Evaluating the perceived immersion of procedurally generated game levels

Taura J. Greig

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Master of Creative Technologies

Table of Contents

List of Figures

Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

Taura J. Greig

T J Greig

14/10/16

Acknowledgements

The work outlined in this research was conducted following ethics approval from the AUT Ethics Committee (AUTEC) on February 10th 2016 under application number 16/34.

There are a few key people, without whom this thesis would not be a reality. One of those people, was my primary supervisor Andy Conner. His commitment and patience has been a major corner stone of this endeavour. I cannot fully articulate my gratitude for his contribution.

I also would like to acknowledge my secondary supervisor Jan Kruse, for his contribution and feedback to this thesis.

Many thanks to the level designers, Patrick Tsunehiro Paul Tubman, Matthew- Rhys Pinto De Menezes and Bradley Joshua Millen who took time out of their studies to contribute their design skills to this endeavour.

I would like to acknowledge Britta Pollmuller, for making available the equipment and facilities of the digital design department for my research.

I would like to thank the community and department of Colab at AUT, for accepting me into the postgraduate program and providing the resources necessary to undertake my thesis. Five years ago, I would have never imagined what I would be able to achieve today, if it was not for the Colab community and their openness, as well as the various supporting lecturers who instructed me during my undergraduate years at AUT. Not only has it been an eye-opening endeavour, but it has also provided experiences that I will never forget. My time at Colab has been, thus far, the best years of my life and I will always remember them.

Finally I would like to sincerely thank my parents, Claire and David, for supporting me throughout this entire endeavour. Thank you for sticking with me till the end, and seeing this through. This was a very important project for me to undertake and I am thank full for their unfailing support in helping the project become a reality. This accomplishment would not have been possible without them.

Thank you.

Abstract

Interactive Computer Games have grown in complexity as the game industry has matured into a multi-billion dollar market. As games have grown in complexity, the content that is produced for games has grown in both complexity and production cost. This has motivated research into Procedural Content Generation. It is generally thought that developments in Procedural Content Generation has the potential to reduce costs of content production in game development. However, currently Procedural Content Generation is primarily concerned with the playability and usability of generated content. There is little research that investigates the player's perception of generated content. The goal of this research is to evaluate the perceived immersion of a game developed utilizing Procedural Content Generation. A game called Vektor was designed and implemented for this study. Two variants of the game were produced. One of those variants featured levels designed by a human designer. The other variant featured levels that were evolved using a genetic algorithm. The immersion scores of each variant was recorded from participants who played on of the game variants using a questionnaire. Statistical analysis was used to identify where significant differences in immersion occurred.

1 Introduction

Interactive computer games have grown from a niche market to a multi-billion dollar industry (Williams, 2002). Today, video games can be seen as another part of media in contemporary culture (Squire, 2002). As the industry has grown and expanded, the complexity of games and their development have expanded alongside it. While games used to be simple 2D adventures like that of Pong, today they feature complex 3D virtual worlds and characters. Games have even be called art (Smuts, 2005), and there are numerous examples of games with novel systems and set ups that would be identified as such.

However as the games have become more complex, the resources needed to make them have increased as well (Edwards, 2006) and it is not unusual for contemporary games to have million dollar budgets. There is also considerable risk in the production of video games, limiting the ventures studios make in the production of the video games.

One of the significant causes of the increased expense is the production of game content. As the complexity of the games have increased, the intricacies of game content has increased as well. To produce good game content, designers often need to iterate several times over their work to ensure quality content. This makes the production of games an intense and time consuming process.

In game research, Procedural Content Generation (PCG) has arisen to answer this problem. While there are numerous examples of commercial games that already use PCG, they usually feature simple PCG methods, and they are highly specialised to the game in question, limiting the universal application of the method. Examples of advanced PCG in commercial games are rare, although that may change in the near future.

It is generally thought that use of advanced methods can improve PCG‘s ability to produce content of a higher quality. These advanced methods may include Artificial Intelligence and Evolutionary Computation. The desired goal is to produce systems that can autonomously produce content of the same or higher quality than human designed content, and that these systems would be universal between different game designs. This would enable the production costs of games to go down allowing more and broader ventures in the industry. PCG could also augment the creativity of human designers, enabling more affordances in the designs and scopes of games.

However, to a large extent, current research into PCG is primarily concerned with the technical feasibility of the methods investigated. The research typically addresses whether the content is technically playable and usable in the games they are produced for. There is little research that investigates the impact of PCG in terms of how the players perceive the generated content and how it impacts the game playing experience. Even if the content is technically feasible, would it contribute positively to the games perceived enjoyment by players?

This is an important question. In order for the goals of PCG research to be realised, the produced content needs to be enjoyable, as well technically feasible. Content that is generated needs evoke a sense of presence in the player. Presence (Singer & Witmer, 1998) refers to the experience of being in a virtual environment while physically situated in another. Presence itself relies on the interplay between involvement and immersion. Measurement of this is subject to the experience of player however, and is thus a subjective measure.

The complexity of the human designed content needs to be matched in these respects. This research is therefore intended to undertake a preliminary evaluation of the impact of using PCG on the game playing experience. In particularly, it uses the concept of immersion as defined by Weibel & Wissmath (2011), as a measure of the quality of the game playing experience. Would the use of an evolutionary based PCG method contribute to the perceived immersion of a game?

To evaluate this, this research conducts a game play test with human participants, in which they play a game with levels designed by either a human or a PCG method. The participants are not informed of the version of the game that they are playing. The game in question was developed for this research, and is called Vektor. It is a top down shooter developed with the Unity game engine (*Unity*, 2016). The game has been intentionally designed to test the hypnosis established in this research. Participants fill out a game immersion questionnaire taken from research in evaluating players perceived immersion of a game (Brown & Cairns, 2004a). The results are then compared to evaluate the immersion of each of the two variants of the game.

The rest of this thesis is structured as follows. Chapter 2 provides an overview of the research literature related to the use of AI in games in general, and specifically to the use of PCG in the game design process. Chapter 3 outlines the methodological stance of this thesis, including a description of the experimental design of the evaluation phase. Chapter 4 describes the implementation of both the game developed in this study and the details of the approach to

PCG utilised. Chapter 5 presents the results from the playtesting evaluation and finally Chapter 6 concludes the thesis.

2 Literature Review

This research outlined in this thesis specifically relates to the evaluation of the perceived immersion in a game that has been developed utilizing elements that are procedurally generated through the use of an evolutionary computation technique. As such, this work is situated in the context of utilising computational intelligence approaches in the game design process. This in itself in an even wider context of game design theories and practice, some of which utilise artificial intelligence or computation intelligence techniques embedded in the game or supporting the design process. There is a variety of literature and pre-existing research that cover these areas and this chapter reviews research that is related to this research to provide an appropriate context to justify and support the current study.

2.1 Games and Design

It may not be apparent to observers who are not familiar with the culture and industry surrounding games, that the various systems that comprise a computer game need to be carefully designed. This is not surprising, as it has been noted that the field of game design has "drifted along under the radar of culture, producing timeless masterpieces and masterful time-wasters without drawing much attention to itself" (Lantz, 2004, p. x). Game design does not just consider technical contexts, but also incorporates a human computer interaction context that considers issues such as flow (Chen, 2007; Csikszentmihalyi & Csikszentmihalyi, 1992), immersion (Brown & Cairns, 2004b) and engagement (Boyle, Connolly, Hainey, & Boyle, 2012) in the production of player-centric experiences. As a result, an entire discipline has emerged centred around the design of games, focused on how games work and how game systems influence players. The design of the game can make or break its success and specific examples of successful or failed games have been analysed to identified particular design features or decisions that influenced the reception of the game (Rollings & Morris, 2003).

The video game industry has grown dramatically over the past decade, cutting into traditional media in participation and revenues as it becomes part of mainstream media culture (Connor & Gavin, 2015; Williams, 2002). The game development industry is increasing in complexity, with both physical and economic interactions between hardware and software (Nichols, 2014). Video games have driven the evolution of computer hardware and multiplayer games in particular created an insatiable demand for better graphics systems (Kushner, 2002), and the resulting diversity and capability of hardware has produced a dynamic market which produces more games that utilise these advances (Jörnmark, Axelsson, & Ernkvist, 2005). Such a market creates demands on game development companies to produce increasingly complex games to maintain their market share. Game design has therefore become a challenging activity and there is a need for game design schemata that guide designers to understanding how their design choices influence the player experience that results from the game. The following section details the schema that was used in the development, execution and discussion of this research.

2.1.1 Mechanics Dynamics, Aesthetics Framework

Interactive computer games are comprised of a series of systems that operate the game in response to player input. What these systems are and how they interact with the player varies considerably between games. Two games can have completely different systems, yet still be well received by an audience. Because of the many possible permutations of these systems in games, classifying which are commonly 'good' and 'fun' is an impossible task. Yet, as it will be discussed, thinking in these terms is a mistake and adds to the confusion and difficulty in analysing games media. Instead, the MDA framework (Hunicke, LeBlanc, & Zubek, 2004) proposes that game systems are looked at through three lenses, which emphasizes how the systems work together and contribute toward the desired game experience.

The MDA framework is a framework for the analysis and design of games that postulates that all the parts of a game systems contribute to the whole game experience, and the better the systems are designed around each other, the more coherent the end user experience becomes. The design of the all the systems of a game should be well thought out and contribute to the designer's desired experience.

The framework formalizes a schema for relating the systems of the game to the end user experience by breaking the game into three components, Mechanics, Dynamics and Aesthetics.

Mechanics refers to the base systems that operate the game. All the code, game logic, math, and even properties of the game play elements, are considered to be mechanics of a game in this schema. This is because each of things either are a defined set of rules or follows a set of rule that govern how things behave in the game.

The behaviour that emerges from these mechanics are referred to as Dynamics. Dynamics are the run time behaviour of the mechanics. Although mechanics puts emphasise on the systems and rules in a mechanics sense, Dynamics is focused on how the system of a game playout over time. While the math of a game character's movement would be considered a mechanic, the dynamic of the game character's movement would be how the math would manifests over time. It is important to understand this distinction.

It is not accurate to evaluate a game based on its perceived 'fun'. Two games may have different systems, yet they may both be considered 'fun'. Instead of looking to the players perceived 'fun' to evaluate the systems in a game, it is more meaningful to understand the experience the systems of the game produce. Aesthetics is the emotional response a player may have to a game's system. Aesthetics are essentially how the Dynamics feel to the player. Instead of being on linear dimension between 'good' and 'bad', the Aesthetics lens is concerned with what systems create what emotional responses, as there are a variety of emotional responses a game can produce.

This framework will be used as the game design schema for this research. This is important, as conclusions about the results of data will not be meaningful without a design schema to contextualize the outcome. This seems to be lacking in other research into PCG methods, as usually the technical feasibility is the focus of the research. It is recognised in this research that design feasibility needs to be considered. How does the PCG method fit into the design of a game, and how does the design of the game, influence the implementation of the PCG method. MDA was chosen because, while it is simple, it provides an effective schema for designing and analysing games through an interactive approach.

2.2 Artificial Intelligence and Games

Given that interactive video games deal with virtual worlds and virtual characters, it would not be a giant leap to assume that the latest computer games feature advanced AI that has been developed in the field of computer science. Unfortunately, this is not the case and a recent review of the state of the art in AI in games (Bourassa & Massey, 2012) concluded that:

> "Commercial games, which might be expected to be at the vanguard of AI development, are not. The reality is that the video game industry, naturally, holds the gamer experience as the highest priority, and all game design decisions are made in that context. Consequently, commercial game development has focused on the environment of the game: graphics and character models. The impact is that AI research by the video game industry is meagre and only a small portion of the computational resources in a game are allotted to AI." (Bourassa & Massey, 2012, p. iii)

Most triple A (AAA) games on the market today feature simple smart systems for the behaviour of its characters. In fact it is rare to find games that employ advanced AI systems that have been developed in academia, such as neural networks and Genetic Algorithms. In fact, the term 'Artificial Intelligence' has a distinct meaning in games that is different than Computer Science research. AI in games usually refers to systems that govern the behaviour of the virtual characters (Yannakakis, 2012), but those systems are not considered real AI in comparison to the computer science research.

This due to a multitude of factors, however the most apparent is that relatively advanced AI systems go beyond the scope for what is required for the development of engaging virtual characters in games. Game developers are more interested in creating an engaging experience for the player (their consumer) than experiment with cutting edge smart systems (Woodcock, 2000). This perspective is not unjustified due to the challenges of game production. The complexity of implementing such academic AI systems into a game, and a means to control their impact on the design of a game, is not a trivial matter, and usually the cost in doing so overshadows the perceivable benefits (Yannakakis, 2012). Tradition solutions would typically include approaches such as pathfinding, rule based systems, finite state machines, behaviour trees and goal planning (Yue & de-Byl, 2006). Such approaches are usually good enough for developers as they offer robust control of autonomous characters with enough flexibility to create a desired experience.

That being said, in recent years there has been a push for more interaction between the two communities, game developers and game/AI researchers (Yannakakis, 2012). Research into the use of novel AI in games is increasing (Yannakakis & Togelius, 2015a), and may find its ways into the commercial game industry. Video games have a variety of systems in which the use of the novel AI are applicable. While the industry has settled with A* and Behaviour Trees for character AI, other systems of a game may benefit from the advent of advanced AI. AI can be used in the process of making a game, where the developers and design can offload process in the production of the game to autonomous systems.

2.2.1 AI in commercial games

Although this research is not concerned with the development of advanced NPC AI, the game that was produced for this research did implement methods that are commonly used in commercial games. This was to achieve a game that can be indicative of a standard game one might find in the industry. Because of this, two methods, Behaviour trees and A* pathfinding will be discussed as they are used in the developed game.

2.2.1.1 Behaviour Trees

Behaviour trees have become a standard method for controlling NPC/AI behaviour in the game industry. Behaviour trees can be thought as a tree of behaviours, where each leaf is an action and nodes are logic gates, which control when leafs are accessed under specific conditions. Like a Finite State Machine, the behaviours are defined individually in nodes. The difference, which makes behaviour trees better at handling game AI and scaling as more behaviours are added, is that the behaviours are organized hierarchically, and conditions for how behaviours transitions to other states are not defined in the behaviour, but rather in the nodes that string behaviours together. (Isla, 2005; Simpson, 2014)

Earlier games with AI driven characters would rely on finite state machines, or some ad hoc method similar to a state machine, to control the behaviour of the character. As games have become more complex, and the character's interaction with the virtual environment and the player more intricate, the complexity of the character's AI systems have become more challenging to implement. Finite state machines (or the ad hoc method a game developer would employ), while effective at managing discrete states of behaviour, become problematic to use as, more states are added to the AI's behaviour repertoire. This is because transitions between states must be defined in each state (which means that each state must have knowledge of other states). As more state are added to the system, more definitions between states need to be implemented. This increases the complexity involved and becomes more prone to error. Finite State Machines, hence, are not effectively scalable in the context of game AI development.

Behaviour trees alleviate this design pattern bottleneck by organizing the behaviours that the character can partake into a tree graph structure. Like a normal tree structure, there are edge nodes, known as leafs, and regular nodes that other nodes are attached to. Leaf nodes contains specific actions an NPC can perform, while regular nodes at as the logic gates of the behaviour tree.

These nodes all return a value upon inspection, Success, Failure or Running. Depending on the type of node, the value returned will dictate will proceed to the next node or return up the tree.

2.2.1.2 A and Pathfinding*

The second method that has found widespread use in the game industry is the A* search algorithm for pathfinding. Ever since virtual characters needed to walk around walls and obstacles to arrive at the players' location, (or whatever location is specified by the designer) game developers have required a means for the characters to navigate from one point to another, both effectively and efficiently (Cui & Shi, 2011).

Pathfinding simply refers to a means in which a system finds a path from point A to point B in a given space. Almost all games, from First Person Shooters to Role Playing Games requires some sort of pathfinding for the virtual characters to navigate the virtual world. In these virtual spaces, there are a variety of obstacles due to the configuration of the space. Pathfinding must be able to find paths around these obstacles, so the virtual characters can interact with the player and other characters without becoming stuck behind a wall.

