to fill out afterwards. If any further conversations are taken in response, the summary points shall be put in transcript.

[note: Included in the appendix is a set of acronyms commonly used throughout the study, they are also available immediately after the first full use of the term]

2.0 Background and Literature Review

This literature review is a comprehensive analysis of all relevant factors that contributed towards developing a PCG horror game. It analyses what exactly is a horror game and why we enjoy it. It looks at it from a psychological standpoint which will aid into the creation of a successful artefact that could meet the aim of the project. The literature being reviewed will be academic papers addressing different aspects of the aim. There is no single literature that explores the same aim of the project, meaning the literature will be pulling from the different aspects such as "Event Driven PCG" and "PCG-Based Game Design".

2.1 What is Horror?

To look at what generates a successful horror game the concept needs to be addressed first. Horror is defined as "an intense feeling of fear, shock, or disgust" as stated in the Oxford Dictionary. Creating a game of that definition almost seems sadistic, however, horror as a genre has always been prevalent even in literature as from the likes of *H.P. Lovecraft* whose iconic 'monsters' and stories are still being told today in different mediums, with recently *Call of Cthulhu (Cyanide, 2018)* using the works of H.P. Lovecraft.

Horror monsters for example can be borrowed from realism, things that *could* be possible but shouldn't be. You can see this in generic tropes such as the living dead and vampires. Anything too far from reality however becomes distorted, and 'silly'. (Freeland, 2004).

2.1.1 Why do we enjoy horror?

The reason we as consumers like the horror-genre is almost undefined, there are many circulating theories, one of which is the idea of the 'stimulation seeking' (Jarrett, 2011). This being the idea that the individual is seeking a sense of relief felt once the initial horror is passed.

Carl Jung who was a phycologist had a theory of the 'shadow', which is one of the most influential theories. "Everyone carries a shadow, and the less it is embodied in the individual's conscious life., the blacker and denser it is." (Jung, 1912). This shadow he refers too, is not a literal shadow, but the whole of your personality and all of the

parts of yourself that you don't want to admit to having. This could be one of the reasons we like horror, it satiates the desire of our 'shadow' selves. (Jarrett, 2011)It can also be suggested that we can explore our shadows and give them form as we explore in a safe way.

One of the intriguing aspects is whether insights from psychology could help guide the designers/writers of a horror game. History suggest this being the case with the likes of Arthur Conan Doyle and H.P. Lovecraft, both influential writers with a medical background.

2.1.2 Horror Games

There are a lot of different takes on the horror aspect of video games, which highlight many different approaches and ideas to this genre. However, there are four key principles that have been identified which almost all horror games take into account. (Bonilla, 2017)

- Disempowerment of the player

A large part of the appeal of video games is the empowerment of the player. This can be anything, from a powerful ability (see figure 1) or an elevated status where 'only you can save the world'. One of the key features of a horror game is stripping the power the player would usually have, even in a real-life scenario, away. (Horti, 2018)

Figure 1: Infamous 2 showcasing the super human powers (Infamous 2)

This can be attained in two ways, the first being the removal of any combat mechanics, needless to say it is horrifying to face an apparent threat with no options other than to run away and hide. *Amnesia: The Dark Descent (Frictional Games, 2010)* makes use of this method exceptionally well. Creating a good atmosphere with the only interactable objects being the lamp (see figure 2) the player is carrying and hiding spots away from the enemies' vision. This is complete disempowerment and the easiest version of disempowering the player.

The second method is to provide the player with a sense of hope and periodically strip that hope away. A good representation of this is the *Resident Evil series (Capcom),* where they give the player a weapon and means of hope, but ammo is hard to come by, so it must be used sparingly and often has almost no way of defeating more than one enemy.

Figure 2: Amnesia: The Dark Descent grunt source: https://www.pcgamesn.com/wp-content/uploads/2018/10/amnesia-580x334.jpg

Figure 3: Resident Evil 2 Remastered source: https://i.kinja-img.com/gawker-media/image/upload/s--IEKFBfb5--/c_scale,f_auto,fl_progressive,q_80,w_800/devlgonhegfqixs76fzs.png

- Isolation

It's in the human nature to seek community, guidance and help from others. When we're isolated it's completely natural to feel scared. This is where horror games capitalise by putting players in situations alone. Requiring players to explore a mental hospital or typical horror scenario would automatically make the player feel uneasy. Players are further isolated by hearing certain sounds such as scratching or whining. (Jarrett, 2011)

- Tension

Creating tension is paramount for a horror game to be successful. This can be achieved in a number of ways, however one of the most common methods is resource management as seen in *Resident Evil 2 (Capcom, 1998),* where the player would have a limited inventory size and have to either prioritize ammo or health, which induces a sense of dread if one is expended too quickly. (Krzywinska, 2002)

One of the other huge factors in creating tension in a horror game is the lighting and sound design. This is something borrowed mainly from horror cinematography, "creating dynamic imagery is not just a matter of adding some light and pointing a camera" (De Leeuw, 1997). A huge factor of creating tension is the sound design and lighting in a given environment, however as stated by De Leeuw it's not an easy-going matter to recreate this in a procedural style of generation as this could diminish the affects.

- Uncertainty and Unpredictability

A huge point of creating the perfect atmosphere for the player to get scared is the uncertainty of a situation. To make them afraid of the dark, not the darkness itself but what could be lurking inside. This attempts to turn the players mind against themselves. A common theme in all successful horror games is this sense of uncertainty the player experiences as they never know what is going to happen next. The Silent Hill (Konami, 1999 - 2004) series captures this essence well, as the enemies are never explicitly explained but given enough of a story to keep the player uneasy and scared of their presence (see figure 4). (Savo, 2014)

Figure 4: Pyramid Head, an iconic character in Silent Hill source: https://upload.wikimedia.org/wikipedia/en/thumb/4/4c/Pyramid_Head.jpg/250px-Pyramid_Head.jpg

This sense of uncertainty is one of the key parts of anything in the horror genre, and it is this key part that lends itself to going stale with a bad horror game. Exploring the effects of the unpredictability within the PCG level is paramount to understanding my original thesis.

To summarise, a horror game has a mixture of different elements but almost all of them have to be performed well simultaneously in order to provide a convincingly scary experience to the user. Many different forms of scares exist, this could be a jump scare or just general feeling of uneasy. Tension is needed to create all of these and to create tension the player needs to be disempowered, isolated and unable to predict what comes next.

Building from this research, two types of horror games can be identified; narrative driven, and rudimentary horror. Defining these as:

"*Narrative-driven horror games are based on the compelling idea of finding out what happens next in a given story.*"

"*Rudimentary horror is the idea of being scared for the sake of being scared, for feeling the relief after the initial scare has passed.*"

Some examples of narrative driven are; *Silent Hill (Konami Computer Entertainment Tokya, 2001*), *Resident Evil (Capcom), Call of Cthulhu (Cyanide, 2018)* and *Outlast (Red Barrels, 2013)*. All these games keep the player engaged by providing them with a compelling narrative.

Some of the examples of rudimentary horror; *Slender: The Eight Pages (Parsec Productions, 2012, The Corridor (AI-exe, 2012)* and *Penumbra: Black Plague (Frictional Games, 2008).* Almost all of these games contain little to no narrative

Despite doing relatively well on the market and having relative success the majority of these titles almost never appear online in lists of "Top 20 Horror Games" (Dingman,

2018), I believe this is due to the stale nature of the gameplay as narrative driven games give the player an opportunity to keep playing and experience new things in terms of narrative, but these rudimentary games do not provide this experience, as instead they become very repetitive.

2.2 Existing PCG Horror

At the time of writing *SCP Containment Breach (Joonas Rikkonen, 2012)* is one of the only horror games using PCG in the market, and is a perfect example of having limited replayability despite including PCG. The development team was originally only one person however this was opened up to open source later on. The scare factor became stale very quickly, as it uses a topological approach to generate the content. This could be down to the sound design or the simplistic design of the levels. The game-play area is created from a random selection of pre-existing rooms and prefabs [see appendix for a detail explanation of prefab], this project and study has taken a similar approach to this however with varied enemies and sound design to try and alleviate the stale factor of the level design. The initial pre-made rooms in this project have a lot more content and atmosphere, this could be a factor to better adhere to the points in 2.1.2 compared to *SCP Containment Breach*.

2.3 Event Driven Procedural Generation

Looking into the different methods of Event-Driven Procedural Generation is key to the success of the project and there are many very diverse ways to apply this.

"Proposing Experience-Driven Procedural Content Generation (EDPCG) as a generic and effective approach for the optimization of user (player) experience." (Yannakakis & Togelius, 2011)

By using this specific paper, identifying what type of EDPCG would be the most applicable for this project would be easier as the paper provides explanations and evidence for all the different methods. Outlining the different approaches such as "Subjective Player Experience Modelling" (PEM). Using this particular model, the developer would ask the player about their experience and build a model based on that data. This can be based on "free-response during play or on forced data retrieved through questionnaires". If you force the players to report their findings it constrains them into specific questionnaire items, the answers would then be subjective to what you have included. Self-reports can be intrusive if injected during gameplay and thus breaks immersion, in turn breaking the tension created in terms of horror game development.

Another type of EDPCG that could be beneficial is "Objective PEM" this is the measure of body alterations in the player that may assist in recognizing the emotional responses of a player. While the procedurally generated level and the original level could be assessed in a much higher quality, as having access to heart-rate and facial features could allow the procedural algorithm to be better tailored to each individual player; it wouldn't necessarily correlate to the enjoyment perceived in the particular level. It would also drastically increase the testing time for the given player, as only one 'experience' could be happening at a single time and would always require access to the specific equipment needed to record the physiology. The last factor in deciding

against this approach is that this approach would require optimal testing conditions, and this would result in a lower player-base to test the artefact.

The last type of EDPCG explored within this literature is the "Gameplay-Based PEM", which would include parameters that are the players behaviour in relation to the system elements (i.e. non-player characters such as enemies or even the sound in a specific environment).

While in a general perspective this paper covers the different types of EDPCG and gives a rough outline of their capabilities, there isn't too much reference to the actual use of the different models in different applications. For example, if that had said the main benefits or negatives for any of the different models it would mean the reader wouldn't have to infer from the context of their own applications.

In this project, opting to adopt a "Gameplay-Based PEM" appears the most beneficial as it wouldn't break the immersion for the character thus relieving the tension they may be experiencing. It would be able to record data while the player is experiencing the test level, which would provide a seamless interaction between the hand-crafted level and the PCG level. This would provide a better basis for the player to compare the two at the end of the experiment as they should be 'fresh' in the participants memory, in turn improving the accuracy and reliability of the results.

2.4 Level Generation

"Game levels are frequently capable of, and indeed designed to, elicit affective responses" (Togelius & Yannakakis, 2016). This literature explores the correlation between emotions and level generation. Togelius & Yannakakis created their own EDPCG framework, adjusting the players experience in real-time. The basis of this first starts off assessing the quality of the level previously generated based on a set of predefined parameters, then generates a level that optimises the experience for the user.

Emotion-driven PCG takes core aspects into consideration such as functionality and aesthetics, it looks at the "generation of game levels and their impact on gameplay and experience".

The literature in question reviewed different aspects of EDPCG but barely scratched the surface of how it would affect the players enjoyment of said game. It concluded that continued player monitoring during different events might infer to the game that the player is bored with the current selection of events and do an "educated guess" for some different opportunities.

While thought provoking, this paper doesn't provide any basis or fundamentals for what the authors are discussing. Instead providing a basis for the hopes of future development in the EDPCG methods. This paper did highlight the importance of Sonancia, which is a system built for horror games where "level generation and soundscape generation are orchestrated by notions of tensions and suspense; the level generator attempts to match a designer specified progression of tension while the sound generator attempts to prompt the player's suspense in rooms where tension is low" (Lopes & Yannakakis, 2015).

Deciding to use this idea of Sonancia in the PCG level, as while it took longer to implement than the standard method of varying sounds in the level. However due to the research of the above papers it was deemed far more affective in provoking a high-tension atmosphere. This should in-turn lend itself to creating a 'scarier' game and providing a better analysis against the EDPCG versus the hand-crafted model.

2.5 Styles of Generation

There are many different styles of generation, within the following segment the different methods will be explored and the merits of each of them placed against a horror-game style of gameplay.