Early solutions to this would include Dijkstra's algorithm and Best First Search. Dijkstra's algorithm is a classic solution to pathfinding and is always able to find the shortest path between two points. The caveat to this is that it is a costly algorithm to perform and is not efficient enough to run in real time. This may not be a problem for applications where performance cost is not an issue, however for games, where a path needs to be computed in real time, this is not desirable. Dijkstra tends to search outward from the source in all directions. Eventually it lands on goal and path is found.

Best First Search (BFS) can be considered the opposite of Dijkstra. The key difference is that BFS choses nodes that are projected to be closer to the goal via a Heuristic function, instead of evaluating the cost from the source. This speeds up the process of finding a path as BFS searches nodes in the direction of the goal. However, as the configuration of the search space becomes complex, BFS cannot guarantee the shortest path. Because of this, even though it is faster than Dijkstra, it is also not adequate for the demands of video games.

A* can be thought of as combination of Dijkstra and BFS. Unlike the methods above, A* uses both the cost from the source and the projected distance to the goal to direct it search. The weight of the two scores, the cost from the source and the projected distance to the goal, can be scaled back and forth to produce Dijkstra or BFS, if it is known that the search time is tolerable or if the search space is not too complex and computation time is short.

Because of this A* is widely used for pathfinding in games, as it finds a path that usually is very close to the shortest path in a viable time frame. It should be noted that specific implementations of A* use optimizations that are appropriate for the developed game at hand. However, the search strategy that defines A* makes it a popular method to use.

2.2.2 AI Assisted Game Design

The development of a game is not a trivial undertaking (Blow, 2004). The production of a game employs many people, from its conception, through development and to distribution. Even post launch support is a demanding job. Many professions contribute to the development of a game, from 3D artist to music composers. Computer programmers, although usually the foundation of a game, are not the only profession the development of a game relies on. Entire skill sets have emerged from the advent of game development and the game industry.

Because of this, the cost in developing games has become significant (Folmer, 2007). Even relatively small game projects, which gaming cultures refer to as Indie game (due to the circumstances of the movements inception), is an expensive undertaking for the individuals involved. There are many moving parts, not only in the game, but also in the process in which games a developed and designed, that contribute to the high cost of development.

With cost of game development so high, processes in the game design/development process can be inspected to consider how to find efficiencies. Whilst strategies such as community development (Arakji & Lang, 2007) address challenges related to time, there is also a growing interest in the use of AI based design tools and procedural generation of game content (Hendrikx, Meijer, Van Der Velden, & Iosup, 2013). If the use of AI in the design process can elevate the manpower in design/development process of games, the cost of game development can come down.

AI assisted game design (AI AGD) is concerned with development of tools that utilize AI, which can support and assist with game design process (Yannakakis & Togelius, 2015a). However, there is a lack of use of any design tools in the game industry (Nelson, 2011), let alone AI based tools. This is despite the fact that it has been noted that AI tools have the potential to not only help explore large game design spaces, but also help develop new player experiences (Eladhari, Sullivan, Smith, & McCoy, 2011). The use of AI in game design is not restricted to content, and it is important to make distinction between game content and the game itself (Togelius, Nelson, & Liapis, 2014). For example, games can be designed explicitly through the use of recombinable mechanics (Nelson & Mateas, 2008) and generative mechanics (Zook & Riedl, 2014). Such approaches allow games to be created with little or no input from the human designer. Whilst such approaches are of interest, the main area relevant to this work is where there is a tool that supports rather than replaces the designer.

Many examples of AI assisted game design can be found in the literature, for example Smith, Nelson & Mateas (2009) describe the use of AI in the play testing of levels that provide insight to the designer as to what the player experience might be. Similarly, Nelson (2011) looks at how game metrics can also be generated without players to understand game artefacts. Other approaches look at the automated generation of storyboards as a tool for the game designers to help communicate the design intent to the game developers (Pizzi, Lugrin, Whittaker, & Cavazza, 2010).

The majority of AI based design tools discussed in the literature relate to the creation of content, either independently from or in conjunction with the designer. For example, Tanagra (G. Smith, Whitehead, & Mateas, 2011) acts as mixed initiative level design tool, developed to aid a human designer in the development of game levels for a 2D platformer. Instead of automating the process of the level development, it instead helps a level designer as they are making levels for the game. The tool suggest potential designs based on the current design laid out by the designer, and can also generate the rest of the level if desired by the designers. It can run fast enough so that this functionality can be performed in real time, a key requirement if human designers is actively making the level. It does by this using concepts of Reactive Planning and Numerical Constraints in its implementation. While we will not go into detail, the system uses notions of player rhythm to map gameplay bits between the actual levels content and the system internal understanding of the game’s design.

Another example of AI Assisted Game Design is the Sentient Sketchbook (Togelius, Yannakakis, & Liapis, 2013). While this tool is also focused on assisting the level design process, it uses different methods to do so. It is also designed around a strategy game. In this case, the designer develops an abstraction of the level as a series of tiles that is later turned into a more detailed map. While the tool also suggests generated map designs, it also reports other gameplay properties of the game level back to the designer, so the designer can ascertain properties of the gameplay quickly. Generated maps are produced from a two population GA, one focusing on the novelty search and the other on constraint optimization. Novelty search is concerned with maintaining high diversity in a population, as candidate content, while not the fittest in a population, may contain information that may be of value.

Many examples of AI assisted game design address procedurally generated content. However, the intent of the approach can be used to make a distinction between approaches. Procedural Content Generation (PCG) is concerned with the development of systems that can to some degree automate the content production process. AI AGD is concerned with leveraging those systems to aid the design process. As PCG is core to the research in this thesis, the following section discusses approaches to PCG in more detail.

2.3 Procedural Content Generation

The rising cost of the game development has already been identified as an issue of concern. One of the areas of research that has significant potential to address this contemporary problem is Procedural Content Generation (PCG). In its simplest form, PCG refers to algorithmic creation of game content with limited to no user input (Togelius, Kastbjerg, Schedl, & Yannakakis, 2011). Although all game content creation is instantiated through some form of algorithm, PCG is concerned with means of creating content that are more autonomous in nature, where the users is only indirectly involved in the process.

The key term here is content, with examples in the literature of PCG being applied to all types of game content. The accepted definition of the term relative to computer games, is that all things which are contained in the game can the thought of as content. The games levels/environment (Compton & Mateas, 2006), the assets used in the game (Dragert, Kienzle, Vangheluwe, & Verbrugge, 2011), 3D models (Yoon & Kim, 2012), textures (Ebert, 2003), sounds and music (Farnell, 2007), components in the game, items, quests (Dormans, 2010), and even the structure of the content itself, the narrative of, or derived from, the game (Fernández-Vara & Thomson, 2012).

Because there are many different permutations of game systems, PCG systems are usually designed for the specific game it is being developed for, even if the underlying method can be applied universally. As such, PCG systems are usually designed around the specific design and affordances of the game. Because the design of PCG systems are specific, the content needs to be successful in the context of the game it is being developed for. This sets PCG apart from other endeavours such as generative art, as there is specific schema in which the content will be consumed by a player. As an example, a level should be expected to be finished within a reasonable time. If the generated content cannot fur fill this criteria successfully, then the method used to produce it can be deemed to not be successful.

The obvious aim for the development of PCG is to alleviate the need for manual content production. One of the most costly processes in game development is the production of game content. Automating this process would help reduce the cost of game development. This is not to say that game content producers would not be part of the game development process. The PCG system itself still needs to be designed, the content still needs to be evaluated, automatic evaluation needs to be trained or designed. The use of PCG systems in the content production process could enable human designers more affordances, and contribute to a higher quality product that can be achieved with solely manual production.

PCG, while an active area of research, have been used successfully in games over the last few decades. There is a significant body of literature related to PCG, with applications such as generating cave and maze systems (Ashlock, Lee, & McGuinness, 2011; Boggus & Crawfis, 2009; Johnson, Yannakakis, & Togelius, 2010), level maps (Togelius, Preuss, & Yannakakis, 2010a), cities (Greuter, Parker, Stewart, & Leach, 2003) and through to more conceptual issues such as dynamic difficulty adjustment (Jennings-Teats, Smith, & Wardrip-Fruin, 2010). In addition to specific applications, the literature contains a number of articles that survey the literature (De Carli, Bevilacqua, Pozzer, & Cordeiro d'Ornellas, 2011; Hendrikx, Meijer, Van Der Velden, & Iosup, 2013).

As outlined by Togelius, Shaker, & Nelson (Togelius, Shaker, & Nelson, 2016), the ideal visions for the field of Procedural Content Generation are the following:

1) Multi-level, multi-content PCG, a generator that is able to produce content of any type, in accordance with a specific game design.
2) PCG-based game design: a game in which the PCG systems are crucial to the design of a game, instead of just filling the role of content generation.
3) Generating Complete Games: instead of the PCG systems being a part of a game, or a part of a game design, PCG systems that are able to produce complete games themselves, based on a set of inputs from a designer. This includes the design of a game as well.

The above are ideal goals for what can be potentially be achieved in the field of PCG. While currently it may not be possible to achieve these goals, it is important to have goals to work towards. The properties of PCG systems that can considered desirable are following. Speed, Reliability, Controllability, Expressivity and diversity, Creativity and believability. Also, there are a series of dimensions are that are common to all permutations of PCG methods:

- Online vs Offline: When the content is generated relative to the runtime of the game
- Necessary versus optimal: Does the content need to be correct in the design of the game
- Degree and dimensions of control: how much control does the designer have over the generated content.
- Generic versus Adaptive: is the PCG method adaptive to the player
- Stochastic versus deterministic: how random vs deterministic the content generator is
- Constructive versus Generate-and-test: Is the content constructed once or over multiple iterations, or is it rejected as regenerated based on an evaluation

- Automatic generation versus mixed authorship: How much of the content is generated vs how much of it is authored by another designer.

Traditional approaches to PCG utilise a number of techniques or theoretical frameworks, such as L-systems (Dormans, 2010) or other space based approaches (Bourke & Shier, 2013), statecharts (Dragert et al., 2011) and petri nets (Lee & Cho, 2011), along with an emergence of declarative approaches (Smelik, Tutenel, de Kraker, & Bidarra, 2011) or those using techniques such as answer set programming (A. M. Smith & Mateas, 2011). However, one of the dominant trends in recent years is the growing interest in search based or evolutionary based approaches to PCG.

2.3.1 Search Based PCG

Recently the distinction between general PCG methods and Search Based PCG has been identified. While the taxonomy above covers all forms of PCG, Search Based is specific to methods that employ evolutionary computation and other metaheuristic techniques.

As defined by Togelius, Yannakakis, Stanley, & Browne (2011), PCG methods that fall under search based PCG, are distinct in that they feature an evaluation function that grades content instead of accepting or rejecting it, and proceeding instances of generated content are based of content that received good grades. These two concepts, an evaluation/fitness function and using information from well graded instances in new instances, are typical of Evolutionary Computation (EC) approaches. PCG methods that utilize EC therefore fall into this category of Search Based PCG. However, Search Based PCG is not only limited to methods that employ EC. It is noted that EC techniques are over represented in Search Based PCG (this research included).

Search Based PCG should not be confused with standard Generate and test PCG methods. Search Based PCG is a subset of Generate and test PCG, where content is graded instead of accepted/rejected.

2.3.1.1 Metaheuristic Search Algorithms

The fundamental premise of search based PCG is the utilisation of some form of search algorithm that explores the solution space automatically, through the use of an evaluation function that guides the algorithm in the discovery of new solutions. Traditional search algorithms would typically use gradient based information that requires an evaluation function that is differentiable. Such approaches are limited in their applicability, and even direct search methods like standard hill-climbing are prone to convergence in local optima (Yuret & De La Maza, 1993). Most search based PCG approaches therefore use some form of metaheuristic based global search algorithm.

A wide variety of such algorithms exists, including Genetic Algorithms (Goldberg, 1989), Simulated Annealing (Kirkpatrick, Gelatt, Vecchi, & others, 1983), Tabu Search (Glover, 1990) and Particle Swarm Optimisation (Eberhart & Kennedy, 1995) to name but a few. Various studies have been conducted that compare the relative performance of such algorithms on different problems (Connor & Shah, 2014; Elbeltagi, Hegazy, & Grierson, 2005; Eberhart & Shi, 1998; Evins, 2013; Ingber & Rosen, 1992; Connor & Shea, 2000). These studies, and many more in the literature, do not provide any insight into any consistent performance benefit that can be attributed to a particular algorithm. This suggests that the choice of algorithm is perhaps less significant than how the problem is represented for the algorithm to explore. In this study, a genetic algorithm has been utilised as the base metaheuristic. This decision was based on the ease of conceptualising the search approach, the popularity of the approach in existing PCG approaches (Mourato, dos Santos, & Birra, 2011; Ong, Saunders, Keyser, & Leggett, 2005; van der Linden, Lopes, & Bidarra, 2014) and also the possibility further exploring interactive genetic algorithms (Cardamone, Loiacono, & Lanzi, 2011; Kruse & Connor, 2016) where the fitness function is replaced through interaction with the designer.

2.3.1.2 Content Representation and Search Space

Search Based PCG systems, situated around evolutionary computation, stochastic optimization and metaheuristics, are concerned about how the content possibilities are represented. Search based PCG systems conceptualize the possibilities of potential content as being a search space. This is similar to how Genetic Algorithms conceptually search an undefined space. Instances of the content represent a point in that search space. In this context, how a point in the search space is represented is of significant interest. The representation is referred to as the genotype. The corresponding instance of content, the phenotype.

Genotypes can fall between a direct encoding or an indirect coding.
There are two extreme specifications a genotype implementation can be. Direct encodings or indirect encodings. Direct encodings are, direct. A direct representation of a maze level, could be 2D dimensional array storing the value of each tile of the maze. An indirect encoding might be a random seed, from which maze is generated. While direct encodings are computationally simple to implement, they are heavy on memory as they usually store most of the data that would be instantiated to construct the content. Indirect encodings on the other hand, would have a smaller memory footprint. However indirect encodings are very computationally specialized.

An important principle in the design of genotypes is that it has high locality. This means that small changes in the encoding results is small changes of the phenotype and thus its position in the search space. Another principle is that the encoding can represent as many interesting solutions as possible. We will call this search breadth, as a term was not ascribed by (Togelius, Yannakakis, et al., 2011).

Even though direct encodings are more memory intense, it is easier to maintain high locality and search breadth, due to being direct. It is harder to maintain high locality and search breadth in indirect encodings.

An example of different representations: (Frade, de Vega, & Cotta, 2010) evolved terrains for a game that were represented as expression trees. As an expression tree, it is clearly an indirect encoding of the content it represents. (Togelius, Preuss, & Yannakakis, 2010b) generated maps RTS. Their representation details the positions of bases and resources as X and Y coordinates. This is direct. However, other features of the maps are represented indirectly. Positions, standard deviations and heights of several two-dimensional Gaussians are evolved, and the height of the terrain at each point is calculated based on those. As such the representation contains direct elements and indirect elements. These are examples of different content representation.

2.3.1.3 Evaluation Functions

As previously described, search based PCG utilize evaluation functions is an automated approach. Like fitness functions in GAs, SB-PCG evaluation functions evaluate and grade instances of content. How to grade an instance of content and for what reasons is not a trivial question. There are many features that determine what makes content 'good', and these features vary wildly with the design of a game. Quantifying what these features are is huge undertaking, and will not be attempted here. Evaluation functions can fall into three categories:

- Direct Evaluation functions: takes direct measurements of the content, and grades the content based on specifications by a designer. Theory driven functions have specifications derived from intuition and/or some qualitative theory. Data driven on the other hand derived from recorded data about game content and their effect on player experience.
- Simulation-Based Evaluation Functions: candidate content is simulated to determine its grade. The simulation usually involves artificial agents in place of the players. Static Simulation based evaluation functions, feature agent that do not change over the course of the simulation. Dynamic simulation, features agents that do change over the course of the simulation
- Interactive evaluation Functions: incorporates real players into the evaluation process. Data about the player's response can either be recorded explicitly, through questionnaires about the game experience, or implicitly, from the monitoring the players behavioural and physiological responses.

Revisiting the two earlier examples; Frade et al. (2010) featured a direct, theory driven evaluation function that scored the content based the largest connected area of flat terrain. While (Togelius et al., 2010b) had a collection of evaluation functions, both direct and simulation based. They both are motivated by gameplay considerations, using A* in the process. As such they are also considered theory driven.