Terrain Generation is a key method of PCG, due to the recent advances in processing power of the average home computer. It's been made possible to "simulate erosion processes near real-time" (Olsen, 2004). One of the common methods of terrain generation is "Perlin noise" then being placed over a mesh. One of the issues of 1/*f* generation is the inherent isotropic[appendix] nature of the terrain [see appendix for information on this particular noise generation].

Dungeon Generation is another method of PCG, akin with the terrain generation this can be achieved in a variety of different ways. This depends a lot on the style of dungeon you want to generate. If the aesthetic is more of a 'cavern' [see appendix for a clarification on cavern] than a dungeon [see appendix], one of the methods of procedurally generating this is to use "Cellular Automata"; this self-organizing structure consists of a set cells that make up it's neighbourhood. A set of rules is applied to each of these cells, then after multiple generations, patterns start forming in the grid which are all dependent on the ruleset. A good example of this is in figure 5 and 6 (Johnson, et al., 2010).[see appendix for neighbourhood generation explained]

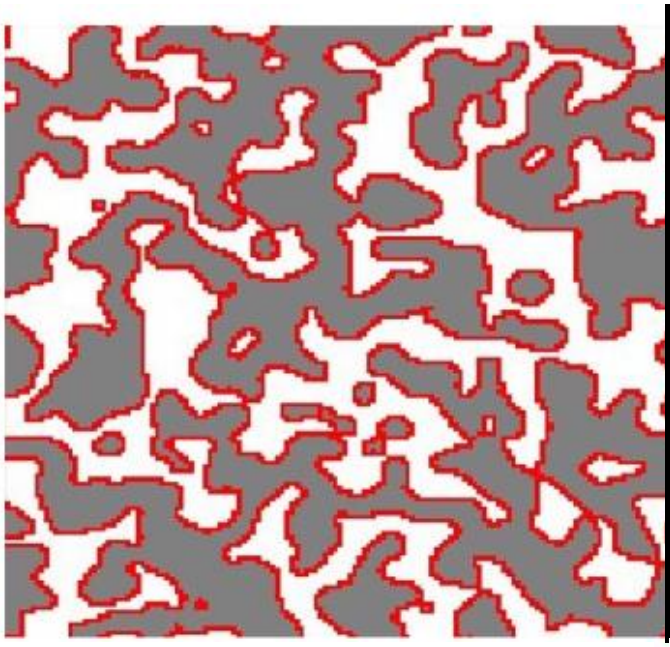


Figure 5: Map generated with cellular automata, the grey areas represent floor, red areas represent walls, and white areas represent rock (Roland van der Linden, March 2014)

Figure 6: Map generation in the cellular automata, in this example there's a different ruleset allowing the creation of corridors and symmetrical rooms. The black represents the rock and the white represents the accessible pathways (Games, n.d.)

One of the other systems to generate a dungeon is the "Generative Grammars", these allow graph and shape generation although originally applied to a set of linguistic phrases. Graph grammars have been previously used by Adams (Adams, 2002), throughout their paper you can clearly see our definition of a dungeon is applicable due to their exclusive use of the term "dungeon levels". Rules of a graph grammar are used to generate topological descriptions of levels, in the form of a graph. Nodes represent rooms, edges and its adjacencies. However, due to the nature of this a few features are left out of this generation method, for example the room size. Graph generation can be controlled through the use of global input parameters. Generating levels to match these input parameters is easily achieved by implementing a search algorithm, which would analyse all results of a production rule at any given moment and selects the most optimal one.

Opting to take an approach similar to the graph grammar with the topological approach. Providing a set of predefined requirements for the level for example: 1) It has to have a 'start room' and 2) It has to have an 'end room'.

2.6 Believability of human development

The inherent goal of a lot of PCG is to mimic a human author as well as possible. In this respect there has been a lot of research into creating believable results from small elements such as rocks (Dart, et al., 2011) to levels and environments (Smelik, et al., 2010). With very few exceptions all of these are systems which are procedurally used in games to replace or augment human authored content as seamlessly as possible.

Smith, et al, wrote a journal article that talks about a how a game with PCG elements that can be deployed "as an on-demand game designer". This article contained information about different ways to manipulate information gained from the player to create an entirely unique experience to each player. This is an interesting take on the PCG elements as everything is almost entirely PCG driven, Smith et al, designed a 2D platforming game as a basis of the research. This is a good basis to develop a horror game off, if the play has a completely tailored experience each time they play it means the game could almost never lose the 'scare' factor. However, due to the nature of PCG it would be limited to what the developers have even implemented. It would raise a number of design challenges such as developing enemies that are entirely PCG driven while still being realistic enough to retain the tension and unpredictability explored in section 2.1.2.

The paper looks at a key feature, PCG-Based Game Design. Where most games that have PCG in use, it is to improve aspects of the game such as; improving replayability, as seen in *Rogue*, to resolve technical limitations and, to provide players with an environment so vast it could not be created by human designers as seen in *No Mans Sky*. PCG-Based Game design is to be defined as:

"A PCG-based game is one in which the underlying PCG system is so inextricably tied to the mechanics of the game, and has so greatly influenced the aesthetics of the game, that the dynamics, player strategies and emergent behaviour revolve around it" (Smith, et al., 2012)

This PCG-Based game design seems like something a valid horror game could be employed in using but further research will need to be completed.

2.7 Why we use PCG

As stated by Hendrikx, et al., PCG could be the solution in solving constraints such as "scalability challenges due to the exponential growth" (Hendrikx, et al., 2013). They also state that "Game content is an important factor in keeping players engaged in gaming worlds" and while this report doesn't reference horror games but looks more at games such as *World of Warcraft (Blizzard Entertainment, 2004)* and the use of PCG to keep delivering the players content to keep them interested. This focuses more on the cost of development and while this is an important factor to keep in mind during game development, this is not applicable as the aim of this project is to see if PCG has validity in horror games. The study by Hendrikx, et al., makes very clear justification for using PCG within a normal games company due to the lower costs for the production and without "putting a large burden on the game content designers". The study goes into large detail about what items within a game can be procedurally generated from vegetation to sound, however just because it takes pressure of the designers doesn't necessarily mean it makes the game better just because it has more content.

Hendikx, et al,. suggested one of the next areas of research following their paper should be based on the realism trade-off for realism control and realism-performance trade-off. This in essence is what makes the proposed project so important, this paper provides justification for the artefact proposed.

2.8 Agent Movement

(Durfee & Rosenschein, 1994) The agent movement is one of the most important things in any game, as it enables a lot of different mechanics and dynamics (Hunicke, et al., 2004)of the game. This segment of the literature review looks at the agent movement and how to best apply realistic agent behaviour. Applying good agent behaviour is a pillar in any game, especially so with horror games as the scariness of games can almost always be directly correlated to the realism. An unrealistic moving architecture of entities would provide the player with comedy not horror. Durfee and Rosenchein first noted the importance of Multiagent Systems where there is a global shared knowledge of the agents and their environments. This could be utilised in a horror game in a scenario where the player would first turn a corner and run into agent 1, proceeding to run away from the position of agent 1 towards the position of agent 2 (figure 7). In turn increasing the tension generated as they are starting to lose their hope noted in 2.1.2

The main downside to the literature discussing the MAS is the outdated nature of this literature as it's dated from 1994, and new ideas have come along to do with the

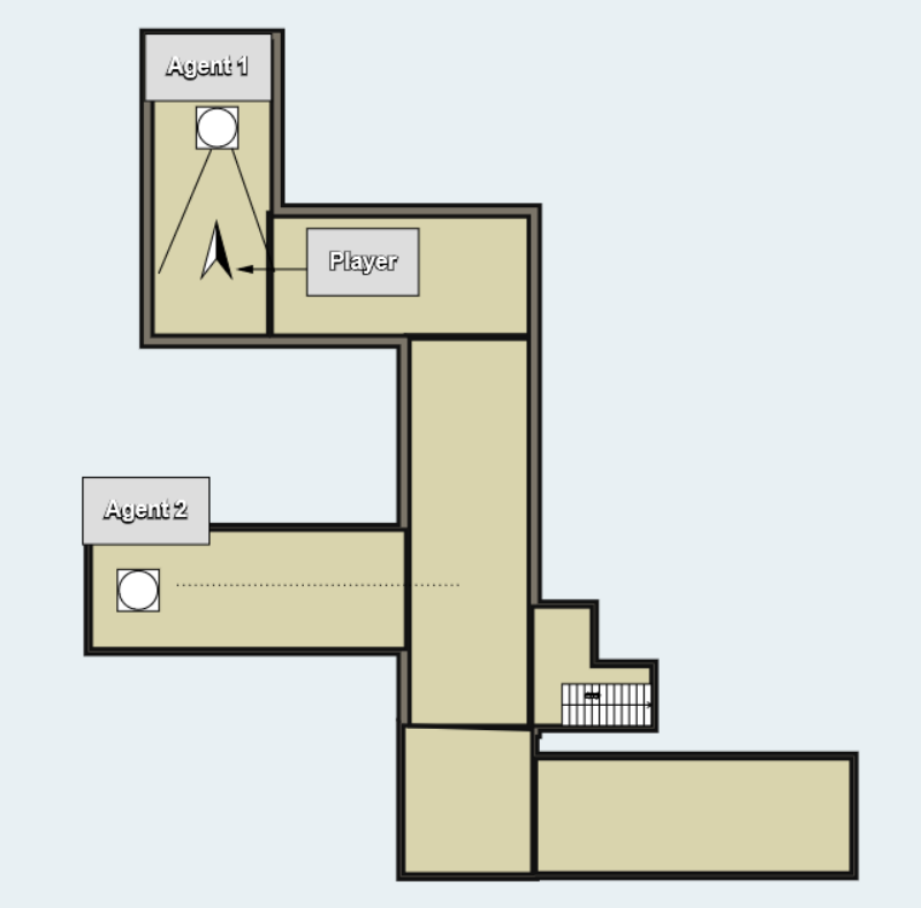


Figure 7:A plan of what the desired path of Agent 2 with shared information from Agent 1. The dotted line represents the route taken from Agent 2 after the player represented by the arrow enters the cone of sight near Agent 1

generation of realistic AI. Despite the date of it, it remains a fundamental design in the video games industry.

One of the more modern papers focusing on real-time pathfinding states:

"One of the greatest challenges in the design of realistic Artificial Intelligence (AI) in computer games is agent movement" (Graham, et al., 2004)

There are a lot of benefits to using the neural networking suggesting by Graham, et al, such as improved speed with the generations. The method suggested lends itself to a machine learning algorithm to teach the monsters how to be better and how to navigate more effectively. While this is affective and useful, using this in the final iteration of the artefact proposed seems almost 'over-kill' as the main aim of the project is to test the validity of PCG not to entirely base the model on PCG that would effectively learn from the player. This is an ideal method of AI for a full horror game based on the PCG method if this study finds PCG to be an effective method of content creation for horror games.

Using neural networks for the AI in a horror game would be extensive and work well if dealing with a 'hive mind' entity. However, due to the nature of the project it would be too constricted by time for the artefact.

2.9 Summary of background and literature review

To conclude this section, research was made into what exactly the Horror Genre is and why we enjoy it. This research was more defined into what makes a horror game inherently scary and the key four principles it needs to adhere too, whilst giving examples of games that have done this well. Taking all of these into account when making the artefact would result into a better thought out final result and more affective at testing the aim of the artefact; that being "If PCG has a valid element in the horror genre".

Looking at the different types of horror games enabled the artefact to be produced coherently on the style of horror game afflicted with the repetitive nature of scaring the player. As narrative driven horror games tend to do quite well with a compelling enough storyline. Focusing the scope onto trying to improve the singular type of 'rudimentary horror game' provides a better scenario to test the viability of the different methods of creation.

With PCG being looked at extensively, from the founding methods of PCG the artefact can be carefully refined into having a more appropriate and tailored method of EDPCG. The identified method of EDPCG was Gameplay-Based Player Experience Modelling. This is the idea of generating Experience Driven PCG based on the players experience and reaction to in-game objects such as enemies and sounds. This was chosen due to data being collected non-intrusively, thus not breaking the immersion of the player and providing a better basis for comparison between the two models of horror game (PCG and Hand-Crafted).

Using other methods of generation such as *Sonancia* (Lopes & Yannakakis, 2015) for the sound design would mean a greater tailored experience as stated by Smith, et al, 2012. Adopting the method of dungeon generation called graph grammars due to the ruleset being global variables and being a simple implementation on the version of predefined prefabs to be included within the generation.

After looking into the varying methods of Artificial Intelligence pathfinding, A* a more efficient iteration than the likes of depth first or Dijkstra's algorithm was the one selected to be put forward into the final version of the artefact. This was due to the amount of research done into it, "A* is a generic search algorithm that can be used to find one solution for many problems" (Xiao & Shi, 2011). Making small adjustments

to the algorithm to enable real-time movement of the goal that the original A* method doesn't allow. Using a neural network in the artefact would be beneficial in making the game scarier and adhere to the player more effectively, however this would be time consuming.

After looking at the different types of PCG we can see how it could benefit the horror genre, in particular the identified 'rudimentary horror' style of horror game as it is these games that lack the replayability and amount of time able to be spent on the game without the scares becoming repetitive and predictable breaking the tension explored in section 2.1.2.

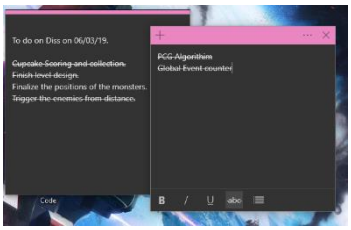
3.0 Methodology

3.1 Project Management

The inherent nature of creating this project was very time consuming and required a lot of data to finalize my thesis proposal. This particular project involved gathering a lot of knowledge based on procedural generation and horror games. Aside from the gathering data aspect having to design, develop and produce an artefact showcasing the testing basis then test said artefact on a range of participants.

To manage this project effectively employing a Waterfall development cycle fits best, the benefits of this is the simple and functional project structure. This project cycle suited the needs as it's for a small project that needed to analyse and the features.

On-top of the Waterfall cycle, using the Kanban methodology had the greatest impact, which meant it was limiting the current work in progress, resulting in a smaller segmented workload.



Kanban 1: Small excerpt from the digitalized Kanban method

3.2 Software Development

One additional feature of the Kanban cycle meant there had to be no external knowledge source of other software, which only furthered the knowledge of software previously used. This methodology employs a visualization technique and helps keep the traction of the product development. Using the variation of keeping notes on a board was helpful as it made meaningful contribution to keeping track of current tasks.

Opting out of using methodologies such as Lean due to the documentation having to be completely precise and generally only being applicable for a group of highly skilled developers.

To keep track of all of the progress within weeks, breaking down each step as stated in the Gannt chart into smaller sections then attaching these to a sticky note stored electronically on a separate monitor is an ideal method as it

provides immediate visual feedback. This would help show the project is gaining traction even without any immediate difference.

Using the waterfall style of methodology was helpful as some of the objectives could only be completed once the other objects were complete. For example, developing the procedural algorithm before designing the method of procedural generation. In producing the artefact, the requirements were very clear and concise meaning it was straightforward to lay out an iterative list to make the development go faster with very set and outlined tasks for specific reasons.

There are several reasons why a more Agile approach like Scrum wasn't taken for this specific project. One of the central reasons for not using this approach was the amount of identifiable reliance's included within the project. For example, the second level in the artefact is generated off of the players experience in the first level. A more straightforward approach for one person to undertake is to go through the order of process (see figure 8).

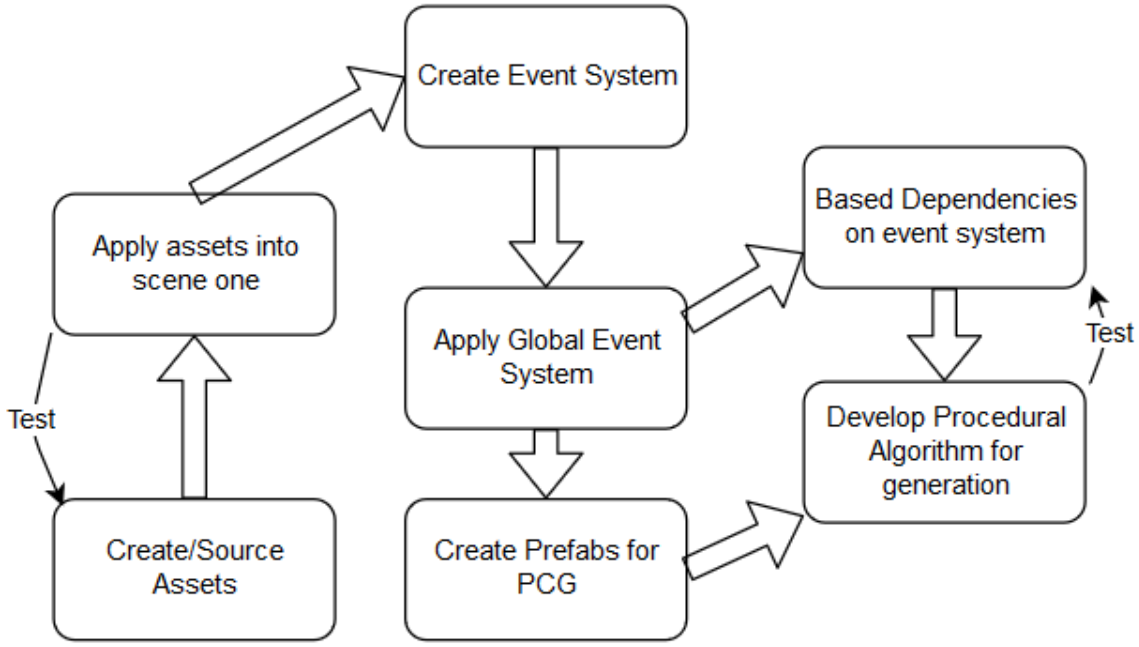


Figure 8: Order of processes in the development. The testing scenarios are on loop until the Scenes are as desired. From this you can clearly see the logical place to start is at the beginning(Creating/Sourcing assets) and work systematically through.

Attempting to determine the faults and then plan out how to alter or incorporate the changes in a "sprint" (Crookshanks, 2015) fashion would take up a significant amount of time that could be more beneficially spent on other parts of the project. Other methodologies such as "V-Shaped" (Ragunath, et al., 2010) were considered due to the similarity of waterfall however due to the nature of the project; no additional

upkeep and maintenance were necessary after the conclusion of the project due to no server maintenance or continued work.

As noted by Ragunath, et al, the spiral lifecycle is good for "large and mission critical project" and "software is produced early in the software life cycle". This wouldn't be suitable for the project as the project by nature needs planning and certainty of the development. This particular method works poorly for small projects as the project repeatedly passes through the four phases; Planning, Risk, Analysis, Engineering and Evaluation. This would also not be suitable as only an evaluation at the end of the project would be suitable due to the type of data to be collected. Its focus is a single question rather than a broad spectrum.

In conclusion taking the default waterfall SDLC (see figure 9) could be a risk as there is no evaluation until the end, if one of the fundamentals of horror game design as set in section 2.1.2 fails then the game wouldn't be scary enough. Making the comparison of the variations of the level generation meaningless. There would need to be a method of testing then developing once again as checking the assets and the fundamentals is paramount to this project. The revised waterfall SDLC (see figure 10) has more iterative qualities rather than the default waterfall model first set out by Royce in 1970 (Royce, 1970). This model would allow the execution stage of the methodology to be revisited in the testing/validation section if features are lacking or need improvement. One of the ideal situations for this would be for when the horror aspects of the game need evaluating/testing to see if they are scary. The testing phase will also be used to test the code and make sure the requirements are still met. Looking at the two models comparatively you can clearly see the sections of the waterfall methodology chosen to put iterations into.

The iterative methodology the project is best suited for reduces the risk of failure and the lessons learned at the end of each iteration can provide positive revisions for the next iteration. It doesn't allow iterations past the release phase, this is ideal as iterations after the release or in this specific project the data being collected from the participants would be void and mean less as the artefact they tested would be altered.

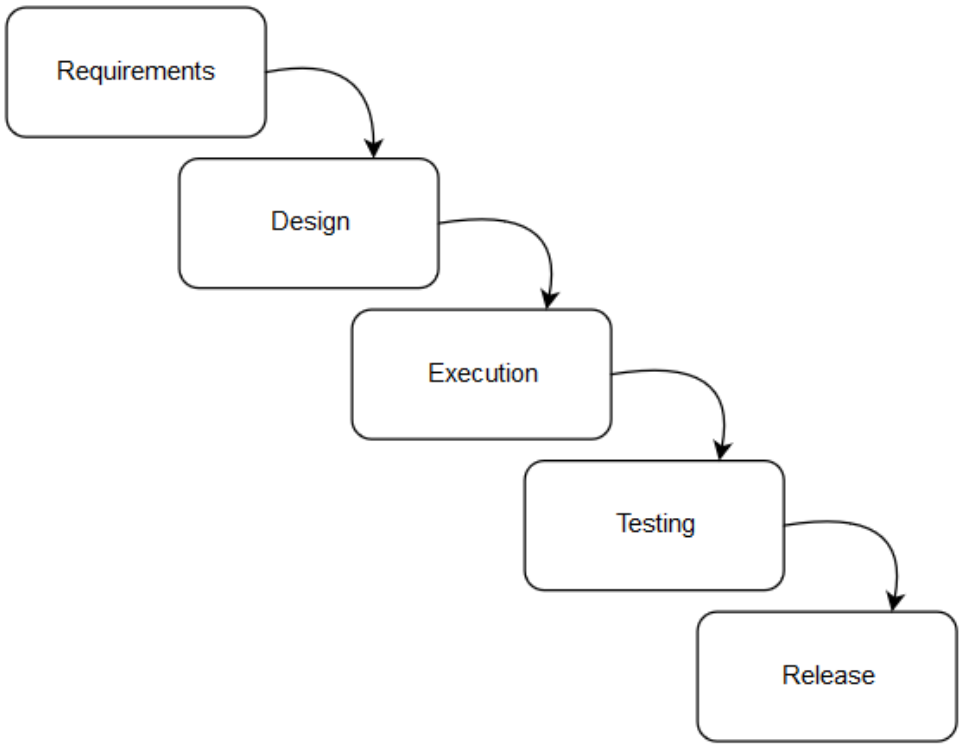


Figure 9: Waterfall model as first noted by Royce

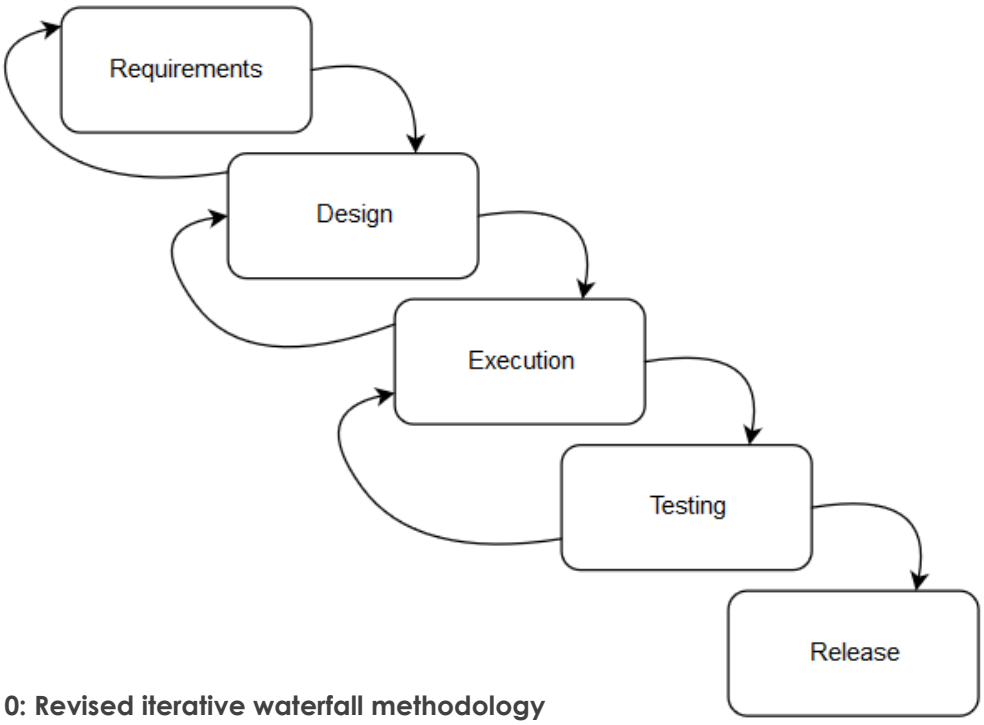


Figure 10: Revised iterative waterfall methodology

3.3 Research Methodology

The research methodology is a huge part of the project, as the research methodology will be used to assess and evaluate if the aims and objectives have been met. There are two types of data that could be gathered; quantitative and qualitative (Jick, 1979). Opting to use more quantitative as it would provide a better basis of understanding if the aim of the project has been met; assessing the validity of PCG in horror games. However, there would be a minimal usage of qualitative after each iteration in the SDLC, assessing if the game and assets provided a sense of unease or scare. This will mainly be used to assess the quality of the assets rather than the finished model. There may be small demos available of the first level to assess how scary the game is or if the assets need to be drastically changed.