This research ultimately falls into search based procedural content generation, as a genetic algorithm that evolves game levels are developed and implemented into a game.

2.3.2 Experience-Driven PCG and user evaluation

This research undertakes user evaluation. However it is not the first instance where user evaluation of generated content is under taken. The following examples discussed here fall into the category of Experience-Driven Procedural Content Generation (Yannakakis & Togelius, 2015b). ED-PCG, as it is abbreviated to, incorporates notions of affective computing into PCG, where the emotional effect on the player is incorporated into the framework. One of the components of ED-PCG, is the Player Experience Model, a function modelling the relationship between the content and the resulting player experience. While this player model can be constructed from theory, the aim is to use data of players responses recorded from plays session. This data can be collected and used in various ways.

Raffe, Zambetta, Li, & Stanley (2015) created a system that utilized multiple evolutionary processes to generate personalized content. Players would initially play randomly generated levels. On the completion of a level, they rated the content, and then choose if they want offspring of the level to be generated or randomly generate another level. Their ratings would influence and train the player model.

Pedersen, Togelius, & Yannakakis (2010) used Infinite Mario Bros (and open source Super Mario Bros clone with generatd levels) to collect data from data from players and train a player experience model, which would then be further used in the generation of subsequent levels. Pairwise emotional preferences were recorded via a survey after an instance of gameplay.

Roberts & Chen (2015) proposed Learning-Based Procedural Content Generation, a PVG framework utilizing machine learning methods. Using an open source version of Quake, a prototype of the system was developed and a subsequent online user test was conducted. Unlike the previous examples, the content generator was trained by an expert before the user test.

While these examples may layout outside the scope of this research, they do conduct user evaluation of the generated content, which is the main similarity to this study. However, the data is used to inform the Player Experience Model. They do not feature a comparison of player response between content designed by a human and generated content. Examples of the comparisons between human content and generated content were not found. Interestingly, Pedersen, Togelius, & Yannakakis (2010) and Roberts & Chen (2015) used games that are well known in computer game culture. The users may have known the original game and may have known the nature of the study. This implication will be discussed in section 3.3.4.

2.4 Summary

This chapter has reviewed some of the literature related to game design, the use of AI in both games and the game design process, and specifically procedural content generation. It can be seen that the use of AI in games is limited because game design companies are focused on the final player experience (Bourassa & Massey, 2012) and that the process of game design is becoming both more complex and more costly (Edwards, 2006). Procedural content generation is seen as one means by which the time and cost of game development may be reduced, and it has also been noted that PCG has the potential to create new gameplay experiences (G. Smith, Gan, Othenin-Girard, & Whitehead, 2011). Given that player experience is the key driver for game development companies, there is surprisingly little research that evaluates whether procedural content generation makes an impact on player experience. The research described in this thesis makes an attempt to fill this gap.

3 Methodology

3.1 Research Objective

PCG is in active area of research in the game AI research community. However, although significant progress has been made in developing PCG methods, it is argued here that there is little empirical evidence to support whether these methods are effective. Technical feasibility seems to usually be the focus of the research. The design feasibility, in relation to player’s engagement, is not usually the focus of the research.

The aim of this research is to fill this gap by evaluating the potential of a PCG method, namely a genetic algorithm, in comparison to human designed content via an immersion rating. This is important because, to fulfil the goal of offloading the content creation process onto autonomous or semi-autonomous systems, these systems need to produce game content that is as good as or better than content designed by a human. To achieve this not only requires methods that can produce content, but also design the content to be engaging to players. Without evaluating the content produced by PCG methods, the effectiveness of the PCG method is thus unknown.

As such the research question is as follows; what affect does the inclusion of a PCG method have on the perceived immersion of a game?

3.2 Research Methodology

This study is experimental in nature, as it involves testing the immersion produced by two different game variants and utilising statistical significance testing to identify whether there is any difference in the degree of immersion experienced by the two groups of participants. This study draws upon elements commonly associated with a positivist research paradigm, in particular the dependence on quantifiable observations that lend themselves to statistical analysis.

However, this study is not strictly positivist in its nature. Ontologically the positivist paradigm is concerned with establishing causal relationships which necessarily reduces people and their behaviours to variables. In the case of this research, an attempt is made to isolate a relationship between the spatial configuration of a game and the immersion a player experiences. However, video games are not truly fixed entities and the individuality of the player cannot be ignored. There are many different types of video games, and many different ways people interact with them. How people interact and engage with a video game is unique and various wildly. To some extent, the role of the player can be controlled through managing the demographic of the participants in the study. But there are simply too many factors that influence the way a group of players could react to the same game. This study therefore combines quantitative data collection with qualitative observations of the game play experience in order to provide insight into the effectiveness of PCG. It is not expected that the outcome of the research will give a definite answer about the effectiveness of PCG. This due to multiple factors of the subject of this study.

The following section outlines the underlying experimental design that underpins this research.

3.3 Experimental Design

3.3.1 MDA Framework and the use of PCG

As discussed previously, the MDA framework (Hunicke et al., 2004) is a framework that is useful for the analysis and design of games. In this research, this framework is used to discuss how the use of a PCG method could potentially impact the immersion rating of the game.

As outlined in the previous chapter, the MDA framework breaks games and game related media down into three distinct lenses: *Mechanics*, *Dynamics* and *Aesthetics (of Play)*. Using these lenses to analyse or design games, it becomes easier to understand how the rules of the game effect the overall experience and how the different systems in game relate to each other.

Mechanics refers to the rules of the game. This includes not only the game logic, what happens when different game elements interact, but also the specific rules and maths that govern the games physics and other systems.

Dynamics refers to the runtime behaviour of the aforementioned rules. How does the physics dictate the movement of the characters? What scenario and complexity emerges from the interaction between game elements? These are examples of dynamics.

Finally, there is the Aesthetics, and this really implies a consideration of the Aesthetics of Play rather than the visual language of the game environment. In this sense, Aesthetics can be considered as the emotional response to the aforementioned dynamics. How does the movement of the player's character feel? How does the player respond to the game elements? The Aesthetics in a game also determines what type of game it is fundamentally. The cohesion between the various dynamics in the game, contributes to what the aesthetic is, and whether or not the aesthetic is communicated clearly and matches the expectations of the player, or not.

With these three lenses, it becomes easier to for game designers to design games and game researcher to understand games, as the relationship between the games systems and the game experience becomes more tangible to understand with the use of this framework.

This research evaluates the use of GA generated levels and what immersion ratings GA generated content is able achieve from such a method. Falling into the category of PCG, it would seem that the use of GA level generation would not affect the Mechanics of the game, since it is related to the generation of content and does not govern any runtime systems in the game. However, although the PCG system here governs offline content generation, it still has a tangible effect on the mechanics of the game.

It is argued that game mechanics also covers the spatial configuration of the content that the game uses as an environment. In the same sense that poorly designed mechanics would negatively impact the game as a whole, a space that is poorly designed would also negatively affect the game experience. While this can be described as bad level design, it is depended on the relationship between the spatial configuration and the design of the game.

A game space is not bad in of itself, until it is engaged with from the player via the mechanics of the game. If the game was designed to emphasize the use of a cover in a combat scenario, a space that does not have cover would be considered poorly designed. However, if the game was designed to emphasize the use of movement, strafing and dodging, a space with too much cover would slow the movement down and be detrimental to the design goal of the game. In both cases the space is not designed for the other systems of game, and contributes to a lack of cohesion of the game dynamics, which in turn, undermines the desired aesthetics of the game.

In this sense, the design of contiguous space the game uses as its environment can be considered a mechanic, as its layout has a tangible effect on dynamics of the game, and hence the aesthetics. A poorly designed space is a space that is not designed around the mechanics of the game, and will contribute to a lack of cohesion between the various game systems. On this level, the design of the space is unlikely to dramatically change the dynamics of the game, and instead contributes to the cohesion of spatial related game mechanics/dynamics. But the games contiguous space should still be thought of in these terms, because there is still a possibility that the design of the space may have a significantly change on the dynamics of the game, without lessening the cohesion of the game systems. This is one of the methods level designers are able to use to differentiate different parts of the game, and to create a variety of experiences in a single game even though the rules of the game may remain constant.

Since the GA generates levels for the game, it must be acknowledged that GA implementation would affect the mechanics of the game, even if it does not execute during the runtime of the game. In this context, this research is essentially evaluating whether or not the collection of game mechanics (which includes the spatial mechanics of GA generated levels) that construct the game, is cohesive enough to make an immersive experience. The notion of immersion will be discussed in section 3.3.2.

For this research, it is intend to evaluate how effective GA's are for content generation relative to human designed content. A comparison between the immersion of GA generated content and the immersion of human design content is needed. For this reason, this research has been designed to evaluate the immersion between levels designed by a human, and levels generated by our GA implementation.

A single game has been implemented in this study, with two variants. The game itself is a basic top down shooter. The design of the game is outlined in detail in section 4.1. The game is 2D, but is played from a top down perspective. The goal is to navigate the game's environment until the level's end is reached. Along the way there are a variety of non playing characters that are hostile to the player. The player must overcome these characters in order to progress.

There will be two variants of this game. In the first variant, the levels that the game uses, would have been designed by human level designers. In the second variant, the levels are generated by a level GA implementation. These two variants are then used in a play test experiment, where participants are assigned a version of the game to play, but not informed of which version they are playing. The details of the experiment are detailed in a later section.

As a game has been developed specifically for this research, it is stressed that comparisons between the outcomes of these two variants are only related to the game in this research and the specific PCG method used.

In the MDA schema established earlier, the game variants can be described as having most of the mechanics consistent between them, with the exception of the game environment. In the GA implementation variant, the game level is generated by a genetic algorithm. In the human designed variant, the levels are designed by a human being. The only mechanic that changes, is the spatial configuration between each variant.

This leaves us with the following potential outcomes, in regards to the design of the game. The GA game variant is more immersive than the human designed variant, meaning that the mechanics of the GA generated levels are designed well with existing mechanics of the game its self. Or the opposite, the GA game variant is less immersive than the human designed variant, meaning that the GA is unable to produce levels that have mechanics that work well with the rest of the game. Lastly there is a third outcome. The GA variant produces levels with the same immersion as the human designed levels, which implies that either GAs are able to produce levels that work well with the existing game, or humans are unable to produce levels that are more immersive that the GA.

This outcome may also imply that the mechanics of GA generated levels change the dynamics of the game without lessening the cohesion of the game dynamics, resulting in a game that is equally immersive to the player but for different reasons than the human designed game. It should be noted that it is not expected that this will be the case, due to the scale of the differences between the two game variants. As discussed earlier, the differences in the design of the space are more likely to contribute to the cohesion between game dynamics, rather than create unintended, but cohesive, game dynamics. If such an outcome is the case, it would be difficult to describe the nuances as to why.

3.3.2 Game Immersion Questionnaire

The previous section discussed that that this research is to evaluate the success of a set of game mechanics which featured a PCG method. To measure this there is a need to evaluate how successful the overall aesthetic is in the context of MDA. If the mechanics of the game are cohesive, the game's aesthetic will be sufficient enough to produce an emotional response.

However, because a video game is a form of media, interactive media in this case, this gameplay aesthetic becomes harder to quantify, especially if you consider the role the GA takes in the overall design of the game. It must also recognize that game play aesthetic is not simply a measurement of "fun". In the MDA framework, aesthetic aims to describe the type of experience/emotional response the game contains or aims for. Not all games are about the same subject, and the design of games various quite wildly. However, two games that have different designs might be equal in "fun". Note that this may be misleading to those who are not familiar with game research/culture/development. As such this thesis will refer to the "Aesthetic" as “emotional response”, as that essentially describes this component of the MDA framework.

As such, an alternative way to measure the success of a game is needed. For this research, player immersion was chosen to be evaluated to measure the success of the game variants. Throughout this research, mention of the notion of immersion has been made in relation to the success of the game. As identified by Jennett et al. (2008) “The notion of absorbing and engaging experiences is not a new concept”. The ability to identify specific qualities of a game that tell us about the success of the experience would be of great value to the game development community and game design researchers alike, as such measurements would inevitably contribute to the success of game development and game research. Game testers and game researchers desire an identification of these qualities for obvious reasons.

However, because video games are a media, and subject to interpretation, identifying such potential qualities in games are difficult. This is important to this research because the goal is to evaluate how successful a GA generated game level is in a specific set of game mechanics. Hence, a measurement is needed to accurately evaluate the game variants. Due to the scope of this research, identifying such qualities is beyond this scope, as the implementation and success of the GA is the primary concern. Because of this, a Game Immersion Questionnaire will be used as it has already been developed and available from the existing literature (Jennett et al., 2008). This questionnaire was developed for the purpose of evaluating the immersive experience of a game.

The notion of immersion itself is not well defined; despite the wide spread use of the term in the game industry and game media. While it has been identified that it does indeed occur when playing games (Brown & Cairns, 2004a; Haywood & Cairns, 2006), the specific attributes that contribute to such a state, is still in the process of the being studied. In Jennett et al.’s work, the term immersion has a specific definition, relating to other notions; Flow (and its subset, GameFlow), Cognitive abortion and Presence, as well as the ground work laid out by (Brown & Cairns, 2004a).

These ideas all overlap in areas, but are all related to the process of being engaged playing video games in some way. Flow, (and by extension GameFlow), describes the process by which "people become so absorbed in their activities that irrelevant thoughts and perceptions are screened out". GameFlow is specific interpretation of this notion for purpose of being applied to video games. Cognitive absorption (CA) is concerned with how people become deeply involved with software. Cognitive absorption is applicable to video games as games are a form of software. Presence describes the physiological state of being in a virtual environment. Since games typically involve simulated virtual worlds, presence are applicable to video games. These three notions are all applicable to the measurement of video games, however Jennett et al. (2008) identifies that these notions do not cover all aspects of games and may not be an accurate measure of the success of games. Pulling from the parts that overlap and are universally applicable between these ideas, Jennett et al. (2008) establishes immersion as a potential measure of the quality of a video game. Immersion is described by Jennett et al. as:

> "immersion clearly has links to the notion of flow and CA and all three use things like temporal dissociation and awareness of surroundings as indicators of high engagement. However, immersion is concerned with the specific, psychological experience of engaging with a computer game. This need not be the most optimal experience, often falling far below being a fulfilling experience, nor even representative of a person's general experience of playing games. Immersion rather is the prosaic experience of engaging with a videogame. The outcome of immersion may be divorced from the actual outcome of the game: people do not always play games because they want to get immersed, it is just something that happens. It does seem though from previous work that immersion is key to a good gaming experience". (Jennett et al., 2008, p. 642)

In this study, the assumption is made that a game with a relatively high immersion rating can be correlated with an effective game design. This assumption seems reasonable, as the term immersion is referenced commonly in video game media and culture, as a quality of a well-designed game.

This means that the systems that comprise the game are cohesive with each other. If this were the case with the GA game variant, it would mean that the GA implementation is successful, in the sense that it is sufficiently successful with the other systems in the game. The immersion rating of the game will be used as a proxy to evaluate the cohesion of the design of game variants. Game design in this sense refers to not only the rules and systems of the game, but the design of the content as well, which would be levels in this study. This is how the success of the GA implementation will be evaluated using a set of human designed levels as a benchmark.

The questionnaire that has been utilised from the work described is described in Appendix A, however for the purpose of the current research it was decided to not utilise the final question of the questionnaire that explicitly referenced the concept of immersion using a 1-10 scale, which differed from the other questions that are based on five point Likert scales. The version of the questionnaire used in this research therefore consisted of 30 questions. Those 30 questions evaluate the following qualities of the player's experience; (e.g. To what extent did you feel you were focused on the game?), temporal dissociation (e.g . To what extent did you lose track of time?), transportation (e.g . To what extent was your sense of being in the game environment stronger than your sense of being in the real world?), challenge (Did you find the task too difficult?), emotional involvement (Did you care if you won or lost?), and enjoyment (Did you like the task?). The immersion score/rating, is calculated by the sum of the all the answers.

3.3.3 Hypothesis

The play test experiment conducted by this research will only use one measure. The immersion rating from the questionnaire. Hence there will be a set of immersion ratings from the participants. However, since the play testers are unknowingly assigned one of two variants, there will be two sets of data. The first is the immersion rating of the human designed game variant, where the levels featured in the game were designed by a human level designer. The second is the immersion rating of the GA game variant. This variant uses levels that have been generated in part by the

Genetic Algorithm. This will be outlined in the implementation section. Accordingly, the null hypothesis is as follows; the immersion ratings between the two game variants will not be significantly different.