This qualitative data is essential to the development cycle as it would provide feedback during the testing phase resulting in further iterations or proceeding to the next step of the development. This qualitative data will be gathered by asking peers their thoughts and interpretations of specific assets, if they suggest an improvement or an alteration to make it better it will be included in the next iteration; then the cycle repeats. Once little or no improvements or alterations can be made, the development cycle will move forward.

The main research method used in this project will be gathering methodological quantitative data, this can be done after the participant has taken part in the artefact experience where they will be presented with a quantitative questionnaire [see appendix]. Testing the artefact is key to the success of the project, as it will assess the main aim of the project. The standard of qualitative data will directly correlate if this project is a success, due to this the most time will be spent gathering this data. It is unquantifiable however, so after the released phase gathering the quantifiable data is important to then analysis the results based on the main aim of the project.

To represent the data in the conclusion, using a mixture of graphs and comparable charts to show which participants preferred the PCG version or the crafted version. The PCG iteration will run twice, giving the participant a chance to get a large enough flavour of the difference PCG makes in level design for horror games.

4.0 Toolset Analysis

In this section a comprehensive list and clarification of choice of all of the tools being considered and chosen for the project and artefact. All of the tools will aid in designing and developing a successful horror iteration comparable to a PCG iteration meeting the Aims set out in section 1.2.

4.1 Middleware and Game Engine

The main aim of this project is to design a horror game, many different middleware's can be considered up to the challenge. Even more alternate libraries/engines can be considered, from *Urho3D (Open Source)* a self-proclaimed "lightweight, cross-platform 2D and 3D game engine implemented in C++" to *OpenGL/SDL (Open*

Source) an engine that has the basis of all other engines. However, both of these are very cumbersome to learn and master as they are bare-bones C++ implementations. A full middleware would be a better option as there are UI based alternatives such as *Unreal Engine (Epic Games, 1998)* and *Unity (Unity Technologies, 2005)*. There are many differences with both of these middleware's, in a comprehensive comparison it was identified that Unity is better for low poly [see appendix for definition] games whereas Unreal Engine is better for graphically intensive games. (Thinkwik, 2018)

Opting to use Unity over Unreal Engine due to the low amount of poly in the game and having prior experience in this engine would be beneficial.

4.1.1 Scripting Language

By default unity now only supports C# and Visual Studio Community. For the purposes of this project, C# was selected for one main reason. It has remained the staple language in Unity through several different iterations, for example (Aleksandr, 2014) since 2017 Unity scraped two other languages due to the domination of C#. Those being Boo and UnityScript. Then again in 2018 (Unity, n.d.), unity scraped MonoDevelop-Unity. C# is widely the most used and because of this support, if there are any issues with aspects of scripting it would be much less time consuming to find a solution. However, if the situation arises where the problem cannot be overcome in C#, Unity supports using two different languages at the same time within the same script.

4.2 Asset Creation and Sourcing

In regards to creating the assets, similar to the middleware there are a lot of very useful tools to be taken into consideration such as *Maya (Autodesk, 1998)* and *Blender (Blender Foundation, 1998)*. Having previous experience with the creation sequence in *Blender* would be beneficial, however there is a focus in *Maya* for the creation of 3D assets for use in *Unity* as the default save file is importable by *Maya* whereas *Blender* must export the image before it can be imported to *Unity*. While both of these excel in terms of creation of 3D objects, the deciding factor was a study that stated "market studies are actually indicated blender as being the best free all-round software" (Alecu, 2010).

One other positive aspect of using *Unity* is the extra support for assets you get as there is a store built into the middleware making sourcing various assets easier. A lot of these assets are free and commercially available (Unity, n.d.).

4.3 Source Control

The source control in terms of the artefact was almost without competition as *GitHub (Microsoft Parent)* is the most widely adopted sources of software artefacts on the internet with over 28 public available million repositories (Kalliamvakou, et al., 2014). *GitHub* offers all of the distributed version control and source code management. It provides access control and is completely open-source.

Due to having easy access to all past submissions of source control, it enables version control effortlessly. Having used *GitHub* before will be beneficial as the general requirements of a repository will be laid out already.

4.4 Documentation/Cloud Storage

For the documentation and Cloud Storage, choosing *OneDrive (Microsoft)* for the storage made the most sense and no other alternatives were considered. This was due to the factor of automatic cloud saving from the *Microsoft Word 2017 (Microsoft, 2017)* desktop application. This would allow simultaneous logins from different computers, allowing transferability between the document versions. OneDrive also has an inbuilt version control software.

5.0 Risk Assessment

Below are all the identifiable risks with the project, their corresponding initial steps to prevent the risk, and the Risk assessment of the risk using a Risk Matrix as shown in figure 11, along with the contingency plan if the initial prevents steps are void. The reason for including a revised Risk Assessment is that the risks have altered and more have been identified since the proposal of the project as the aim of the project has been adjusted slightly.

RISK OUTCOME
Low
Moderate
Significant
High

Likelihood	Consequence				
	Insignificant 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Almost Certain 5	5	10	15	20	25
Likely 4	4	8	12	16	20
Possible 3	3	6	9	12	15
Unlikely 2	2	4	6	8	10
Rare 1	1	2	3	4	5

Figure 11: Risk Matrix; sourced from https://alghandourblog.files.wordpress.com/2015/12/risk-assessment-form.jpg?w=1000

Risk	Consequence of the Risk	Initial steps of prevention	Risk Assessment Severity	Contingency Plan

Data Loss	Loss of large amount of data is always a huge risk in terms of software development. Especially because it has the potential to lose a lot of work, an example of this would be losing the initial level of the game or losing assets.	Cloud Saving is important in the prevention. Microsoft OneDrive saves automatically as mentioned in section 4.4. For the protection of the files of the artefact GitHub will be primarily used as a repository and have a back-up saved to the local hard drive.	15	Revert to the backup of the artefact or version on OneDrive.
Missed Deadline	If the deadline is missed and falls behind on the Gannt chart there is a chance I would have to miss out on implementing some features into the game and thus produce an unsatisfactory artefact.	To prevent this, the Gannt chart will be carefully considered and updated appropriately throughout the project.	6	If deadlines are missed then sacrificing time from another object will be taken into account and then adhere to the Gantt chart more effectively.
Lack of Knowledge in middleware or generally	This could cause missed deadlines as more time will have to be invested into gaining the knowledge required to complete the desired effect.	The project is being created in unity to help alleviate this reason, it is middleware I have previous experience with and using C# as there is plenty of documentation of this.	4	If lack of knowledge becomes a huge issue then extra time will be dedicated to learning from online tutorials and books until there is sufficient knowledge to carry on.
Game isn't scary	This would cause the artefact to fail in the release phase (see section 3.3). Meaning all results gained from the	To alleviate this there will be peer qualitative data being gathered throughout the development process. If the assets aren't scary	15	If this happens, rethinking the asset design and replacing the assets would be paramount. This would put the project on hold

	participants would be meaningless as the main aim of the project is to assess the validity of a horror game using PCG.	enough or produce enough unease in the player the assets will be revised.		until it is completed.

6.0 Gantt Chart

Producing a Gantt chart (see figure 12) allows tasks for the project to be visualised with time constraints. The primary use of Gantt charts is to coordinate activities so projects flow smoothly. They are best used to visual small projects due to the specificity of the tasks at hand. (Wilson, 2003)

Figure 12:Gantt Chart made in Microsoft Excel and updated with Microsoft Excel

7.0 Requirements

The requirements of this project are borrowed from the aims and objectives which are laid out in 1.1. The research into the methods of horror game design and procedural generation assisted in refining the requirements for the artefact. The main aim of the project is to determine if PCG has a valid component in horror games, as a result it is a requirement to create an artefact to demonstrate the different version of content to see what is the most enjoyable to player.

The requirements for the whole of the project are outlined, possible constraints and issues have been identified and described.

7.1 Requirement 1

Perhaps the most important requirement is to make the game scary or at least un-nerving. Without this central requirement the following study would be almost negligible. The aesthetics have been critically evaluated by peers to assess the capability of the scare developed. The four key aspects as set out in section 2.1.2 would need to be taken into careful considering. This will help meet the initial goal and requirements set out for the artefact.

7.2 Requirement 2

The game would have to make use of Experience Driven Procedural Content Generation in the second level, this would need to be used as it would tailor make the second level on the specifications retrieved by the Game-Based E.

7.3 Requirement 3

Varied enemies and agent behaviour will have to be implemented to provide the player with a fresh take on every iteration of the level. Each of these variations will be directly correlated to the experience the users had in the previous level.

7.4 Requirement 4

The final requirement for the game is for the game aesthetic and evoked feelings to be similar enough in each level to be comparable to each-other.

8.0 Game Concept and Overview

From all of the requirements and from taking the literature review into account, a game overview has been created that uses all of the above requirements. The game overview is a guide for the design section to follow.

The game is a dungeon (Johnson, et al., 2010) explorer game with minimal interaction and narrative. Dungeons are a very generic term in games [see appendix for definition] and as a result are very similar to one another. A good example of a dungeon can be seen in figure 13, the game *Dark Souls 3 (FromSoftware, 2016)* features dank, scary looking dungeons. Imitating this design to provide the uneasy feeling from the player.

Figure 13Picture of player in Dark Souls 3 standing in 'Irithyll Dungeon'. Sourced from https://cdn.vox-cdn.com/thumbor/PTmMxIolCOf3jNiz5Sbae92Tf5Q=/148x0:1768x1080/1200x800/filters:focal(148x0:1768x1080)/cdn.vox-cdn.com/uploads/chorus_image/image/49322769/Irit

In order to meet all of the aims set out in section 1.2 two scenes of dungeons will be made, the first will be the hand-crafted level. Taking careful considering of the design aspects set out in section 2.1.2. This is the model that should be inherently scary, as both the level and experience are fresh to the player. This should introduce the different monsters and their corresponding behaviours, as well as the different sounds the player could expect in the game. To provide a game over scenario, collecting a certain number of items then proceeding to the next level would be a good basis. A simple score counter could be achieved in the top corner.

Placing these items near the different types of monsters/events should give the player a chance to experience these events and provide information to the global variables that are used to generate the next level. Once a certain number of items have been collected, it should automatically load into the procedural level. This is a cycle that could continue throughout the levels.

Asking the player to go through at least 3 iterations of the level, the first being the crafted and the following being PCG, would give a good basis of comparison. This should take no longer than 5 minutes. Due to the quality of sound, 'isolating' the player would be important. To do this, using high quality headphones would block out external sounds and provide a better quality of game sounds. Sounds being one of the attributing factors to perceiving threat (Xu, et al., 2005).

In the main menu, to adhere to the ethical guidelines it should be made possible to 'disable' certain enemies and events that could be distressing to a certain player. However, due to the nature of the game being a horror it will be recommended that none of these settings are turned off and the default values for sound is not touched either to properly gauge the full effect of the game.

8.0 Design and Execution

The design specifications map out how the final artefact has been created and what research the design decisions are based on. The design of the artefact meets the requirements and objectives of project. The layout of this section will be similar to the order of process as stated in figure 8.

As the project is focused on researching the application of PCG in horror games, there will not be a huge amount of detail showing how the procedural generation was done. It will have a basic overview rather than an in-depth evaluation. The complex evaluation will be done on the results obtained from the artefact.

8.1 Creating/Sourcing Dungeon Assets

A highlighting feature of horror games is the general aesthetic. It's usually dark, depressing and foreboding of the fears within. Since the project is a procedurally generated dungeon, creating the right textures was very important as they will be similar throughout the level. Creating the assets was relatively simple, first starting with the walls. This was done by creating a single brick in blender, a simple process that didn't require any external knowledge about blender. I had a basic Cube mesh, elongated across the X axis then using the sculpting tool, carve away the corners then apply smoothing to make it look like figure 14. Once one brick is created multiply and add small details into various bricks (see figure 15). This polygon count was still astronomical (figure 16) so using subdivide it made the polygon count lower, this meant it was more feasible to put into a 3D game. Higher polygon counts mean slower loading times and lower frames per second (FPS).

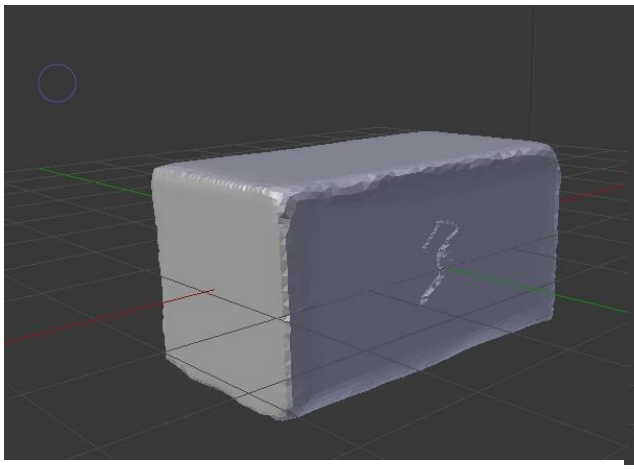


Figure 14: Single Blender Brick

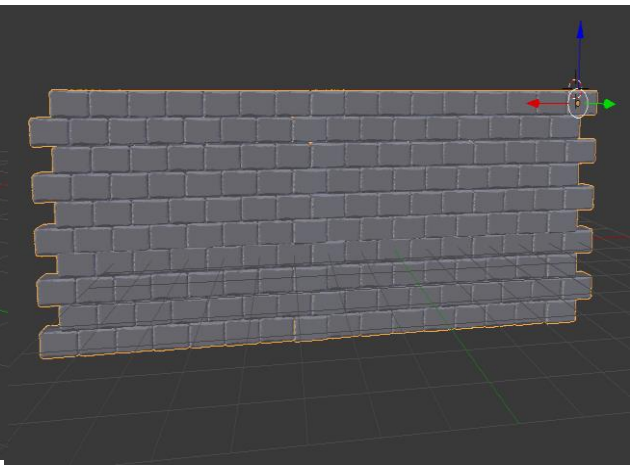


Figure 15: Wall of bricks from figure 14

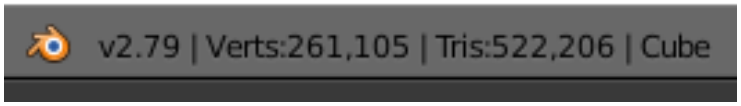


Figure 16: Verts is the polygons or faces

For the floor, this method was working but the created floor had too small of a surface area for the polygon count and any subdivision amount made the floor lose quality drastically.

After looking on the Asset Store, Prodigious Creations had uploaded pre-made dungeon pack called "Decrepit Dungeon" contained floors and various other applicable prefabs complete with meshes. There are a multitude of different assets available on the store for free, from spider designs fully implemented with animations to entire creature packs. The asset store in Unity is something that should be taken full advantage of.

8.1 Applying Assets to Scene 1

Applying these prefabs to the level, resulted in the beginning of the 'spawn room', which is shown in figure 17. This was the first iteration shown to my peers to assess if the assets included were scary enough, overall good feedback was received however there were minor alterations needed to add atmosphere. This can be shown by comparing figure 17 with 18.

Figure 18: Beginning of Spawn Room

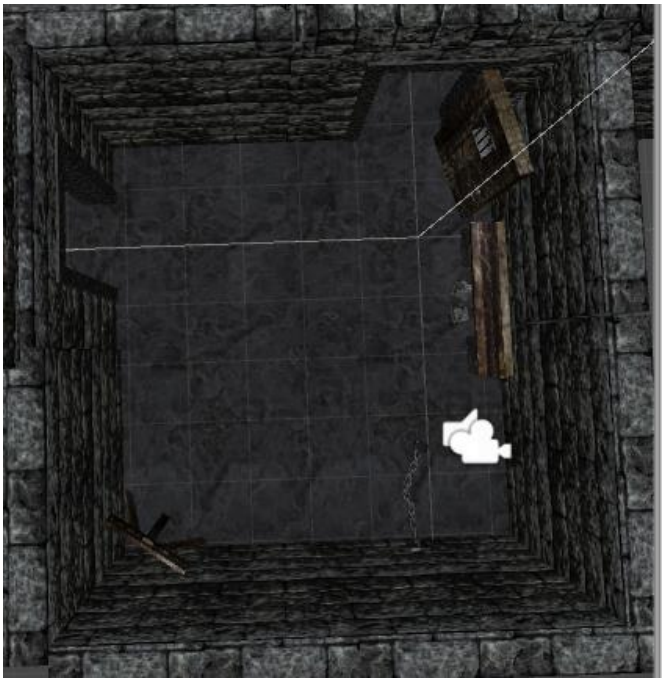


Figure 17: Spawn room with assets

After the process had been repeated several times asking peers for feedback, making use of the iterative waterfall approach, the final layout of scene 1 can be seen below in figure 19.

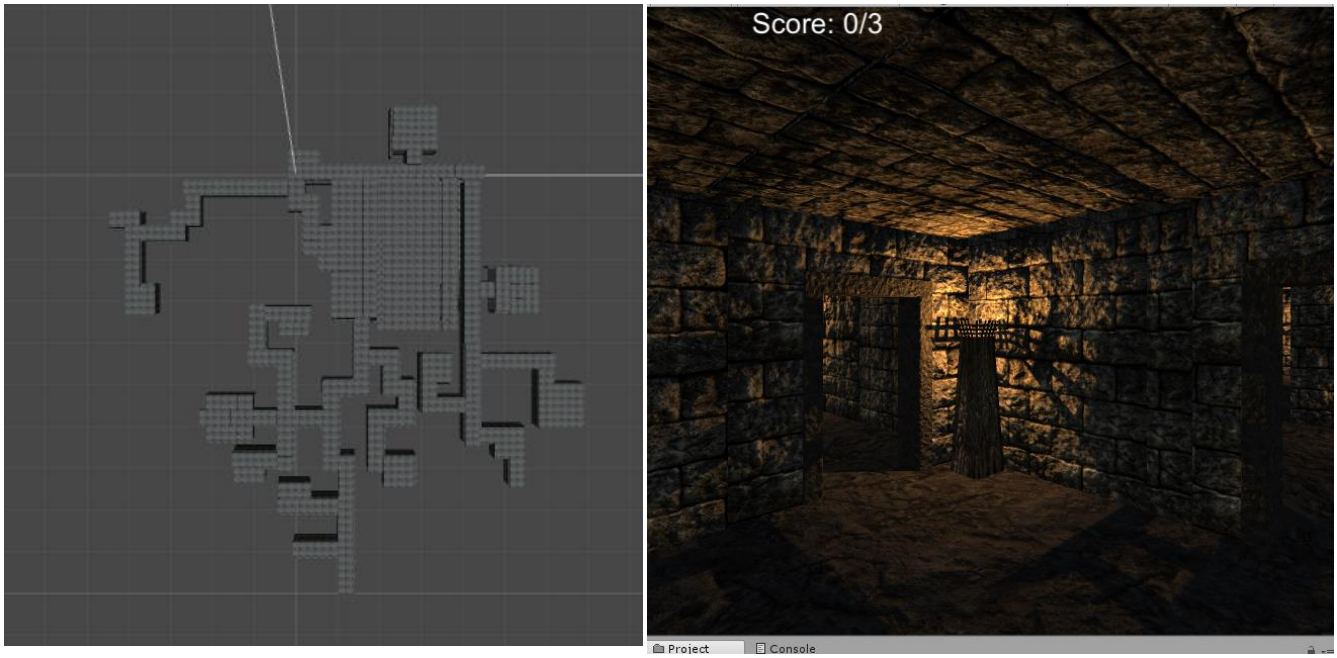
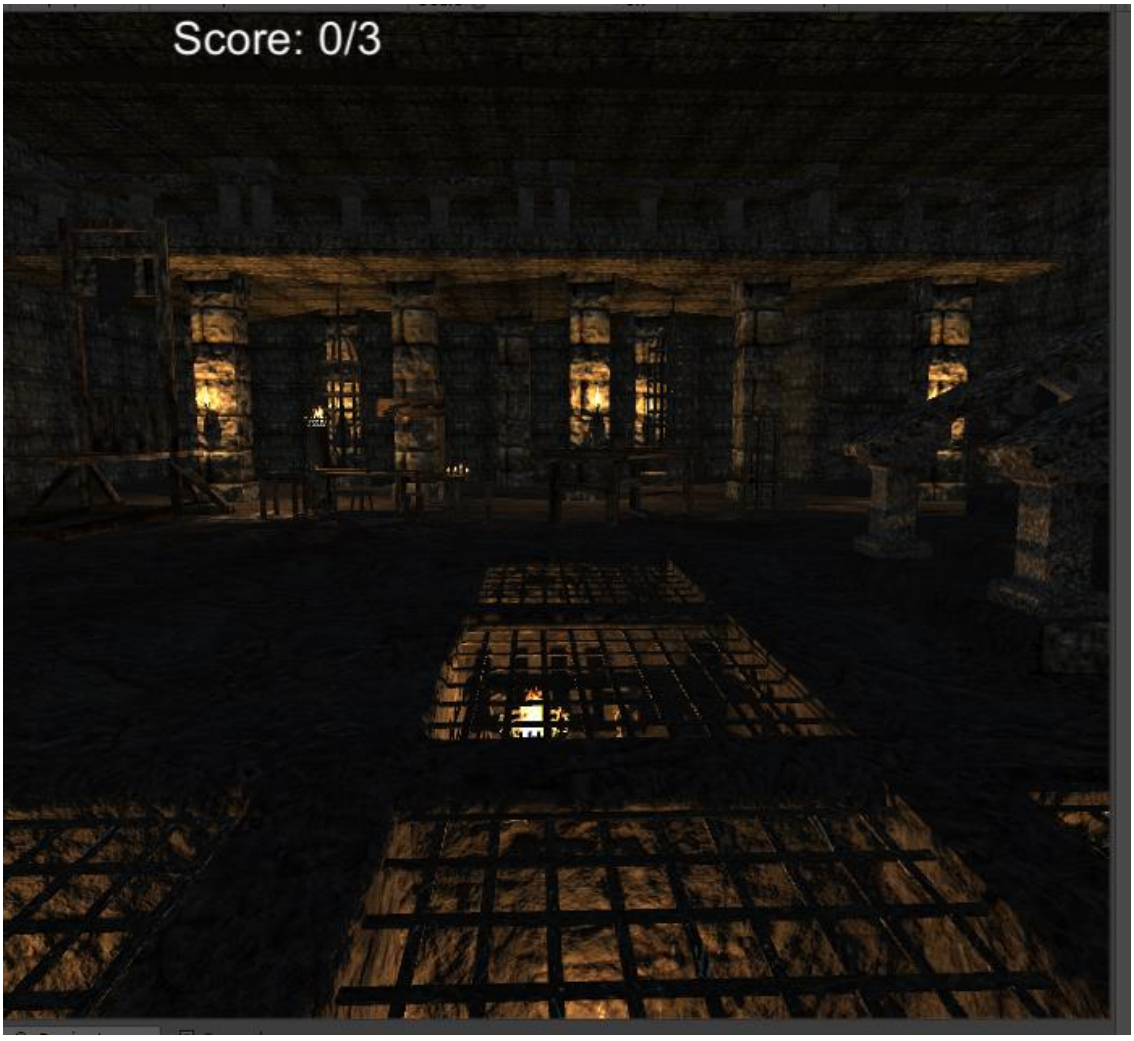


Figure 19: Final Layout of the map. The other screenshots are just the general finished aesthetic



8.2 Creating Event System

The event system is very simplistic, it has a game object that doesn't get destroyed on level load. This is done with a very simple line and is used on multiple game objects throughout the project, the specific code is in figure 20. Beyond that, the event system was attached to nothing in the first level. Separate game objects with colliders acted as triggers with the function OnTriggerEnter they then passed the mouse movement value into the global events script that stored the data in a public float.

```
function Awake () {
        DontDestroyOnLoad (transform.gameObject);
    }
```

Figure 20: Don't destroy on load

To attach two different scripts in Unity, declare a public variable then set the script in the editor. This will allow the different script attached to the collider to pass a value into the public float on the global events script.