Either the GA is able to produce levels of sufficient immersion to match the human design levels, or humans are unable to create levels that are more immersive than GA generated levels. Or, due to the mechanics of the game, the difference in quality between human designed levels and GA generated levels does not significantly impact immersion rating/score of the game. The alternative hypothesis would be; the will be a significant difference between the immersion ratings of the two game variants.

In this hypothesis there are two outcomes. The GA game variant receives a rating higher than the human designed variant. The human designed variant receives an immersion rating higher than GA game variant. The implications of these outcomes are less ambiguous. If the human variant has a higher immersion rating then the GA variant, then the GA implementation is not able to produce levels have similar or higher quality than human design levels, and future work would require notions off good design incorporated into the GA implementation. Or the reverse to true, the GA is able to produce levels of a higher quality, which puts into question the need for human level designers.

The one-way analysis of variance (ANOVA) is used to determine whether there are any significant differences between the means of the two groups. This approach is common in game immersion studies (Denisova & Cairns, 2015) and the one-way ANOVA is considered a robust test against the normality assumption, which ensures that it performs reasonably when data is non-normal (skewed or kurtotic distributions).

3.3.4 Experimental Structure

As described earlier, to evaluate the immersion ratings of the game variants, game play tests were conducted with participants. This was used to generate data needed to assess the success of the GA implementation against human designed levels. In total, twenty participants were recruited to take part in the study.

Although there is no restriction on who was eligible to participate, participants who are familiar with video games were targeted in the recruitment process. This was to counteract the correlation between peoples previous experience with video games and their current immersion with video games, which has been identified in the research literature (Brown & Cairns, 2004a).

Participants were recruited from the local game development community, as well as game design students at AUT University, as the demographics of these communities are familiar with games. An announcement was made informing people that the study was being undertaken, and individuals who interested in participating, after approaching the announcer, was given a participation form. Individuals whom filled out the form and indicated that they were interested, were coordinated with to organize a time for a play test.

In informing potential participants about the study, the true nature of the study was not revealed to the participants. This was to avoid participant's potential expectations adding noise to the data, or biasing the results. The rational for doing this is that, it has been identified that a player's pre-existing expectation can influence the players actual experience of a game (Denisova & Cairns, 2015).

Denisova & Cairns investigated two groups of the people playing a game called "Don't Starve", a survival themed game. The first group of participants were told the game featured "Adaptive AI", were in truth the game did not. While the second group acted as the control and were not given this fiction before playing the game. The results from this study showed that the first group had a higher immersion rating. This demonstrates that that the player's pre-existing expectations does indeed influence the experience they have.

For this study, this outcome was not desired as the game does indeed involve an AI related system for the generation of levels. If players knew that they were playing a game that featured levels generated by a GA, it would potentially bias their experience. Instead, participants were simply told that the study was concerned with evaluating game dynamics, and what game mechanics lead to what dynamics. Although it was somewhat based on the MDA framework, it is vague enough to avoid seeding specific expectations about the game experience.

The play test experiment itself was carried out in a dedicated game room. The game was displayed on a living room television that is indicative of a casual gaming environment. The participants were seated on a couch in front of the TV. The participants were given an Xbox One Controller to use to play the game. This was chosen as it was easy to implement support for this controller and its design is standard and well known in the game industry.

When participants arrived for the play test, they first had to fill out a prequestionnaire. This questionnaire was designed to generate demographic data about the participant. As well as age and gender, the questionnaire asked questions about people experience with video games and their gaming habits. This was important because it has been identified that people's previous experience with games shapes their current game experience (Brown & Cairns, 2004a), thus it was important to record this with their immersion ratings.

Once the prequestionnaire was completed, the participant was seated in front of the TV that displays the game, and was handed the controller. The researcher then runs the game. Since the game runs off a computer instead of dedicated game console there is brief window of time where the player is able to see the operating system and the shortcuts used to run each game variant. To maintain the illusion that the play test is not related to PCG and/or EC, each shortcut is titled theta and zeta from the Greek alphabet. Only the researcher knows which shortcut corresponds to which game variant, and the wording of the shortcuts implies that they are simply different versions from the development process, (alpha and beta a used to describe different phases of game development progress) while far enough down the alphabet to not immediately recognised by most. The procedure of determining which game variant was used is simple. Every second participant plays the human designed version.

The researcher explained that if there was any uncertainty while playing the game, the participant were allowed to ask questions (although over the course of the entire study only 2 to 3 question were asked, which is a good indicated as it means that the game conveyed enough information on its own). The run time of the play test was limited to around 30 minutes, although if the participant finished all levels before that the play test would finish. The first level in the game would always be an in game tutorial to explain how to play the game. Participants were advised to pay attention to the tutorial to ensure that they are able to play the game properly, and not at noise to the data.

Once the play test was finished, the participant would then fill out the Game Immersion Questionnaire described in section 3.3.2.

4 Implementation

To evaluate the hypothesis, a PCG method and an accompanying game have to both be instantiated to be used in the experiment. This chapter cover the implementation of both the game used in this research and the GA PCG method developed to test.

4.1 Vektor

The game developed for the thesis is a simple 2D top down shooter similar in appearance to the game Hotline Miami (Dennaton Games, 2012), where the player must navigate a labyrinth and find the end of the level. Along the way, there are a variety of NPCs (Non-Player Characters) that engage the player in a hostile manner. The player must successfully evade and/or overcome these hostile NPC's and find the end of the level after which they will proceed to the next level.

The game was designed to be simple and straight forward while indicative of a basic game one might find on the market. While game design is always changing due to trends in the video game industry, for the purpose of this thesis, a standard game design was all that was necessary.

The game, named Vektor, was not given any fiction, or narrative to contextualise the gameplay, as, for the purpose of the research, only immersion derived from gameplay and the GA generated content's effect on the aforementioned gameplay is important to the hypothesis. Any possible effect of immersion from the design of a narrative or the visual aesthetic is not desired, as it could potentially interfere with the immersion of the game and the variants. Because of this, the game can be described as an abstract game, where the use of basic shapes and symbols are used to communicate the purpose of gameplay elements, without invoking any emotional relationships between entities in the game, relative to peoples past experiences. While this design itself would have an effect on immersion, any configuration of the game design would effect immersion. This design ensures, that as many aspects between the two variants, apart from level topology, remain consistent and that the only differences that may effect immersion are due to the different methods in the development of the variants, human designed levels and GA generated levels.

A screenshot of the game is shown in Figure 1.

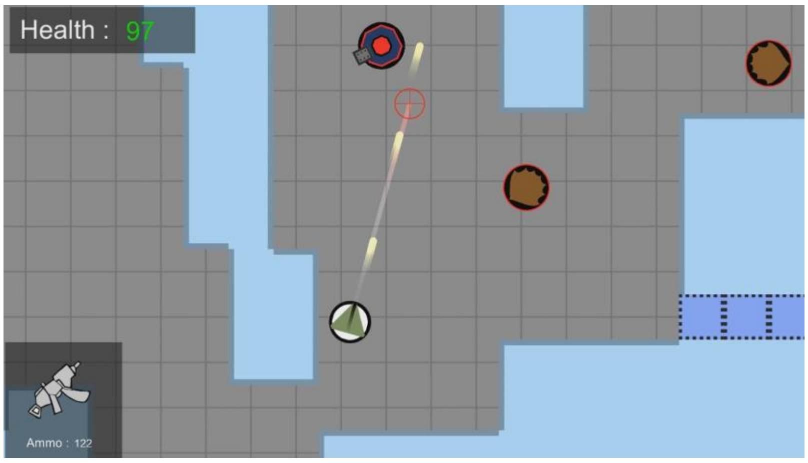


Figure 1: Screenshot of the game Vektor

4.1.1 Objective of the game

As mentioned above, the objective of the game is to find the goal of each level. While it can be argued that this would constitute a narrative, there is no explicit narrative as to why the player must find the goal. Most games have an explicit objective of some kind. Narrative derived from gameplay cannot be counteracted, and is not of concern. The player starts at a predetermined location, usually chosen from the level designer. When the player reaches the goal, the game proceeds to the next level, where the player, again must find the goal.

Although the objective of the game is to arrive at the levels goal, the path from the start location to the goal location it is not guaranteed to always initially be reachable. In almost all instances of the game, the path from the start location to the goal location is obstructed by colour coded doors, which can only be opened by the corresponding coloured key. The player must find these keys, which, on acquisition, allows the corresponding doors to be opened. This gameplay element was chosen to encourage the players to explore the space and backtrack once they have found the key, so that the game has more opportunities to allow encounters with hostile NPCs and pickups. It is also an established gameplay mechanic, as well received games such as Wolfenstein 3D, Doom and Quake (all of which were well received) employed this mechanic in their single player components (ID software, 1992; ID software, 1993; ID software, 1996).

This mechanic is also well established in level design, although it will not always be represented as finding keys of the same colour (a lot of new games use a scenario instead of the collection of a corresponding key as the motivation of the player, however the gameplay is still the same, an area is blocked off from the player and the player must reach/find something to enable access to that area).

Hostile NPC are also distributed throughout the levels in the game. NPCs are characters that act as a gameplay element. Hostile NPCs are characters that engage the player in a hostile manner. Because of this they act as obstacles that the player must overcome. There are a variety of NPC types, each of whom has unique behaviour, differentiating the challenge in different parts of the level. They will be explained later in this section. When hostile NPCs are successful in performing a hostile action (when a projectile actually hits the player), the player's health is decreased by the appropriate amount. If the player's health falls below 0, the game is reset and the player has to start again at the beginning of the level they failed in.

4.1.2 Player

The player has a standard set of abilities that one would expect from most games. Because the game is a 2D Top Sown Shooter, the player can move in any direction, up, down, left, right, and any variation in-between. The view of the game is a top down perspective, as if the virtual camera is held up by a crane looking down. To reinforce the lack of narrative, the player is represented using an abstract icon as shown in Figure 2

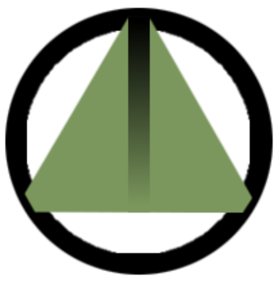


Figure 2: Image of the player in the game

As well as being able to move in any direction the player can focus their direction at any angle, regardless of the player's current movement direction. The focus direction (aim direction) will determine the direction that projectiles are fired at. If the player is not actively aiming (if the right stick of the controller, commonly used for aiming in games, is not being used), the aim direction is by default focused in the direction the player is moving. The player also has the ability to lock their aim onto a hostile NPC (provided the NPC is within a certain range), allowing them to focus on strafing and avoiding fire without having to aim. Although on paper, this addition sounds like it would compromise the design of the game, in practice it was a necessary feature to ensure the accessibility of the game. This point will be discussed in section 6.1.3.

The player is able to fire projectiles in the aim direction, the player's primary method of overcoming the hostile NPCs. The behaviour of the projectiles is determined by the active weapon the player has selected at the time the player chooses to fire. The player has access to three types of weapons, provided the player has picked them up and they have enough ammo for the selected weapon.

- Machine Gun: A standard weapon that continuously fires individual projectiles, with a fast fire rate.
- Shotgun: This weapon fires a spread of projectiles at the same time. However, the projectiles have a shorter range than the weapon described above.
- Laser: Will charge when the player fires and will continue to charge as long as the player is holding the fire button. After a given amount of the time, the laser will fire, dealing a higher amount of damage as the weapons above. However, due to the fire rate and the charging time, it is harder for the player to successfully connect shots with the desired target.

The icons used to represent these weapons are shown in Figure 3, and their firing patterns shown in Figure 4 with the top pattern being the machine gun, the middle pattern the shotgun and the lower pattern the laser.

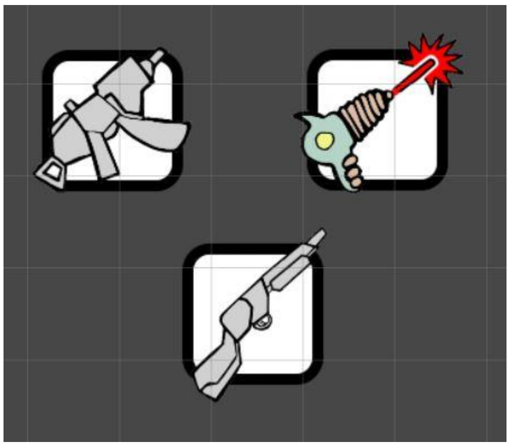


Figure 3: Image of weapons from the game

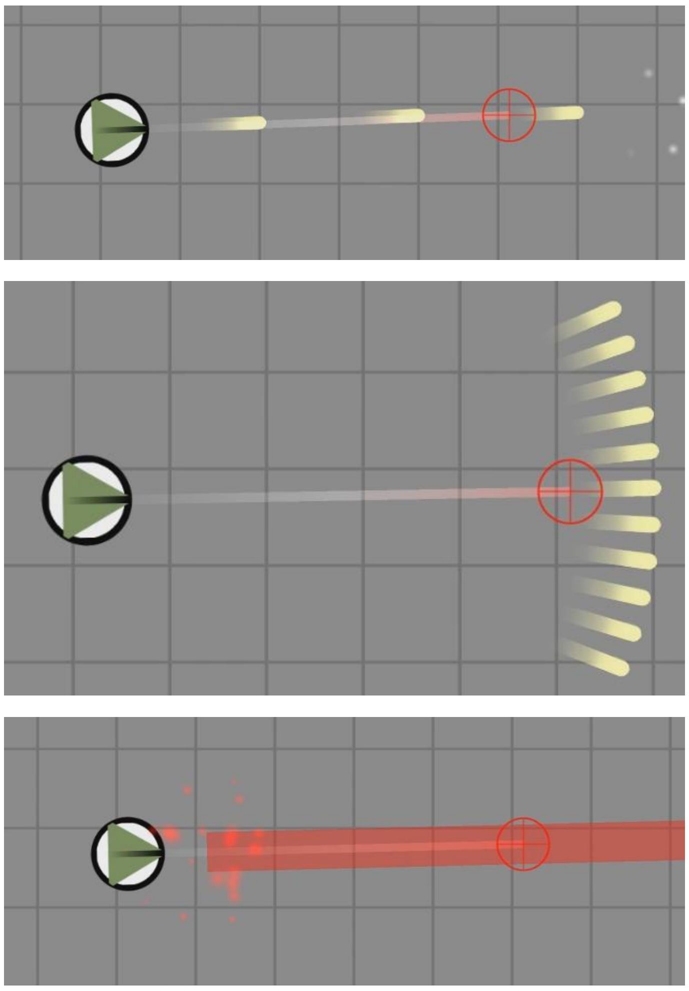


Figure 4: Examples of different Weapons being fired

As well the as the ability to move and the ability to fire projectiles, there are a variety of pickups the player can interact with. The set of pickup types are relatively straight forward and standard in most games. Pickups augment the properties of the player in some way. Colour coded keys mentioned above are the first example of pickups, allowing the player to progress though the corresponding coloured doors. There are also weapon pickups, enabling the use of the corresponding weapon for the remainder of the level. Ammunition pickups increase the ammunition count for the corresponding weapon. Finally, Health pickups restore a portion of the player's health. The icons used for the various keys and ammunition pickups are shown in Figure 5.

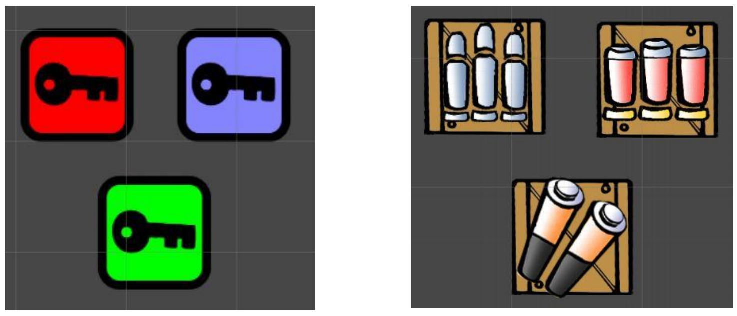


Figure 5: Images of Key and Ammo pickups in the game

4.1.3 Hostile NPCs

There are variety of hostile NPC that are distributed throughout a level. As discussed previously, NPCs are characters that act as a gameplay element, while not being controlled by other human players. Hostile NPCs are characters that engage the player in a hostile manner, like moving toward the player to perform a hostile action or firing projectiles at the player. Because of this they act as obstacles that the player must overcome, either though evasion or taking out the Hostile NPCs with the players own weapons. There are a variety of NPC types, each of whom has unique behaviour, and each type has a subset of different variants. There are five NPC types.