Recording the mouse movement makes use of the Event.mousePosition the snippet of code is available in figure 21.

```
public class Mouse : MonoBehaviour {

    int counter;
        private List<Vector3> recordedMousePositions = new List<Vector3>();

    // Update is called once per frame
    void Update () {
        if (counter == 1)
        {
            recordedMousePositions.Add(Input.mousePosition);
        }
    }

    void OnTriggerEnter()
    {
        counter++;
    }
}
```

Figure 21: Simple record mouse and store on trigger

8.3 Create Prefabs for PCG

Due to the graph grammar method of procedural generation, the easiest way to achieve a nice generation algorithm was to create prefabs of the rooms (see figure 22). These all had to have at least 1 doorway in, creating an array of the different doorways in the prefab in a simple script then creating a OnDrawGizmo attached to the individual doorways, this will be aligned with other doorways in the prefabs to connect them in the right orientation. This process is done for each of the rooms, a layout of some of the rooms can be seen in figure 23.

For the prefabs, the level complete scenario is the collection of 3 cupcakes. Instead of varying the cupcake spawn, there were 3 prefab rooms called 'end room' these end rooms had a cupcake in. There's a prefab for a start room to go along with these end rooms.

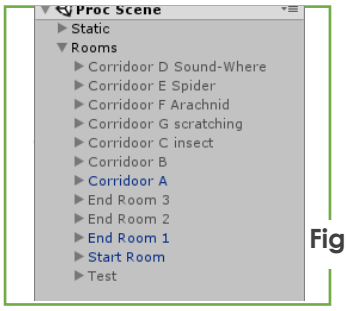


Figure 22: The list of the spawn able rooms

The other prefabs are called corridors, these corridors unlike end rooms are empty but have a spawn point for any monster. This monster is directly correlated to the global event system.

Figure 23: A preview of different prefab rooms available to spawn

8.4 Develop the Procedural Algorithm and Base Dependencies on Event System

The procedural generation algorithm involved is a relatively straightforward one, first creating a list of the possible rooms and defining what is the start room and end rooms (figure 24). Once this list has been populated we need to generate the spawning room or 'start room' followed by however many iterations until we spawn the end rooms. The code for this is in figure 25.

```
public Room startRoomPrefab, endRoomPrefab;
public List<Room> roomPrefabs = new List<Room>();
public Vector2 iterationRange = new Vector2(3, 10);

List<Doorway> availableDoorwars = new List<Doorway>();
```

Figure 24: Prefabs for the start room and end room, iteration range global variable

This iteration is a global value so it can be changed in the menu easily. However, this isn't the most important thing, spawning the start room then adding the doorway to the list of doorways to then spawn the iterations of random rooms is. This list of doorways will be accessed for all of the generation for full code see figure 26.

```
yield return startup;
Debug.Log("Place Start Room");
PlaceStartRoom();
yield return interval;

int iterations = Random.Range((int)iterationRange.x, (int)iterationRange.y);

for(int i = 0; i < iterations; i++)
{
    Debug.Log("Place random rooms from list");
    PlacedRoom();
    yield return interval;
}

Debug.Log("Place End Level");
PlaceEndRoom();
yield return interval;

Debug.Log("Level Generatation Completed");

yield return new WaitForSeconds(3);
Debug.Log("Reset Level Generation");
StopCoroutine("GenerateLevel");
StartCoroutine("GenerateLevel");
```

Figure 25: Iterations for placing the rooms

Figure 26: More of the Starting room code and adding to the doorways

```
void PlaceStartRoom()
{
    Debug.Log("Placed Start Room");
    startRoom = Instantiate(startRoomPrefab) as StartRoom;
    startRoom.transform.parent = this.transform;

    AddDoorwaysToList(startRoom, ref availableDoorwars);

    startRoom.transform.position = Vector3.zero;
    startRoom.transform.position = Quaternion.identity;

}

void AddDoorwaysToList(Room room, ref List<Doorway> list)
{
    foreach (Doorway doorway in room.doorways)
    {
        int r = Random.Range(0, list.Count);
        list.Insert(r, doorway);
    }
}
```

```
void PositionRoomAtDoorway (ref Room room, Doorway roomDoorway, Doorway targetDoorway)
{
    // Reset room position and rotation
    room.transform.position = Vector3.zero;
    room.transform.rotation = Quaternion.identity;

    // Rotate room to match previous doorway orientation
    Vector3 targetDoorwayEuler = targetDoorway.transform.eulerAngles;
    Vector3 roomDoorwayEuler = roomDoorway.transform.eulerAngles;
    float deltaAngle = Mathf.DeltaAngle (roomDoorwayEuler.y, targetDoorwayEuler.y);
    Quaternion currentRoomTargetRotation = Quaternion.AngleAxis (deltaAngle, Vector3.up);
    room.transform.rotation = currentRoomTargetRotation * Quaternion.Euler (0, 180f, 0);

    // Position room
    Vector3 roomPositionOffset = roomDoorway.transform.position - room.transform.position;
    room.transform.position = targetDoorway.transform.position - roomPositionOffset;
}
```

Figure 27: Using MonoDevelop as Visual Studio wasn't working my laptop. This places rooms in the right orientation

In figure 27 we can see the snippet of code for the room placer. This places the doorways and their conjoining rooms in the right orientation.

So far in the code shown, we have a list of the doorways available in a given room, this doesn't actually do anything. The code in figure 27 doesn't inherently do anything, it just moves the room to the right position if the room hasn't been placed and if the doorways bool returns as false. First, we reset the room to 0,0,0 on the global map then we rotate the room to match the OnDrawGizmo that was added to each of the doorways separately. This combined with a simple bool to check if a doorway has been placed makes the entirety of the procedural algorithm.

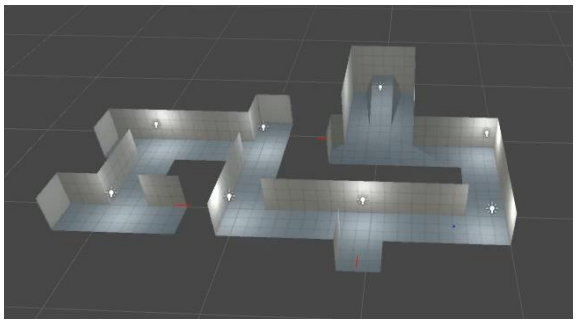
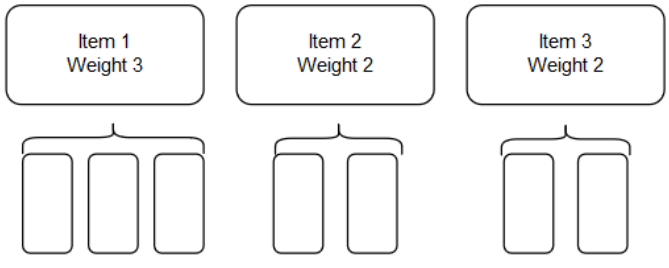


Figure 28: Generated using basic prefabs

Figure 28 on the left is an entirely generated level using basic prefabs.

Basing the dependencies on the event system is just setting the enemy spawn by creating a list of the available prefabs (enemies) then just spawning one in that place. This is a very easy method of varying the enemies spawn. In the global event, a random number generator with a float value that's not defined is added.



In figure 29 you can visually see the weighted means of spawning an enemy. There are 6 different enemies, each enemy has a default weighting of 2.

Figure 29: How to weight probability. This increases or decreases the probability of the items

Writing some pseudocode for these functions in figure 30 so the principle can be easily examined and referred too at a later time without having to read through several different scripts. Within figure 30 it stores the mousePosition in an array then simply sets the weighting of an enemy higher or lower. This would in turn make the enemy spawn as set in the prefab rooms with a higher chance to spawn the enemy if the player reacts. This is a clear use of EDPCG as described within the literature review.

```
  var vectorArray : Vector3[] = new Vector3[1000];
  var currentArrayPos : int = 0;

  private var body : Rigidbody;
void OnTriggerEnter  () {
      if(Time.deltaTime = 0.f){
          print(currentArrayPos.ToString);
          currentArrayPos = 0;
          vectorArray[currentArrayPos] = Input.mousePosition;
          currentArrayPos++;

      } else if (Time.deltaTime = 1.0f){
          print(currentArrayPos.ToString);
          vectorArray[currentArrayPos] = Input.mousePosition;
          currentArrayPos++;
          if (vectorArray[currentArrayPos] => 600){
            enemy1 = 3;
            }
          if (vectorArray[currentArrayPos] <= 400){
            enemy1 = 1;
            }
           else (){
            enemy1 = 2;
           }
      }
```

Figure 30: Psuedocode to avoid having to post several scripts

8.5 Other Method of PCG Explored

```
}

// Update is called once per frame
void Update () {

}

public void DisplayMap() {
    string output = "";
    for (int r = 0; r < mapRows; r++) {
        for (int c = 0; c < mapColumns; c++) {
            output += map [r, c];
        }
        output += "\n";
    }
    Debug.Log (output);
}

private void InitializeMap() {
    map = new char[mapRows, mapColumns];

    // Put 'X's in top and bottom rows.
    for (int c = 0; c < mapColumns; c++) {
        map [0, c] = 'X';
        map [mapRows - 1, c] = 'X';
    }

    // Put 'X's in the left and right columns.
    for (int r = 0; r < mapRows; r++) {
        map [r, 0] = 'X';
        map [r, mapColumns - 1] = 'X';
    }

    // Set 'O' for the other map spaces (which means 'free').
    for (int r = 1; r < mapRows - 1; r++) {
        for (int c = 1; c < mapColumns - 1; c++) {
            map [r, c] = 'O';
        }
    }
```

Figure 32: New Method of PCG

Another version of PCG explored for this project was generating a map then building level off of the map. It uses a simple character conversion; this generates the different maps.

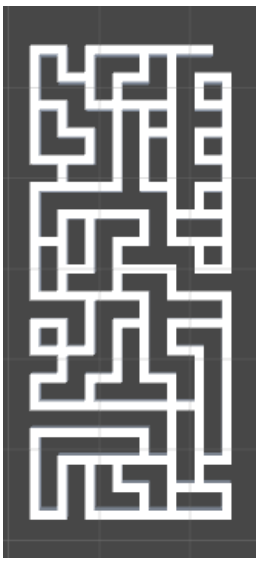


Figure 31: Procedurally generated map layout

8.6 Release Version

In the final iteration of the game a few things were added on for ease of access and quality of life. To adhere to the ethical guidelines set out, the player is immediately greeted with figure 32 the main screen. Each button is event driven, clicking start loads the player into the first level, settings enables the player to change various aspects of the game (see figure 33).

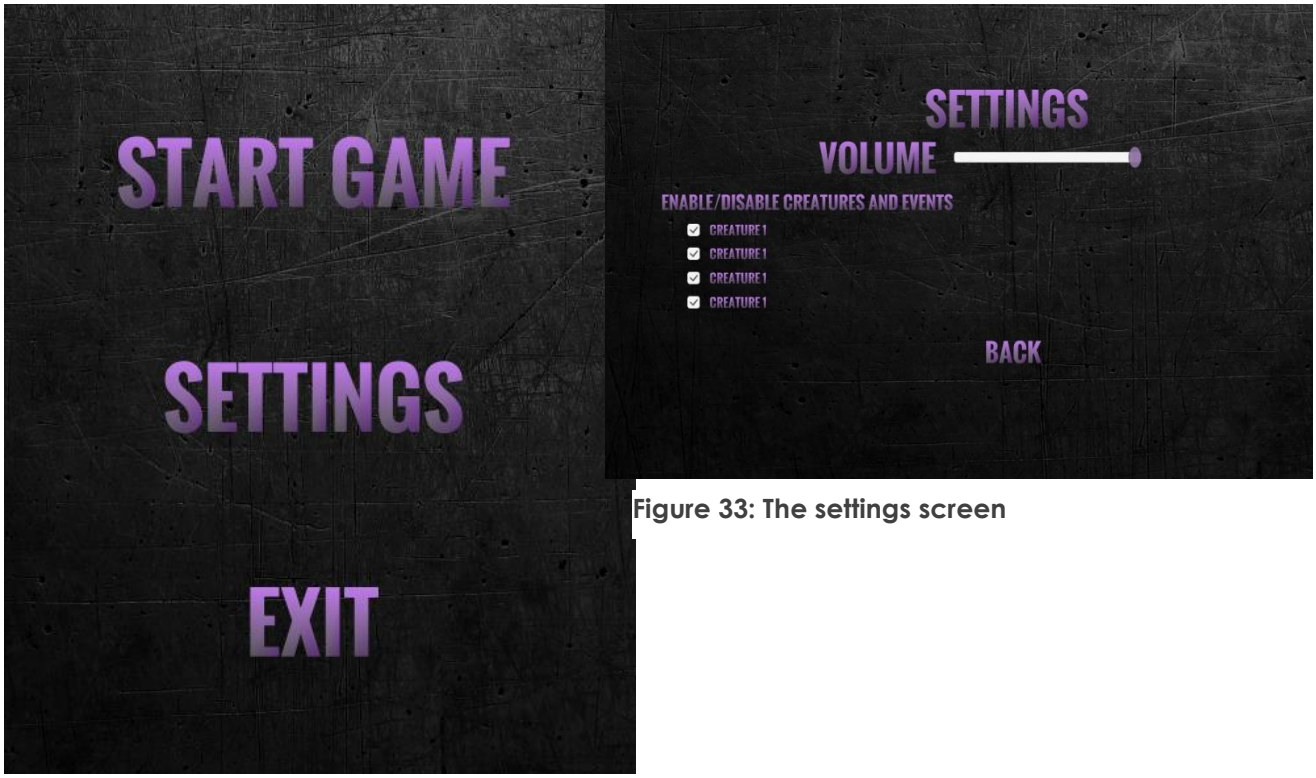


Figure 33: The settings screen

Figure 34: The first screen the player see's opening up the game

Once the player has loaded into the game, they will see figure 19, the spawn room. Then the player is free to explore in either direction as both will need to be explored eventually to provide information to the global event system. In the release version, most of what the player is required to do to play the game will be learnt as they play.

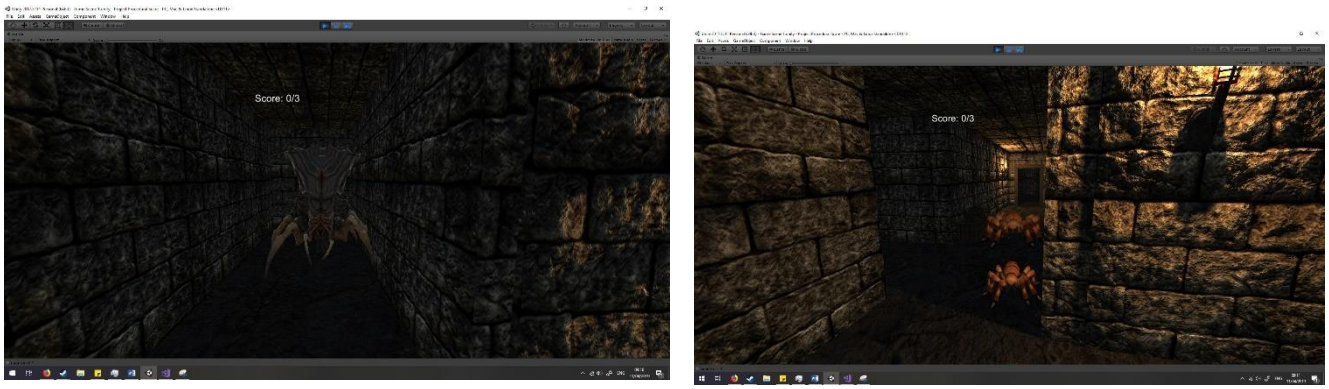


Figure 35: Two of the monsters in the game, this monster on the left is titled "Arachnid" due to the 8 legs and the multitude of spiders on the right

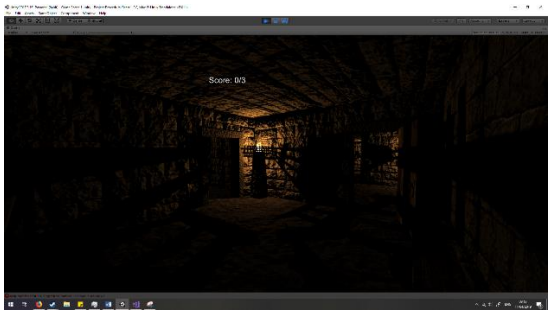


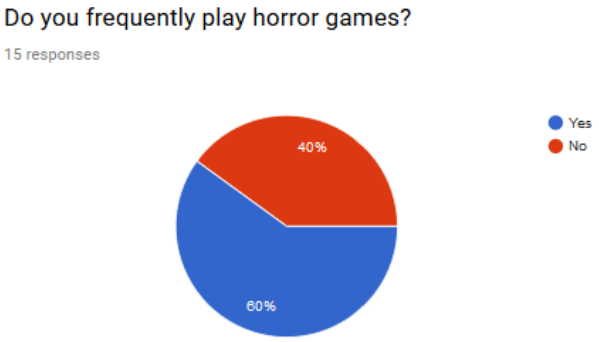
Figure 36: A different lighting model for the level that was considered. This makes the game inherently scarier due to being darker.

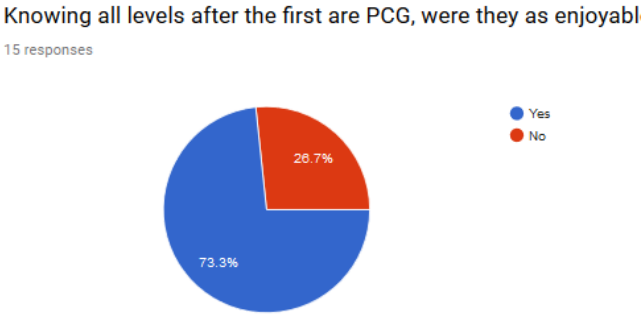
9.0 Testing and Conclusion

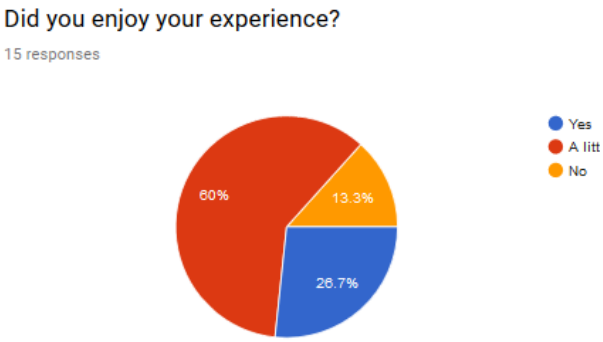
Testing was one of the most important parts of this project. This was to assess the validity of EDPCG in horror games, to meet this aim gathering information was important. Testing the artefact on peers, the process of this was; explaining in detail about the idea of the game and the contents within, giving them the exact specifications of the testing. That being stating it would be best if they played through at least two iterations of the PCG level, this was so they could get a feeling for it and then asking them to put on the noise cancelling headphones as to allow them to be completely engrossed in the game. If anything wasn't understood in the explanation, explaining again would be an option. Proceeding that, the participant read through and signed the consent form [see appendix].

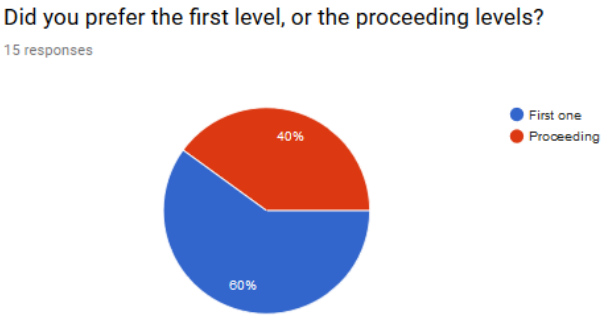
There was constant testing of the assets throughout the project, however due to the nature of the testing basis it was generally the case of asking a peer what they thought of a specific asset.

Once the participant had gone through the horror game, they would be presented with a multiple-choice questionnaire [see appendix]. Gathering the participants through peers and friends, 15 responses were gathered. Below are the results.









From this we can gather a few things, there appears to a correlation from the participants who frequently play horror games to the participants who preferred the first level. This may be because they have a larger base of comparative game. Another potential conclusion is, the ones who answered yes to frequently playing games had a higher bar for the quality of the proceeding levels. A large portion of the participants answered "a little" for the enjoyment of the experience. This could be improved by adding in scarier assets however the artefact generally fulfilled the project aims.

However, one of the most notable findings from this is the high percentage of participants who thought the PCG levels were just as enjoyable as the hand-crafted level. This could be due to the method of procedural generation as all of the rooms were predefined prefabs, giving the effect of hand crafted when in fact the rooms were randomly generated.

This artefact meets the aims and goals as set out in section 1.2, it created a subjective scary level as the initial test basis. This was adhered too during tests throughout the development based on the methodology. The level design was carefully thought out and all of the key features set in 2.1.2 were not necessarily done perfectly but were

adequate for the study. It had significant monster design and thoughtfully placed lighting and event driven generation.

The artefact created a procedural level that included carrying the monsters and sounds based on the measurements set out by the player in the original level. This procedural level had limited generation and possibilities due to the scope of the project.

It was also able to critically evaluate the PCG levels compared to the original level. The questionnaire had some interesting results however no further conversations of note took place.

Some of the ideas put forward by this project have been carefully thought out however further study would be required by parts of this. As this is the first study into PCG validity in the horror genre it provides a valid basis for all future research.

One of the suggestions of thought or further research is; Terrain PCG being considered for the same proposal. This may receive better results as the asset design is easier to be refined to the 4 principles set out in 2.1.2.

Another suggestion is to use a PCG-based game model as set out by Smith, et al, 2012. This could be a good area of research as it would apply a completely customized experience to each player that plays the game. This would be a completely unique take on the horror game genre.

From this research alone EDPCG appears to have a valid place in horror games. However, due to this being mainly preliminary no definitive conclusive data can be gathered.

Personal views after developing the artefact is that the element of narrative cannot be understated in a horror game. Scaring the player as a whole gets stale fast, the player needs another gripping factor to keep them engaged. Mixing PCG and narrative could be a very interesting school of thought.

10.0 Critical Reflection

The general concept of the study was well received, the game intrigued the participants. This initial intrigue suggests the premise for an EDPCG game is well received however I do feel as though a lot more work needs to go into this study of thought until it can be used to create fully functional game that adhere to every principle set out in section 2.1.2 while keeping the player fully engaged.

In terms of procedural generation, in hindsight I would have tried other iterations and styles of generation such as terrain and Perlin noise. Originally, I wanted to implement two types of procedural generation and alternate the player between them, I'm not happy with how linear the procedural generation turned out to be. There are only a finite number of generations with the limited corridors and end rooms.

Reworking the entire method of PCG and starting with a tile-based approach could have been beneficial. Essentially starting from 2D generation then applying the same method to a topological 3D method.

Personally, and generally speaking the artefact produced was not of high enough quality as I believe a neural network could have improved the believability of the monsters in the game and turn the results in section 9.0 from "a little" amount of enjoyment to an overly enjoyed experience.

I'm happy with the research done into the areas of PCG and how horror could be incorporated into this structure; and I'm happy to provide a basis for other researchers to build their sources.

The project was well thought out and the research was done well but the artefact could have been done better. Being given prior knowledge of how best to test for my proposal I would have used a terrain generation method instead. This could have provided a better basis for different generation such as structures within the terrain and entire valleys and different topology.

As stated in section 2.3 constraining the results from the participants by using a multiple-choice questionnaire could have been the wrong approach as it didn't provide any qualitative data. Analysing this data could have been a good basis for future research.