1) Zombie: As the name suggest, this type behaves like a zombie. When the player is perceived, the NPC will move towards the player, as if it was a zombie. When the zombie collides with the player, it will apply damage to the player's health and it will immediately die. The zombie also has the least amount of health compared to the other types. Names of hostile NPC are not shown in the game. This is named zombie simply to describe the behaviour during the development of the research and the game.
2) Charger: This type is similar to the zombie, in the sense that when it perceives the player, it will move towards the player. However, once it is at a certain range to the player, it will begin a "charge". It will move at a faster rate,

however only in a straight line and only for a certain amount of time, after which it will wait a moment and then turn around and repeat its behaviour. If the player is hit while the NPC is in the Charge state, damage will be dealt to the player. Depending the variant of the Charger, the NPC will only be vulnerable while in the charge state.

3) Turret: As the name suggest, this type acts as a turret. It will stay still and shoot at the player while the player is in range. Depending on the variant, the NPC may fire a machine gun or a laser weapon.
4) Moving Turret: This type shares its behaviour with the type above, with the exception that if the player moves out of range or out of sight, it will pursue the player until the player is in range/sight again.
5) Trooper: The type is designed to be most difficult to overcome. When it perceives the player, it will pursue the player until in range. Once in range, it will fire its weapon, and also strafe around the player, making it difficult to hit. The trooper is essentially the conceptual equivalent to the player. Depending on the variant, the trooper will either be using a machine gun or shotgun, and the strafe speeds and timings will be varied.

The various enemies are shown in Figure 6, with the types being ordered from left to right following the number sequence from the above list.

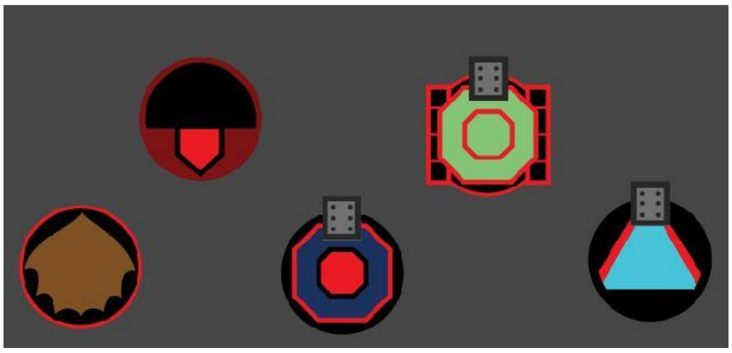


Figure 6: Image of enemies in the game

The behaviours outlined above are controlled by behaviour trees (Simpson, 2014), a well-established algorithm for controlling characters in games in the industry.

4.1.4 Game Design and PCG evaluation

The design described above was deliberately developed to aid in researching the main hypothesis of this research. While it may not sound like the game's design may have a tangible impact of the success of the research, the relationship between the design of a game and PCG systems is in truth important. As it has been established in chapter 3, this research evaluates what immersion ratings emerge from a set of game mechanics that include a GA generated levels.

For this research to be effective, the design of the game mechanics have to enable the design of content produced by a PCG system to influence immersion. This would allow the hypothesis stated in chapter 3 to be tested to determine whether there is a significant difference between the immersion ratings of the two game variants. Again, on paper, this may not sound important, as one may argue that regardless of the specific design, there would always be reliance on the design of the content. However, it is conceivable for the design of the game mechanics to alleviate the impact of the design of the content, which is generated by a GA generated levels in this research.

In the interaction between the player and the game, it is conceivable that there are three things that can influence the perceived immersion. The first is the player. The immersion due to the players' previous experiences and expectations is hard to minimize. The second and third is the mechanics of the game and the content of the game. The mechanics can have an effect on the immersion of the game and the content can have an effect on the immersion of the game. It is conceivable that the mechanics may create more immersion than the content, or the opposite, the content creates more immersion then the mechanics. It was discussed that design of content can be considered a mechanic of the game in section 3.3.1 as its design effects the dynamics. As such the various mechanics of game can create more immersion then the mechanics of the content. If the content is poorly designed, the mechanics can offset this. If the game mechanics are poorly designed, then the well-designed content can offset this as well. Consider the following example.

Diablo (Blizzard North, 1996), is an action Role Playing Games where the player explores a dungeon filled with hostile monsters and features randomly generated dungeons. However, the combat mechanics of the game significantly lessens the impact of the contiguous space of the dungeons on the active game play. In the game, when the player clicks on enemy characters, the player character will walk over to the enemy and the play an attack animation. At the same time, the health property of the enemy will diminish in accordance with the player's characters' prowess. The player cannot dodge out of the way of arrows, or parry blows, or jump off walls for a special attack.

In this sense the combat is representational, which is common in Role Playing Games. In a single combat encounter, the contiguous space of the environment does not have a meaningful impact on the game as the combat mechanics of the game do not leverage the environment. In the case of Diablo, the mechanics elevates the need for carefully thought spatial configuration. This is not to say that levels do not have an impact on the game, because other properties of the game, like the pacing would potentially impact the games immersion. However, we need to be aware of this relationship between the design of the game, and the content for which whatever PCG systems are used to produce it. A PCG system could in theory generate option content, and the game design could completely alleviate the need for it to be designed well.

For this research, game levels have been generated through the use of a genetic algorithm. Because the PCG system deals with levels, whether or not it is successful is based on the spatial configuration of the level that is produced. Hence, we need a game design where the contiguous space has a tangible effect on the immersion rating of the game. For this reason, a top down shooter game was implemented, as the mechanics of firing and avoiding projectiles necessitates the use of cover and having enough room to evade.

The perspective of the Top Down Shooter also simplifies the design space for this research while maintaining the complexity required to accurately test the hypothesis. In contrast, had a 2D platformer game been chosen, pathfinding that takes into account distances between platforms would have to be implemented, an unnecessary complexity. Having the game world directed horizontally along the screen would make the structure of the GA to specific to a specific game design. Similarly, a 3D first person shooter would increase the complexity of the game to the point where it would be too big for the scope of the research, as game development is very time consuming and skill intensive practice.

4.1.5 Vektor and MDA

While MDA is situated as an iterative approach, it still provides an effective schema for the discussion of games. As Vektor was designed for the evaluation of players' perceived immersion, its design needed allow the interaction between the content and the player to an extent. As the PCG method produces game levels, the design needs to require that the spatial configuration of the levels effects the immersion. This is why a Top down shooter design was chosen. The mechanics of firing weapons, avoiding projects, avoiding enemies, is intended to put emphasis on the use of the environment and movement though that environment. The dynamics hence, is to emphasize movement, either out of the way of projectiles, around an enemy, or behind cover. This should, in theory, put emphasis on the layout of the space, whether there is enough space or enough cover.

4.2 Genetic Algorithm Implementation

Although the concepts behind a genetic algorithm are general and applies to serval potential search spaces, often a specific implementation is required for the search space at hand. This research is no exception and involves the implementation of the fitness function and mutation function that are specialized for the search space defined by the goal of creating playable levels. It is common for custom implementations of various functions in GA to be devised. However, it was found that using a standard crossover function was actually viable for the study. This section will detail the implementation of the GA used to generate levels for the GA variant.

4.2.1 Chromosome Representation

All Genetic Algorithms start with generation of a population. Each "individual" of this population represents a position/solution in undefined search space. We refer to them as a Chromosome, a packet containing information that specifies a solution. Search spaces are not always literally defined as a search space. Chromosomes are usually not a literal x and y coordinate.

The search space consists of all possible configurations of a game level. In Vektor a level is comprised of a series of tiles, arranged in a grid along a rectangular shape. Such a level, with additional gameplay elements added, is shown in Figure 7.

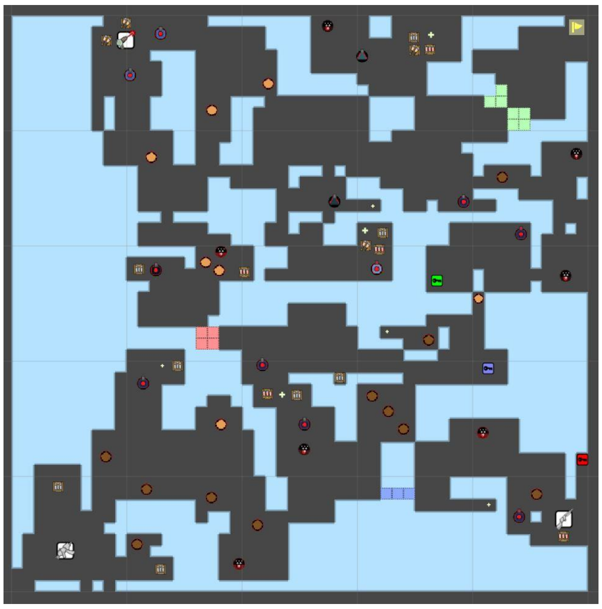


Figure 7: Level generated by a genetic algorithm

A tile can either be a wall, shown as a blue square, or a void, which represents an open space. Although contemporary games have levels designed as set pieces, where the complexity of the contiguous space is parallel to the real world and the level is implemented as another art assert/3D model, older games usually resorted to a tile system to store the level data. This had the benefit of storing large worlds in small amount of space, and enabled developers to reuse environment assets without breaking player immersion at the time. There are contemporary games that use tile systems, but these games are rare in the triple A game industry as they do not usually satisfy players expectations of triple A games. More typically, such games use more complex, object based representations. A tile based implementation was chosen for this research for the reason stated above, and because it simplified the search space. As such the chromosome consists of;

- A one dimensional set of Booleans. This set would contain all the values for the tiles in the level.

- Two integer variables, width and height, which is used to interpret the 1D set above as 2D grid of cells. Levels are not limited to perfectly squared dimensions.

This chromosome is mapped to a level by reading the values of the tile set, and instantiating walls when the wall value is found. Void tiles need not to be instantiated as they are unoccupied. The start of the level is always assumed to be at the start of the tile set, and the end of the level is always assumed to be at the end. Therefore, the start of the level is always in the bottom left of the level and the end is always at the top right. As the beginning and end is always assumed, the chromosome is not required to store that information. An example of this chromosome representation and the resulting level is shown in Figure 8.

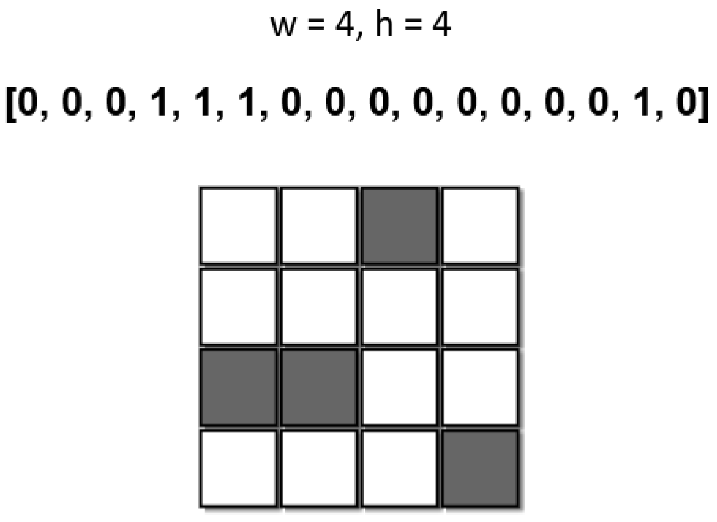


Figure 8: An example of the mapping between the chromosome and a level

4.2.2 Population Generation

In order for the Genetic Algorithm to function, it needs an initial population to act on. The generation of that initial population is just as important as the other operations in the GA. The implementation in this thesis uses a specific population generation function.

Initially we experimented with having each cell of the level randomized to either a wall or a space/void with a 0.5 probability. However, this proved to create levels that were too chaotic. Even though the levels were technically playable (you could reach the end from the start), there were features that would severely compromise the clarity of the research. Almost all pathways the players could move through was limited to width of 1 cell, meaning that the pathway would only be as wide as the player. There would be many divergent paths, which ultimately lead to a dead end. While multiple paths and a degree of divergence in level design could be identified as features that players might want in today games, the reality is such diverged qualities of a level would be desired with moderation by the designers. Using random level generation, there was simply too much divergence. It appeared to be hard to navigate which produces a different experience in terms of flow. This can be seen as a potential flaw with the fitness function, which will be detailed in a following section. What will be said here, is that defining a fitness function that would take completely a random generated level, and making it playable and navigable would be significantly complex. To maintain simplicity this responsibility is shared by the generation function.

For the reasons above, it was decided to devise another generation function. Instead of randomly assigning each cell to a wall or a void, generating a series of rectangles with a random position and size inside the bounds the level, in accordance to the levels dimensions and the desired aspect ratio between reachable space the rest of the level, was found to be a more stable and controllable method. The cells in the level that lay within the rectangle were set to void, while the rest of the cells are be set to a wall. This generation function is referred to as the “falling rectangle” function, as it analogues to throwing rectangular shapes into a space, and using their resting place to devise the configuration of the level. The operation of the falling rectangle function can be controlled by a user, with the following series of variables;

- P Population: the initial population of levels generated.
- w_{rmin} Rectangle Minimum Width: this is the minimum width in level cells/tiles a rectangle can be instantiated at.
- h_{rmin} Rectangle Minimum Height: this is the minimum height in level cells/tiles a rectangle can be instantiated at.
- w_{rmax} Rectangle Maximum Width: this is the maximum width in level cells/tiles a rectangle can be instantiated at.

- h_{rmax} Rectangle Maximum Height: this is the maximum height in level cells/tiles a rectangle can be instantiated at.
- R Desired Space Ratio: this is a percentage representing the desired ratio between the space that can be reached from the begging of the level, and the total cell/tile of the level as a whole. This will also be discussed in the fitness function.
- O Rectangle Overlap Percent: this is a percent that represents the addition rectangles that will be added to the amount of rectangles that is used to generate the level. This will be explained shortly.

These variables are used in an equation which determines how many rectangles are used in the generation of a level. The user who is instantiating the GA can change the values of these variables to control the outcome of the level generation. The following equation is used to calculate the number of rectangles used (n_r):

$$n_r = \left(\frac{n_c \times R}{a_p}\right) \times (1 + O)$$

Equation 1

R And O are the Desired Space Ratio and Overlap Percent described above respectfully, however the two new variables, n_c and a_p have yet to be described. These are defined as:

- n_c is the total number of cells in the level, where $n_c = width \times height$
- a_p is the projected average of rectangles in cell/tiles.

Before the rectangles are generated, the average size of rectangles is not known, but it can be guessed. The following equation is used to calculate the projected average size of rectangles:

$$a_p = \left(\frac{w_{rmax}}{2}\right) \times \left(\frac{h_{rmax}}{2}\right)$$

Equation 2

Note that overlap percent is increased by one $(1 + O)$ and then multiplies the rest of the equation $\left(\frac{n_c \times R}{a_p}\right)$. This is because its role is to add additional rectangles to the level generation process. The following pseudo code describes the process in which levels a generated from the falling rectangle function.

```
$n_r = \left(\frac{n_c \times R}{a_p}\right) \times (1 + O)$ // we calculate the number of rectangles
Create Set population // create the set for the population

loop $P$ times // we run the loop for $P$ times
    Create level with width and height
    Initialize all level.cells to WALL

    loop $n_r$ times // we run the loop for the $n_r$ times

        Create rectangle, with RandomX in bounds of level
                               RandomY in bounds of level
                               RandomWidth between $w_{rmin}$ and $w_{rmax}$
                               RandomHeight between $h_{rmin}$ and $h_{rmax}$
        loop for each (level.cell in bounds of rectangle)
            level.cell = VOID
    add level to population set
```

For the final implementation of the GA, the generation function described above was used to generate the initial levels in each population. This population would then be subject to other operations that are used in a GA, fitness evaluation, selection, crossover, and mutation.

4.2.3 Fitness Function and Selection

4.2.3.1 Fitness Function

The fitness function that was devised in the research operates on the assumption that a good level has a specific ratio between space that can be traversed and the total size of the level. Also, a level that can be completed, the player can reach the end from the beginning, is assumed to be necessary. With these two criteria; the space ratio, and the playability of the level as a whole, a fitness function is able to score levels appropriately. Using this fitness function it was possible to generate a series of levels that were all playable and had interesting features. This section details the implementation of the fitness function.

The first criteria, the space ratio, was briefly described in the previous section. It is the ratio between cells/tiles that are reachable from the beginning of the level, and the total amount of the cells the level contains. The total amount of cells is calculated from width and the height (width * height). This level space ratio is compared against the desired space ratio, which is the between ratio between the space and the level size that is desired by the user instantiating the GA. The difference between the space ratio and the desired ratio gives us the ratio error, which we can use to manipulate the final score. The math for this will be discussed later in this section.

The second criterion is the playablility of the level. If a player can reach the end of the level from the beginning the level is deemed playable. Although this also true for the random level generate described previously, the ability to navigate and the find the end is also a quality of the playability level. However, as described, this functionality is shared by both the fitness function and the generation function.

Levels that are playable are deemed better then levels that are not playable. However, just because a level is not playable, does not mean it should be discarded. While it may appear that levels are that are not playable should be discarded, doing this may prevent the population from arriving at a potential solution. It is not guaranteed from the falling rectangle function described above that a level is playable when initially generated. However, unplayable levels may already hold information that could enable it be playable in a future generation. So if all the levels in a generation are not playable, it is necessary to identify features that show that a level may be closer to being playable than others.

This is where most of the complexity of the fitness function rests, in scoring levels that are not yet playable. While the space ratio error is easy to calculate, the second criteria is more complex to calculate, as it involves searching the levels space multiple times to conduct measurements of a levels features.

The first step in doing this, is to identify the “main” space of the level. Here we are defining a space as a contiguous arrangement of void/unoccupied cells. The main space, is the largest contiguous arrangement in the level as shown in Figure 9.

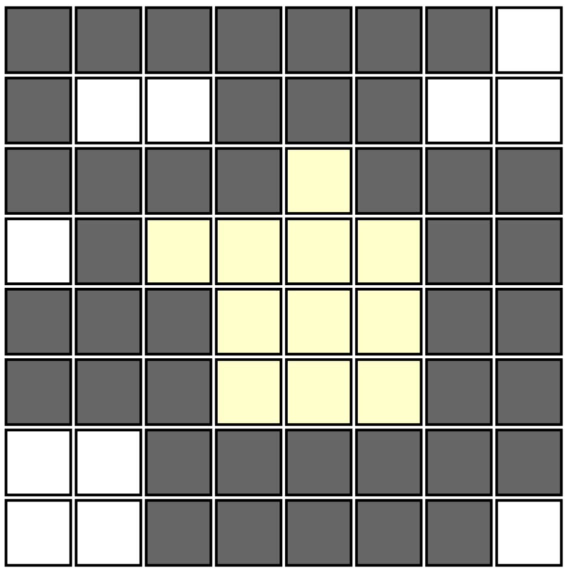


Figure 9: Level with main space highlighted

This is accomplished by using a part of Dijkstra pathfinding algorithm. While Dijkstra pathfinding is designed to find a path from point A to point B along a series of nodes, the search strategy to accomplish this, adding each adjacent node to a list representing searched space, allows a list of nodes of a space to be build. Most measurements about the level a focused on features of the main space. Other spaces are ignored. Devising a fitness function that would evaluate properties of all the spaces would be too complex, or at least unnecessary and beyond the scope of this research. However, other spaces could become part of the main space in later generations from crossover and mutation.

The space ratio error is based off the size of the main space to the rest of the level. This is because the main space is being treated as the space that is expected to be traversed by a potential player. In early generations, the main space is not guaranteed to be reached from the beginning. However, this is accounted for in other parts of the fitness function.

After we have the main space, we then identify the start space and end space. These are the spaces that contains the start and end locations. This gives us three spaces; the main space, the start space and the end space. This is shown in Figure 10.

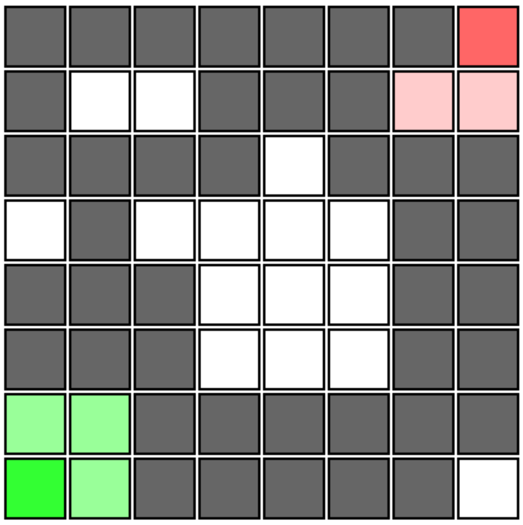


Figure 10: Level with the start (green) and end (red) spaces highlighted

Then we calculate the longest possible distance that is possible in the level based on the height and width of the level, as shown in Figure 11. This measurement is assigned to the term d_m. This is demonstrated in Figure 11.

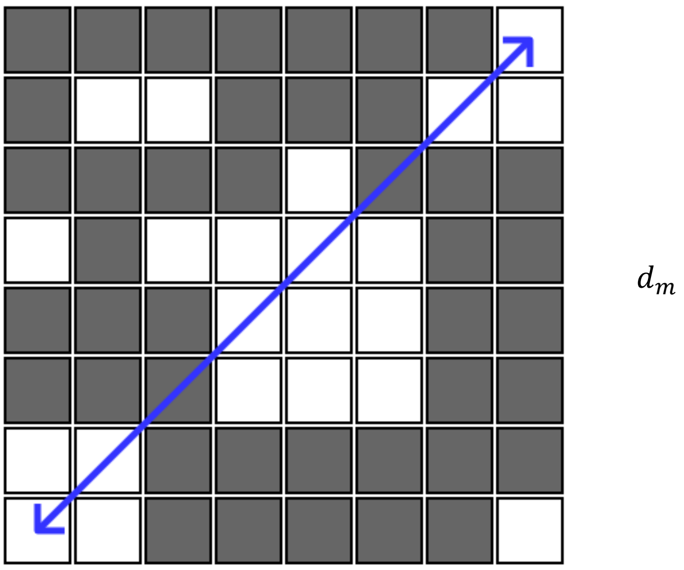


Figure 11: The levels longest possible distance

After this we then calculate how close the main space is to the start d_s and end d_e. This is done by evaluating the Euclidean distance from each cell in the main space to the start/end cell as illustrated in Figure 12. The shortest distances that are found are assigned to the terms.

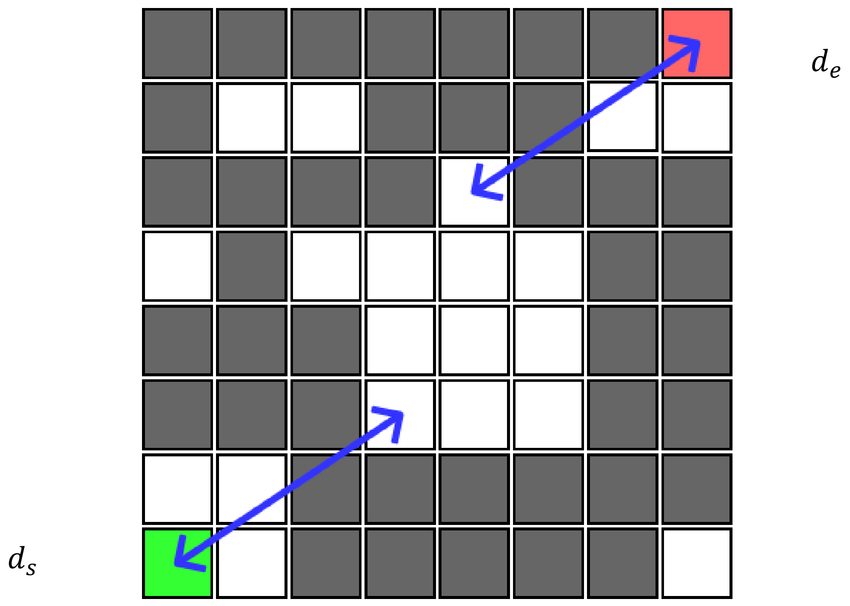


Figure 12: The shortest distance between the start/end and the main space

Once we have these spaces identified, we can start making measurements of the level. Each measurement is assigned to a unique score. After all the measurements are taken, their scores are added together in the following equation.

$$s_f = s_{ss} + s_{es} + s_{sd} + s_{ed} + s_w$$

Equation 3

Each of the terms in the equation above corresponds to a specific measurement about the relationships between the level, main space, start space and end space. Each term is calculated by a different equation or is assigned a value if a condition is meet. Each of these terms are defined briefly as:

- s_{ss} is the score of the start space. The start space score is how close the start space is to connecting to the main space.
- s_{es} is the score of the end space. The end score is the score of how close the end is to connecting with main space. This is the inverse of the s_{ss}
- s_w is the score of the whole space. Unlike the above, this is not assigned to an equation. Instead, depending a condition, it will be assigned either 0 or 100
- s_{sd} is the score of how close the main space is to the start space
- s_{ed} is the score of how close the main space is to the end space

The descriptions above are a brief overview of what each term represents. However, the equations that are used for these terms are more complex. Below is a detailed description of how these terms are assigned.

Both s_{ss} and s_{es} are assigned from similar equations. However, before they are assigned, a condition is checked. That condition is weather the main space contains the start or end, respectively. If that condition is true, then the terms are assigned 30 instead of the result of the equation. The following equations are used in the event that the condition is false. Note that the condition is checked independently for each space.

$$s_{ss} \;=\; 20 \times (\frac{d_{fs}}{d_s} \Lambda 1)^2$$

Equation 4

$$s_{es} \;=\; 20 \times (\frac{d_{fe}}{d_e} \Lambda 1)^2$$

Equation 5

The new terms in these equations is d_{fs} and d_{fe}. These are the furthest Euclidian distances found in each space. Figure 13 demonstrates this in the start space, where d_{fs} is assigned. d_{fe} is assigned similarity, but from the end space instead.

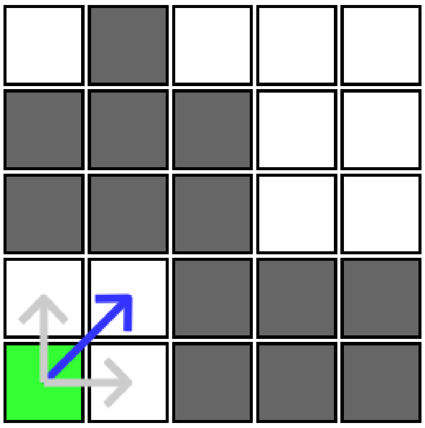


Figure 13: Portion of level, Start Space search, blue arrow is the longest distance in the space, d_{fs}

s_w represents the score of the whole space. This term is not assigned by an equation. Instead, it is either assigned 100 if the main space contains the start node and the end node. If that condition is false, then it is assigned to 0. This was devised to elevate levels that have both start and end reachable from the main space, and hence are playable.

$$s_w = 100 \text{ } or \text{ } s_w = 0$$

Equation 6

The last two terms, $s_{sd}$and s_{ed}represent the scores of the how close the main space is to being connected to the start space and end space respectively. They are the scores of the distances from the start/end space to the main space. The following equations describe how they are calculated.

$$s_{sd} = 20 \times \left(1 - \frac{d_s}{d_m}\right)^2$$

Equation 7

$$s_{ed} = 20 \times \left(1 - \frac{d_e}{d_m}\right)^2$$

Equation 8

Although the terms d_s and d_e describe the shortest distance between the main and start/end respectively, s_{sd} and s_{ed} are the scores of those distances, As such, they are not direct. If the main space already contains the start and/or end, this equation will be maximized.

Once the terms described above are calculated, they are added together with the simple Equation 3. The resulting term, s_f is the final score before being multiplied by the space ratio error. Once the final score is calculated we then progress to the space ratio error. First we must calculate the actual space ratio (r_a) of the level.

$$r_a = \frac{n_{rc}}{n_c}$$

Equation 9

Remember from Equation 1 that n_c is the number of cells in the level and is equal to $(width \times height)$. n_{rc} is the number of reachable cells in the main space. Once we have the actual space ratio we can then calculate the space ratio error (R_e). The equation is squared to punish levels with a high space ratio error.

$$R_e = |R - r_a|^2$$

Equation 10

Once the space ratio error is calculated, the score from equation is multiplied to produce the final score which is assigned to the level by the fitness function. The following equation does this.

$$s = s_f \times (1 - R_e)$$

Equation 11

s is the score that is assigned to the level. However, if the start and/or end is occupied by a wall cell, then the s is assigned -10000. This is to ensure that if mutation does effect the start cell or the end cell, that the level is devalued significantly so that its information does not effect the future generations. The following pseudo code will give an overview of the steps described above.

```
main_space = largest space in level
start_space = space of the start
end_space = space of the end

$d_m$ = maximum possible distance in level
$d_s$ = shortest dist between main_space and start_space
$d_e$ = shortest dist between main_space and end_space

if (main_space does not contain start)
    $d_{fs}$ = longest distance in start space
    calculate($s_{ss}$)
else
    $s_{ss} = 30$

if (main_space does not contain end)
    $d_{fe}$ = longest distance in end space
    calculate($s_{es}$)
else
    $s_{es} = 30$

if (main_contains start and end)
    $s_w$ = 100
else
    $s_w = 0$
calculate ($s_{sd}$)
calculate ($s_{ed}$)

calculate($s_f$)
calculate($r_a$)
calculate($R_e$)
calculate($s$)
if (the start or end is blocked)
    $s$ = -10000
return s
```

4.2.3.2 Selection

Once the population has been scored by the fitness function, the individuals in the population are ready to mate in order to produce the next generation. However, to ensure that the principle of “survival of the fittest” is followed there needs to be a function that selects pairs of individuals for crossover, where the selection is in some way weighted or biased towards the most fit individuals. This is the selection function. In this research, a roulette wheel selection method was used.

First the function starts by getting the sum of the fitness scores of the entire population. Then, the score of each individual is normalized against the sum of the entire population. Each individual is given the percent of their original fitness score in the whole population. Assuming the population is ordered from highest to lowest, the function then iterates over the population. For each individual, the population is iterated over again. A probability is cast from the normalized fitness score. If it is true, then those two individuals are selected for crossover. The function then proceeds to next individual.

4.2.4 Crossover Function

Once individuals are selected for crossover, the contents of their chromosomes need to be exchanged. The Crossover function is responsible for this. Between two parent chromosomes, the crossover function uses information from each to construct two new children. Often the crossover function is uniquely specified for the search space at hand.

A standard crossover function was chosen and implemented for this research. After two chromosomes are selected, a random number between 0 and n of the level cells is selected. We will refer to this as the *index position*. The new chromosome will inherit all of the cells from before the *index position* from parent A, and after *index position*, cells are inherited from parent B. The function then repeats on the same parents, and creates a second child chromosome, except the parents are reversed, so before the *index position* cells are inherited from parent B, and after, cells are inherited from parent A. This is to ensure that the population remains constant. It also allows potential good features of a level to not be lost in the process, as the *index position* is random. The by-product of this is that the crossover function cuts the level horizontally. While this may not be ideal for a GA developed with an emphasis on the layout of contiguous space, it does surprisingly work, without artefacts that hinder the search for viable solution.