Another aspect of the project that was almost left out completely was that Sonanification as stated by (Lopes & Yannakakis, 2015) this is the idea of an entirely procedurally generated sound system. Given a different method of PCG, such as terrain as explained earlier I believe including Sonanification would have been a lot easier to implement due to the reduced complexity of having to only great Perlin noise rather than the orientation and method of display of entire prefab rooms.

Broadening the scope of the project to look at "Narrative-Driven" horror games could have yielded much more positive results, but I'm pleased I focused on one rather than using both styles of game. Retrospectively wanting to change what one was focused on during this project however.

The project didn't exhibit many issues from the start, at was always fairly straightforward however, due to the implementation of varying kinds of PCG and

EDPCG being explored for the first time by the developer, there were definitely code misuse and bad practices put into place. The research-based methodology could have been revised on a few iterations. This could be then set to include a separate ideology, for example making better use of the qualitative data and storing it somewhere to then later refer to it in the conclusion stage would have been beneficial to the project and identifying what did and what didn't go well throughout the project.

One personal criticism is the asset placement in the first level. With such a focus on the believability, a lot of the textures of the assets overlap and cause graphical glitches on graphically lower machines than the one it was developed on. There needs to be some sort of texture control, as that is the main criticism from the crafted first level.

References

Adams, D., 2002. *Automatic Generation of dungeons for computers games,* Sheffield: Sheffield University Dept. Computer Science.

Alecu, F., 2010. Blender Institute-the Institute for Open 3D projects. *Open Source Science Journal 2,* 2(1), pp. 36-45.

Aleksandr, 2014. *Documentation, Unity scripting languages and you.* [Online] Available at: https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/ [Accessed 09 04 2019].

Anon., n.d. *Urho 3D Github.* [Online] Available at: https://urho3d.github.io/ [Accessed 07 04 2019].

Bonilla, A., 2017. *What Makes a Horror Game?.* [Online] Available at: https://www.gameskinny.com/n6l4l/what-makes-a-horror-game [Accessed 2019].

Carli, D. M. D., Bevilacqua, F., Pozzer, C. T. & dOrnellas, M. C., 2011 . A Survey of Procedural Content Generation Techniques Suitable to Game Development. *Brazilian Symposium on Games & Digital Entertainment,* pp. 26-36.

Crookshanks, E., 2015. Development methodologies and SDLC. *Practical Enterprise Software Development Techniques,* pp. 37-59.

Dart, I., De Rossi, G. & Togelius, J., 2011. SpeedRock: procedural rocks through grammars and evolution. *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games,* Issue ACM, p. 8.

De Leeuw, B., 1997. *Digital cinematography: Lighting and photographing computer generated animation.*. s.l.:Morgan Kaufmann.

Dingman, H., 2018. *20 terrifying PC horror games to play with the lights off.* [Online] Available at: https://www.pcworld.com/article/2838314/15-terrifying-pc-horror-games-to-play-with-the-lights-off.html [Accessed 3 April 2019].

Durfee, E. & Rosenschein, J., 1994. Distributed problem solving and multi-agent systems: Comparisons and examples. *Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop,* pp. 94-104.

Freeland, C., 2004. Horror and art-dread. *The Horror film,* pp. 189-205.

Games, G. S., n.d. *Mapgen: Cellular Automata.* [Art] (https://www.gridsagegames.com/blog/gsg-content/uploads/2014/06/cogmind_map_automata_mine.png).

Graham, R., cCabe, H. & Sheridan, S., 2004. Nerual networks for real time pathfinding in computer games. *The ITB Journal,* 5(1), p. 21.

Haas, J. K., 2014. *A history of the Unity game engine,* s.l.: s.n.

Hendrikx, M., Meijer, S., Van Der Velden, J. & Iosup, A., 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications,* Issue 1, p. 1.

Horti, S., 2018. *Four developers of scary games explain how to make scary games very scary indeed.* [Online]
Available at: https://www.rockpapershotgun.com/2018/06/21/four-developers-of-scary-games-explain-how-to-make-scary-games-very-scary-indeed/
[Accessed December 2019].

Hunicke, R., LeBlanc, M. & Zubek, R., 2004. MDA: A formal approach to game design and game research. *Preceddings of the AAAI Workshop on Challenges in Game AI,* 4(1), p. 1722.

Jarrett, D. C., 2011. The lure of horror. *The Psychologist,* Volume 24, pp. 812-815.

Jick, T., 1979. Mixing Qualitative and quantitative methods: Triangluation in action. *Administrative science quarterly,* 4(24), pp. 602-611.

Johnson, L., Yannakakis, G. & Togelius, J., 2010. *Cellular automata for real-time generation of infinte cave levels,* New York: s.n.

Jung, C., 1912. *Psychology of the Unconscious.* s.l.:s.n.

Kalliamvakou, E. et al., 2014. The promises and perils of mining GitHub. *Proceedings of the 11th working conference of mining software repositories*, May, pp. 92-101.

Krzywinska, T., 2002. Hands-on horror. *Screenplay: Cinema/videogames/interfaces,* pp. 206-223.

Lopes, P. L. A. & Yannakakis, G., 2015. Sonancia: Sonification of procedurally generated game levels. *Proceedings of the 1st computational creativity and games workshop.*

Olsen, J., 2004. *Realtime procedural terrain generation,* s.l.: s.n.

Ragunath, P. et al., 2010. Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC). *International Journal of Computer Science and Network Security,* 1(10), pp. 112-11.

Ramachandran, N., 2009. *Opinion: What Makes a Horror Game Truly Scary? Exclusive.* [Online]
Available at:
https://www.gamasutra.com/view/news/113338/Opinion_What_Makes_a_Horror_Game_Truly_Scary.php
[Accessed December 2018].

Roland van der Linden, R. L. a. R. B., March 2014. *Procedural Generation of Dungeon.* [Art].

Royce, W., 1970. The software lifecycle model(waterfall model). *Proc. Westcon,* Volume 314.

Savo, 2014. *What Makes the Perfect Horror Game?.* [Online]
Available at: https://tay.kinja.com/what-makes-the-perfect-horror-game-

1619847904
[Accessed 2019].

Shaker, N., Togelius, J. & Nelson, M., 2016. *Procedural content generation in games,* s.l.: Switzland: Springer International Publishing.

Smelik, R., Tutenel, T., de Kraker, K. & Bidarra, R., 2010. Integrating Procedural Generation and Manual Editing of Virtual Worlds.. *Proceedings of the 2010 Workshop on Procedural Content Generation in Games.*

Smith, G., Othenin-Girard, A., Whitehead, J. & Wardrip-Fruin, N., 2012. CG-based game design: creating Endless Web. *Proceedings of the International Conference on the Foundations of Digital Games,* pp. 18-195.

Thinkwik, 2018. *CryEngine vs Unreal vs Unity: Select the Best Game Engine.* [Online]
Available at: https://medium.com/@thinkwik/cryengine-vs-unreal-vs-unity-select-the-best-game-engine-eaca64c60e3e
[Accessed 08 04 2019].

Togelius, J., Kastbjerg, E., Schedl, D. & Yannakakis, G., 2011. What is Procedural Content Generation? Mario on the borderline..

Togelius, J. & Yannakakis, G., 2016. Emotion-driven level generation. *Emotion in Games,* Volume Springer, Cham, pp. 155-166.

Unity, n.d. *Unity Programming and you.* [Online]
Available at: https://unity3d.com/programming-in-unity
[Accessed 06 04 2019].

Unity, n.d. *Unity Store.* [Online]
Available at: https://assetstore.unity.com/
[Accessed 01 04 2019].

Wilson, J., 2003. Gantt ChartsL A centenary appreciation. *European Journal of Operational Research,* 143(2), pp. 430-437.

Xiao, C. & Shi, H., 2011. A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security,* 1(11), pp. 125-130.

Xu, M., Jin, J. & Chia, L.-T., 2005. Affective Content analysis in comedy and horror videos by audio. Emotional event detection. *International Conference on Multimedia and Expo,* Volume 4, pp. 4-pp.

Yannakakis, G. & Togelius, J., 2011. Experience-driven procedural content generation. *Transactions on Affective Computing,* 2(3), pp. 174-161.

Ludology

Beneath Apple Manor (Don Worth 1980, available from https://www.myabandonware.com/game/beneath-apple-manor-256)

Rogue (A.I. Design, 1980 available from https://www.myabandonware.com/game/rogue-4n/play-4n)

Borderlands (Gearbox Software, 2009 borderlands official website https://borderlands.com/en-US/welcome/)

Call of Cthulhu (Cynaide, 2018 call of Cthulhu website http://www.callofcthulhu-game.com/en)

World of Warcraft (Blizzard Entertainment, 2004 website available from http://www.callofcthulhu-game.com/en)

Infamous 2 (Sucker Punch Productions, 2011 website for the developer https://www.suckerpunch.com/)

Dark Souls 3 (From Software, 2016 website https://www.fromsoftware.jp/ww/products.html)

Anmesia: The Dark Descent (Frictional Games, 2010 website https://www.amnesiagame.com/#mainx)

Slender: The Eight Pages (Parsec Productions, 2012 website https://www.indiedb.com/games/slender-the-eight-pages/downloads)

Outlast (Red Barrels, 2013 steam store https://store.steampowered.com/app/238320/Outlast/)

Resident Evil (Capcom, 1996 website https://www.residentevil.net/uk/)

Resident Evil 2 (Capcom, website https://www.residentevil.net/uk/)

Silent Hill (Konami, team Silent 1999 https://silenthill.fandom.com/wiki/Silent_Hill_(video_game))

Moon Hunters (Kitfox games, 2016, steam https://store.steampowered.com/app/320040/Moon_Hunters/)

No Man's Sky (Hello Games, 2016, website https://www.nomanssky.com/)

Unreal Engine 4 (Epic Games, C++ , website https://www.unrealengine.com/en-US/)

Unity (Unity Technologies, 2005 with constant work , https://unity.com/)

GitHub (Microsoft, 2008, website https://github.com/)

Appendices

Neighbourhood Procedural Generation

They define the neighbourhood of a cell as a singular cell surrounded by 8 cells. The rule set is iteratively applied in multiple generations, this particular one states that: 1) a cell is rock if the neighbourhood value is greater than or equal to T (T = 5) and floor otherwise; and 2) a rock cell that has a neighbouring floor cell is a wall cell. Based on these rules cave like structures can be produced this method allows real-time infinite map generation.

Isotropic

Having a physical property which has the same value when measured in different directions.

Cavern

A large cave or chamber in a cave

Prefab

Within Unity you can create base of an asset with all of it's properties still inside the prefab. This can include different animations and controls or rigid bodies. It acts as a template.

Acronyms

PEM: Player Experience Modelling

EDPCG: Experience Driven Procedural Content Generation

PCG: Procedural Content Generation

Questionnaire

Do you frequently play horror games?

- ○ Yes
- ○ No

Did you enjoy your experience?

- ○ Yes
- ○ A little
- ○ No

Knowing all levels after the first are PCG, were they as enjoyable?

- ○ Yes
- ○ No

Did you prefer the first level, or the proceeding levels?

- ○ First one
- ○ Proceeding

SUBMIT

Research Consent Form

Name of Researcher: Jez Stuart Edward Horton

Title of Study: Assessing the Validity of Experience-Driven Procedural Generation in Horror Games

Please read and complete this form carefully. If you wish to participate in this study, circle the appropriate responses and sign and date the form at the end. If you do not understand something, and require more information, please do not hesitate to ask.

- I have had this area of research satisfactorily explained to me in verbal and/or written form by the researcher.

YES/NO

- I understand that the research will involve: The participant will be required to play a horror video game with noise cancelling headphones through a PCG level twice. The participant will be given a questionnaire when they have completed the video game to assess the validity and the aims of the study. The video game should take no longer than 5 minutes, however the participant may take as long as they wish.

YES/NO

- I understand that I may withdraw from the study at any time without having to give an explanation.

YES/NO

- I understand that all information collected about me will be strictly confidential and that I will not be named in any written work arising from this study.

YES/NO

- I understand that you will be discussing the progress of your research with others at the University of Lincoln.

YES/NO

I freely give my consent to participate in this research study and have been given a copy of this consent form for my own records.

Name: ……

Signature: ……

Date: ……