The following pseudo code describes the steps above:

```
new_population // we have a new population to fill

Parant_A // the first parent
Parant_B // the second parent

index_pos = random number between 0 and Parant_A.cells.Length // $n$

loop 2 times, with $i$ as index
    currentParentA // temporary first parent reference
    currentParentB // temporary second parent reference

    if ( $i$ == 0 )
        currentParentA = Parant_A
        currentParentB = Parant_B
    else
        currentParentA = Parant_B
        currentParentB = Parant_A

    child_level // create a child level

    loop for each ( $n$ cells in child_level), with $j$ as index
        if ( $j$ < index_pos )
            child_level.cells[ at $j$ ] = currentParentA.cells[ at $j$ ]
         else
            child_level.cells[ at $j$ ] = currentParentB.cells[ at $j$ ]

    new_population add level_child
```

4.2.5 Mutation Function

Lastly, after the crossover occurs between two chromosomes, the child is subjected to the mutation function. Mutation allows new features to be added the population, remove features that may be harmful, or adjust a feature that may be close to being desirable. Note that mutation is random in accordance with a probability. The Mutation Function should not target specific information in the chromosome, or adjust information towards a desirable end. Regardless, mutation in the natural world allows species to adapt to the environment, and in a GA, allows chromosomes a chance to get closer to the desired solution in the absence of desired features.

The Mutation Function utilised in this research was devised with a notion of contiguousness. This is to say that it operates on the contiguous arrangement of cells. Due to the nature of a level being a contiguous arrangement of tiles, this seems appropriate. Unlike a standard mutation function, where each element of a chromosome is given the chance to mutate, this contiguous mutation function selects a space at random and mutates the edge of that space. This allows spaces in the level to expand or contract in a mutation operation. This may allow rooms to connect, become two separate rooms, or carve a hallway over time. However, the GA did still allow a chance for a standard mutation to occur, where all the elements of the chromosome are given a probability to mutate.

Not all children will experience mutation. The mutation function is only applied if the mutation probability is cast and passes. In the event that it does, a mutation will occur and a random mutation probability is cast. As mentioned earlier, there is a chance a standard mutation occurs. This was intended so that mutations may occur to cells that are not part of a space, or not along an edge of a space. However, the probability of this is small, so it has a small chance of occurring. In the event that did does occur, each cell/chromosome element is given a probability to mutate, which switches the cell.

If the probability for random mutation fails, then the function proceeds to the contiguous mutation instead. The contiguous mutation function selects a cell at random, and finds the space containing that cell. If the cell is not associated with a space, the contiguous mutation function is repeated on another cell, until a cell which is part of a space is found. This concept is illustrated in Figure 14.

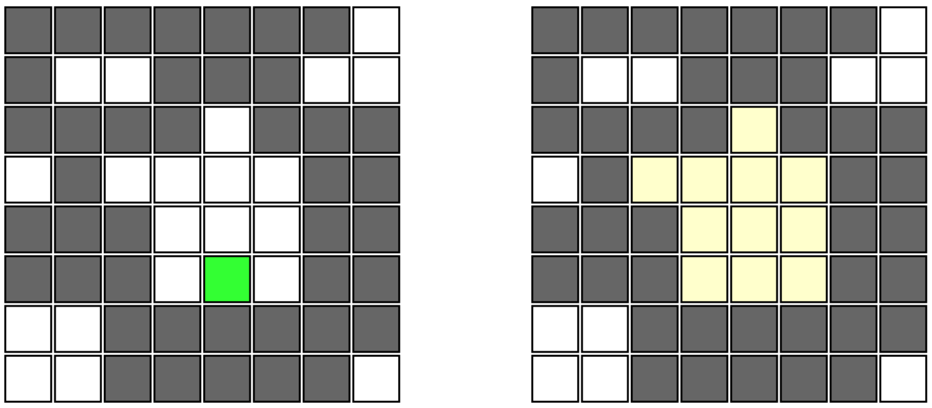


Figure 14: Space selected for mutation

Once the space is retrieved the function proceeds to the next step. There are two different operations the mutation function can perform on a space. Add to the space; make it bigger, or subtract from the space; make it smaller. Based on a probability, it will do one or the other. To add to a space, the outer edge of the space is retrieved. The outer edge are all the wall cells that surround the space, as shown in Figure 15.

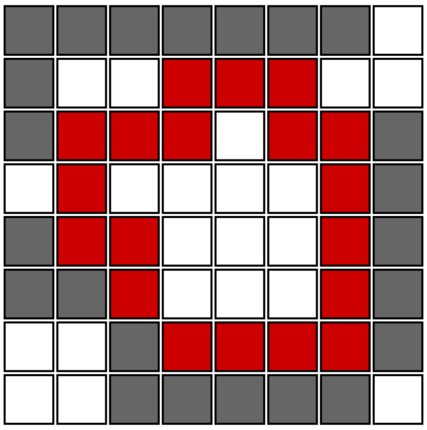


Figure 15: Outer edge of space

The outer edge of the space provides a set of candidate cells that can be mutated. Initially, a cell is chosen at random and its type change from WALL to VOID, as shown in Figure 16.

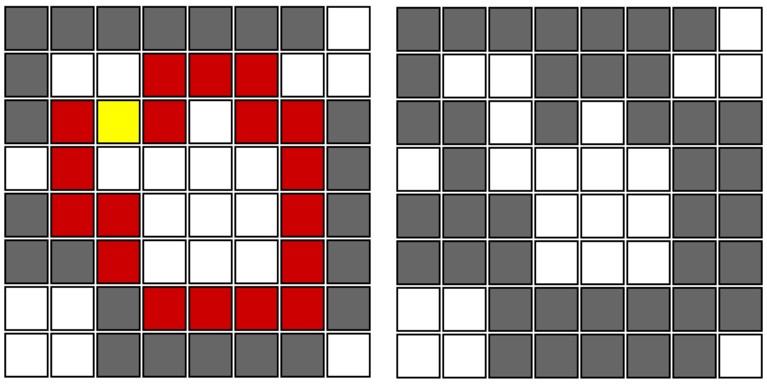


Figure 16: Mutation on cell (in yellow) in outer edge

Following the change of the initial cell, further cells may be mutated based on a probability. This is repeated each time on a random cell from the original set of outer edge cells. This continues until there are no more cells to mutate or the probability test fails.

To remove from a space, the steps above are repeated, but instead of getting the outer edge, the inner edge is retrieved. Figure 17 shows the inner edge of the same space.

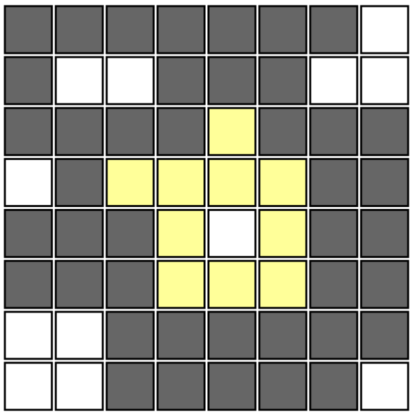


Figure 17: Inner edge of space

An inner edge are all the VOID cells that are along the edge of the space. Instead of being mutated to a VOID, it is mutated to WALL. This will cause the space to subtract.

Mutation is not limited to one space. After a space is mutated, a probability is cast that another space is mutated. If the probability passes, then steps above a repeated on another random space. However, a single space cannot be mutated multiple times in the same mutation function. The follow pseudo code gives an overview of the steps above.

```
mutation prob // the probability for mutation
random mutation prob // probability for random mutation
random mutation cell prob // probability for cell to mutation in
                          // random mutation

contig mutation volume prob // probability another space will be
                               // mutated
contig mutation add prob // probability a space will be added to
contig mutation edge prob // probability another cell in the edge will
                          // be mutated

function AddToSpace( space ) // space to be mutated
{
    outerEdge = get outer edge cells from space
    loop if ( contig mutation edge prob ) at least once

        rand_cell = random cell in the outerEdge
        if ( rand_cell has already mutated )
            continue

        rand_cell = VOID
}
function RemoveFromSpace( space )
{
    innerEdge = get inner edge cells from space
    loop if ( contig mutation edge prob ) at least once

        rand_cell = random cell in the innerEdge
        if ( rand_cell has already mutated )
            continue

        rand_cell = WALL
}
```

```
function randomMutation()
{
    loop for each ( cell in level )
        if (random mutation cell prob )
            switch cell
}
function contigMutation()
{
    loop if ( contig mutation volume prob ) at least once

        loop if ( space has not been mutated ) at least once

            rand_index = random between 0 and n
            if ( rand_index space has already been mutated)
                continue

            if ( rand_index points to a cell that is not a VOID)
                continue

            space = space of rand_index

            if ( contig mutation add prob )
                function AddToSpace( space to be mutated )

            else
                function RemoveFromSpace( space to be mutated )
}

if (mutation prob)
    if (random mutation prob)
        function randomMutation()
    else // since the random mutation failed,
         // we will perform a contiguous mutation
        function contigMutation()
```

4.2.6 GA Gameplay elements

The GA implementation described over the previous sections only evolved level topology. It does not evolve, or place gameplay elements in the level. This was intention to keep the GA relatively simple. In this research, gameplay element refers to any instance of game logic, such health pickups, weapons and enemy placements. As a consequence, game play elements have to be hand placed after the generation of the levels.

4.2.7 GA in PCG Taxonomy

As a PCG method has been created for this research, it needs to be situated in the taxonomy described in section 2.3. Evidently it is offline. The GA is executed prior to the runtime of the game. The content produced by the GA is necessary. The level needs to be playable. The user has various control over the GA fitness function, mutation function, and falling rectangle function. For the fitness function the user can only control desired aspect ratio. However, for the falling rectangle function the user can control following;

- Width
- Height
- Min rectangle width
- Min rectangle height
- Max rectangle width
- Max rectangle height
- Population
- Overlap percent

Note these were described in the equations in section 4.2.2.

The various probabilities of the mutation function can be controlled by the user. These dimensions of control all relate to the functionality of the GA. The user does not have control over any aspect of the level design. Once the GA is instantiated by the user, the user does not have any control.

The content produced by the GA can be identified as generic. There is no input from the player, or any information about player behaviour. As this the GA is a search algorithm, it falls under search based procedural content generation, which is a subset of Generate and Test.

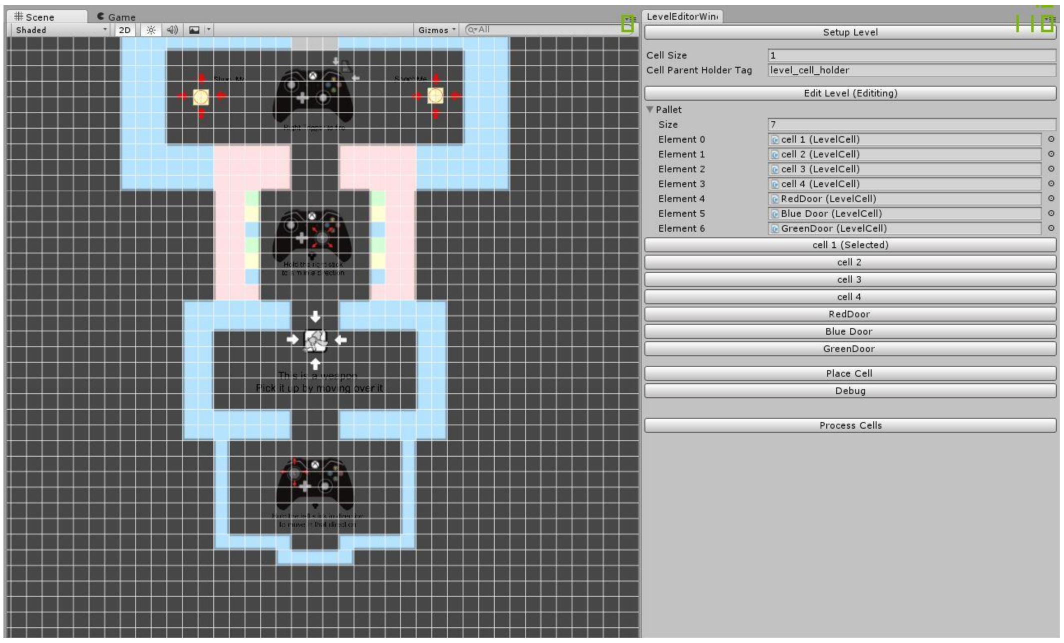
While the generation of the topology of levels is automatic, because the GA is unable to place game play elements, which is hand placed after the generation of the level, this PCG methods falls under mixed authorship.

4.3 Human Designed Levels

The previous section has outlined the implementation of the Genetic Algorithm used to generate game levels for the evaluation. In addition, to support the comparative aspect of the research, a number of human designed game levels were also developed.

These levels were developed for the human variant of the game. Game design students were recruited to design levels for the game. They were instructed to make a level to the best of their ability, using the game play elements that were already implemented in the game. To ensure that they understood the design of the game, the designers were given a copy of the game as a unity project file (*Unity*, 2016). The designers were only able to design levels and not change the underlying code of the game.

The level designers were also instructed in the use of a level editor that was implemented with the game. The level editor was needed so that a game level could be constructed intuitively, while maintain the level as an arrangement of tiles. Unity's default scene editor is not designed for level design. As such a customized level editor was developed. This level editor allows the placement of tiles along grid. Figure 18 displays a screen shot if this level editor in Unity.



Cell Size	1
Cell Parent Holder Tag	level_cell_holder
Pallet	
Size	7
Element 0	cell 1 (LevelCell)
Element 1	cell 2 (LevelCell)
Element 2	cell 3 (LevelCell)
Element 3	cell 4 (LevelCell)
Element 4	RedDoor (LevelCell)
Element 5	Blue Door (LevelCell)
Element 6	GreenDoor (LevelCell)

Figure 18 Screenshot the level editor with the tutorial level open

This was done to avoid a potential bias on part of the researcher who implemented the GA. There may be a bias if the individual whom implemented the GA also designed the levels.

4.4 Summary

This chapter has outlined the implementation details of the game developed to aid the research, along with consideration of the methods used for generating levels both procedurally and through the recruitment of human designers. Whilst not a formal output of this research, the development of the game mechanics and the game level editor was a significant effort that was necessary to ensure that the play testing evaluation was genuinely based around identifying differences that arose from the different spatial layout of the levels. The results of the playtesting are presented in the next chapter.

5 Results

This chapter will outline the data that was recorded from the play tests experiment. As previously discussed in section 3.3.2, participants that were recruited for the study were asked to fill out a pre-participant questionnaire, then proceed to play the game for approximately 30 minutes, after which they would fill out a post- play immersion questionnaire. The pre-questionnaire provides data about the participants' previous experience with games, which is important when examining the immersion questionnaire.

5.1 Participants

A total of 20 participants were selected for the study and randomly assigned a game variant to play. Immediately prior to play testing, demographic and game playing experience was collected using the pre-participation question given in Appendix A. Figure 19 shows the age range of all of the participants.

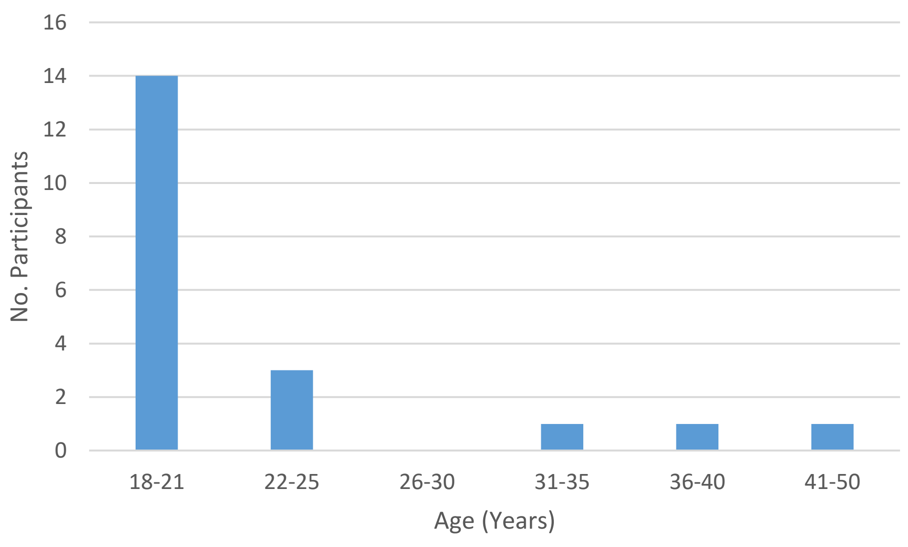


Figure 19 Age of participants

Figure 20 shows the gender distribution of all of the participants.

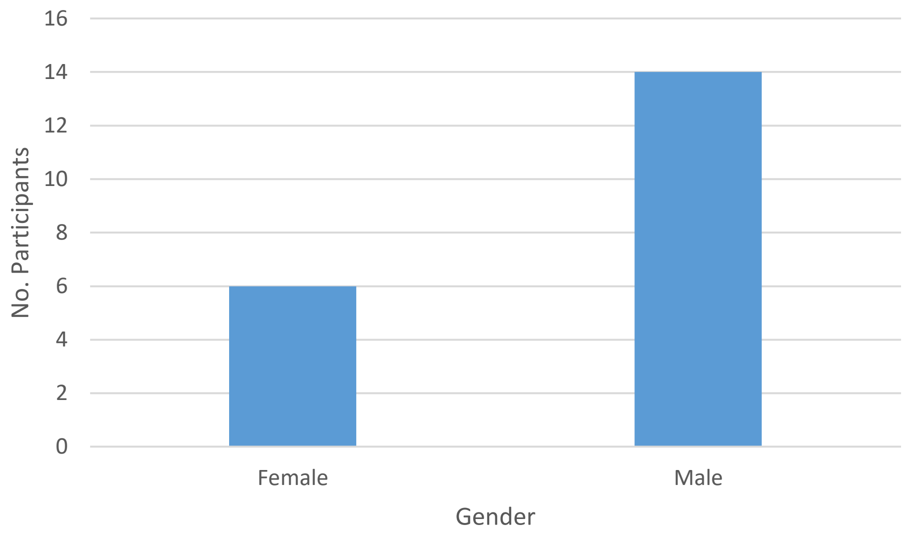


Figure 20 Participant gender distribution

Figure 21 shows the gaming experience of all of the participants.

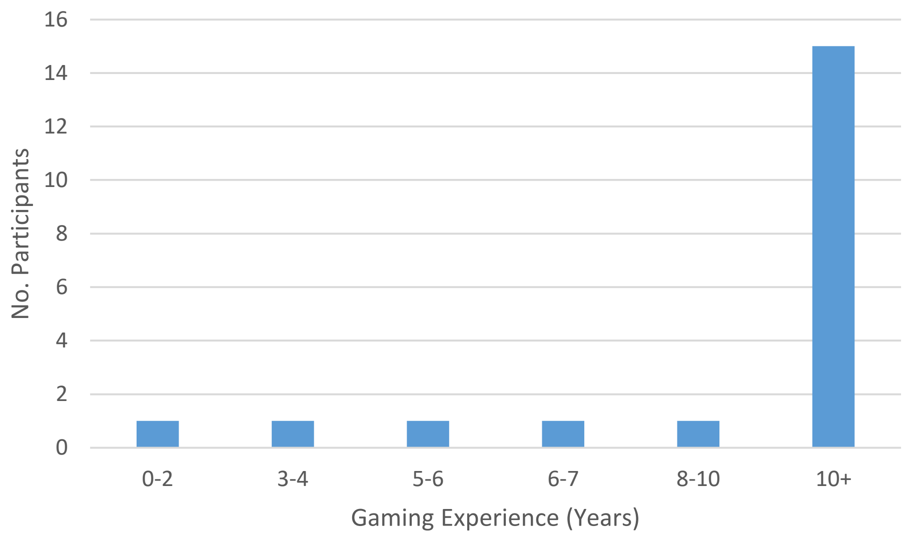


Figure 21 Participant gaming experience

Figure 22 shows how frequently the participants played video games.

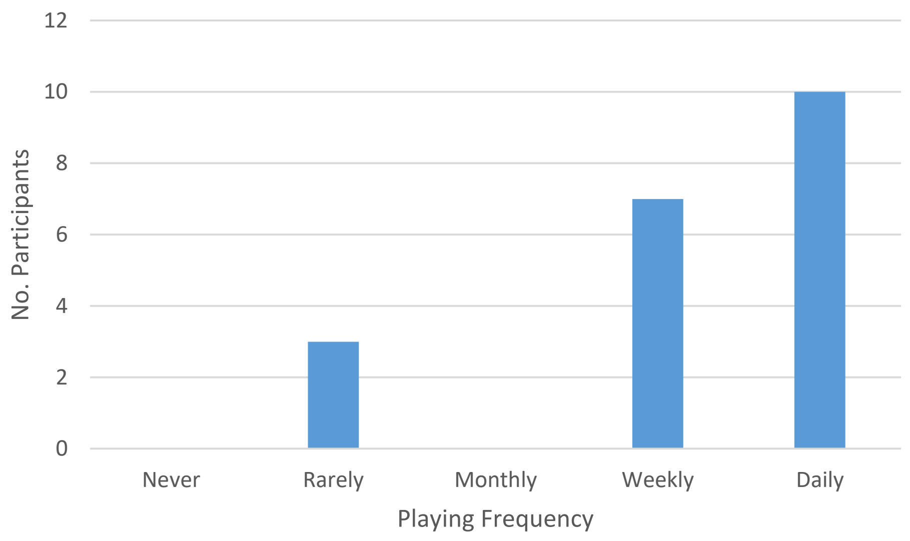


Figure 22 Participant playing frequency

Figure 23 shows the typical duration of a game play session for the participants.

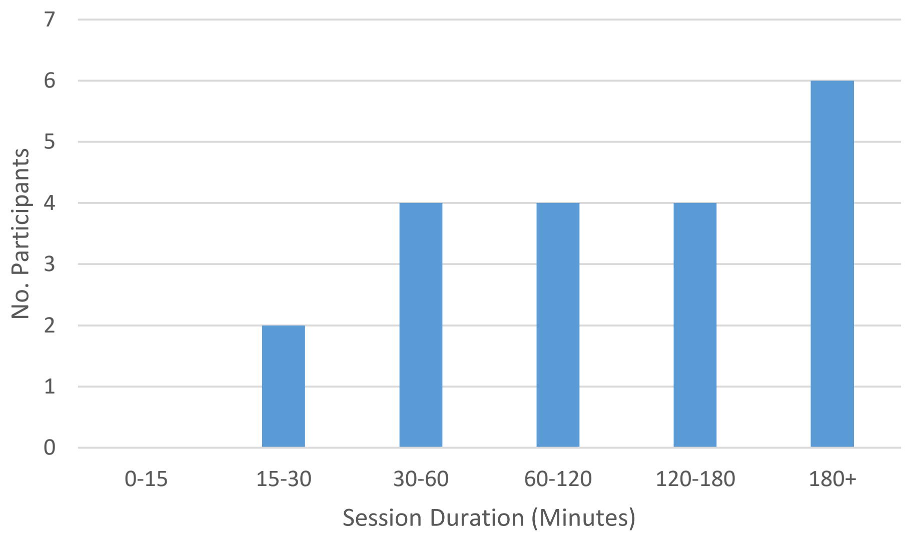


Figure 23 Participant session duration

During the playtesting sessions it was noted that a number of the less experienced gamers were experiencing difficulties with the controller and navigating through the game.

It was therefore decided to do a post-hoc pruning of the data to remove participant data from the analysis for those participants not playing video games at least an hour per week.

5.2 Playtesting groups

Following the post-hoc pruning, data was available for 7 players who played the human designed game levels and 9 players who had played the PCG levels. A further demographic analysis of the groups was conducted to show that the compositions of the groups were roughly comparable. Figure 24 shows the ages of each group.

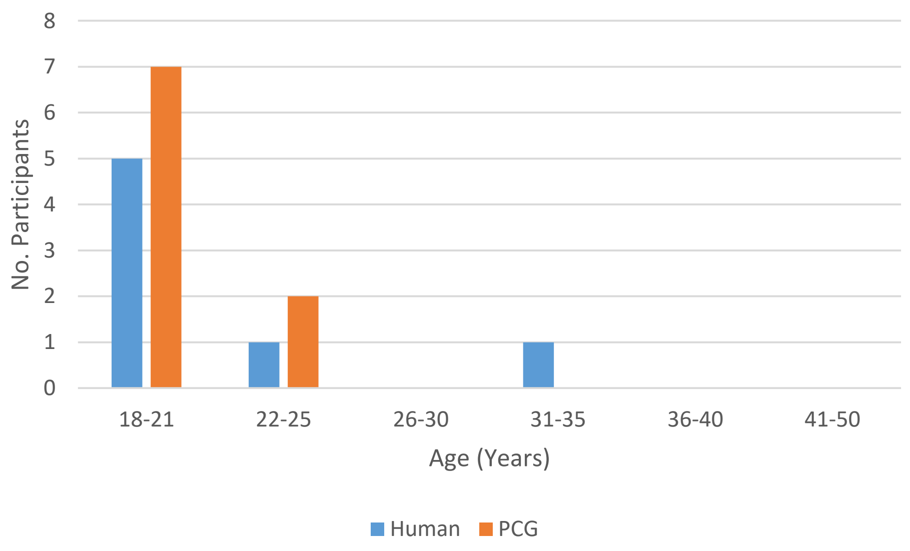


Figure 24 Comparison of ages in groups

Figure 25 shows the gender distribution of the groups.

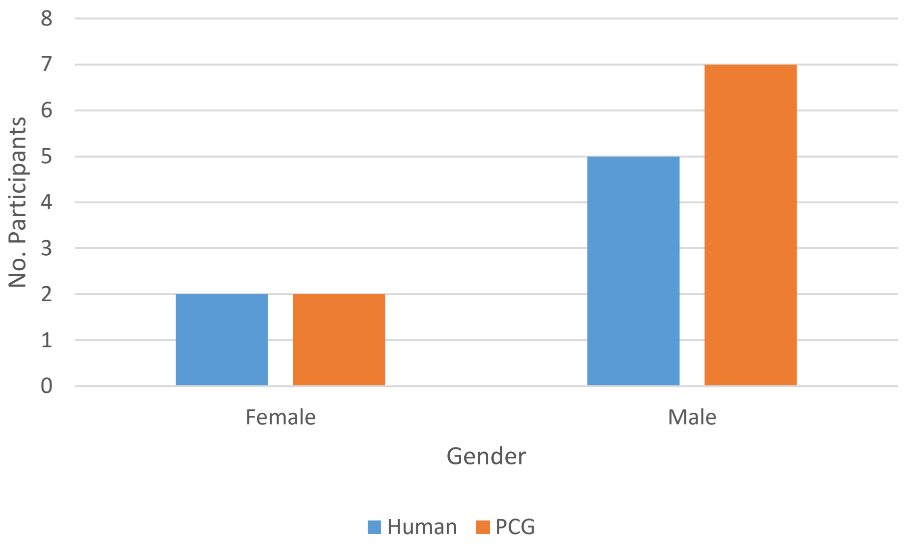


Figure 25 Comparison of gender distribution in groups

Figure 26 shows the gaming experience of the two groups.

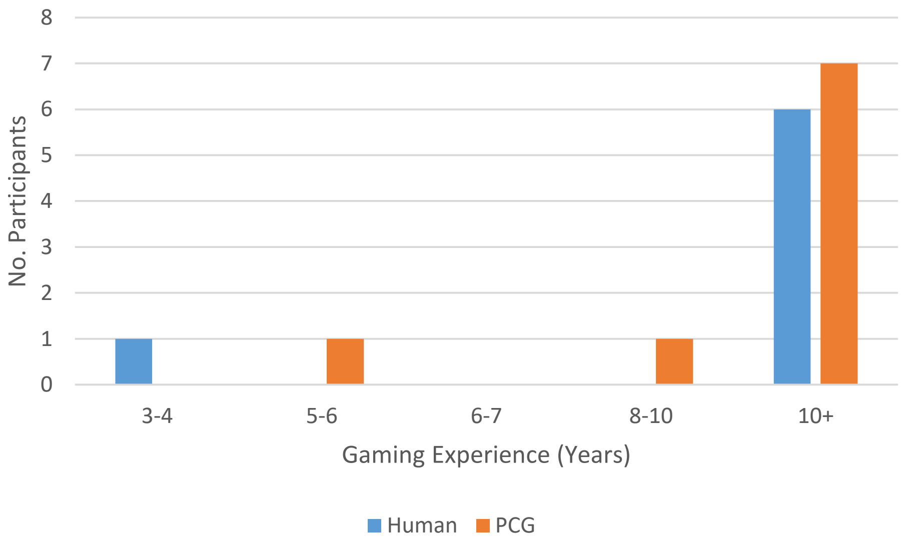


Figure 26 Comparison of gaming experience in groups

Figure 27 shows the frequency that participants in each group were playing video games.

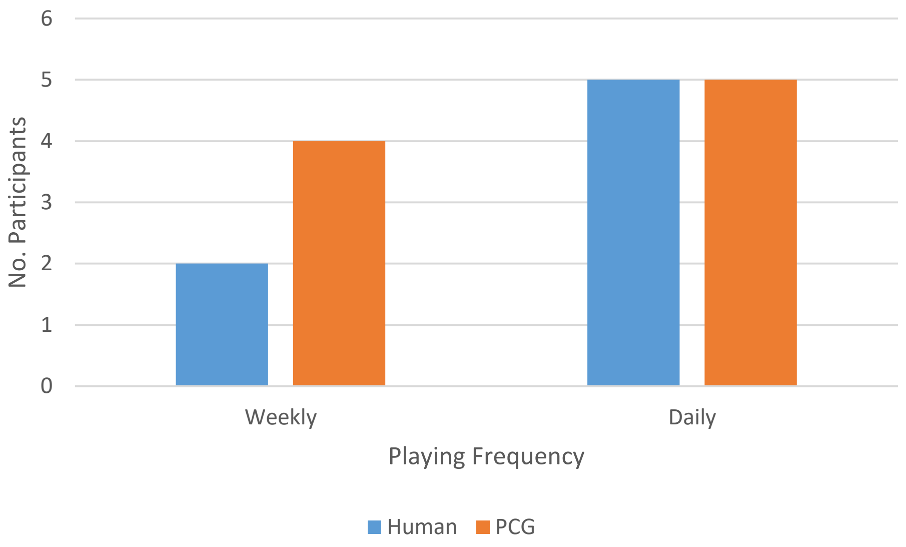


Figure 27 Comparison of playing frequency in groups

Figure 28 shows the typical length of a gaming session for each of the two groups.

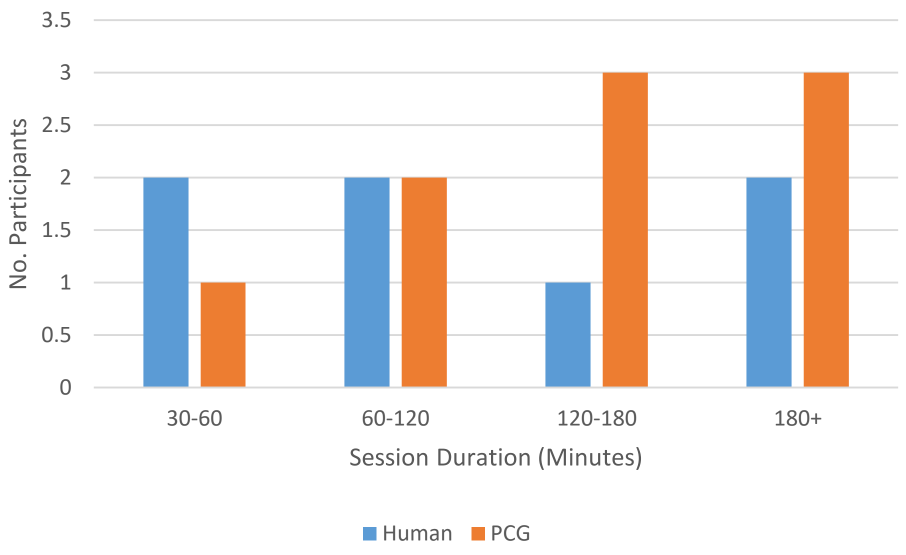


Figure 28 Comparison of session duration in groups

Whilst there are some minor variations between the two groups, they are sufficiently close to be considered a homogenous mix of active gamers, and as a result the likelihood of bias being present in the data as a result of differences between the two groups is low.

5.3 Total Immersion Analysis

Total immersion is a concept previously used by Denisova & Cairns (2015) where the responses of participants in the Immersive Experience Questionnaire (IEQ) are simply summed to produce a total score for the participant. To facilitate this, the data captured using the IEQ was adjusted so that all questions resulted in a score of 5 being the best score. The results for each participant for questions 5, 7, 8, 9, 17 & 19 in the IEQ were flipped to the corresponding value if the scale was reversed (e.g. a 2 becomes a 4, a 5 becomes a 1). Figure 29 shows an analysis of total immersion for the data.

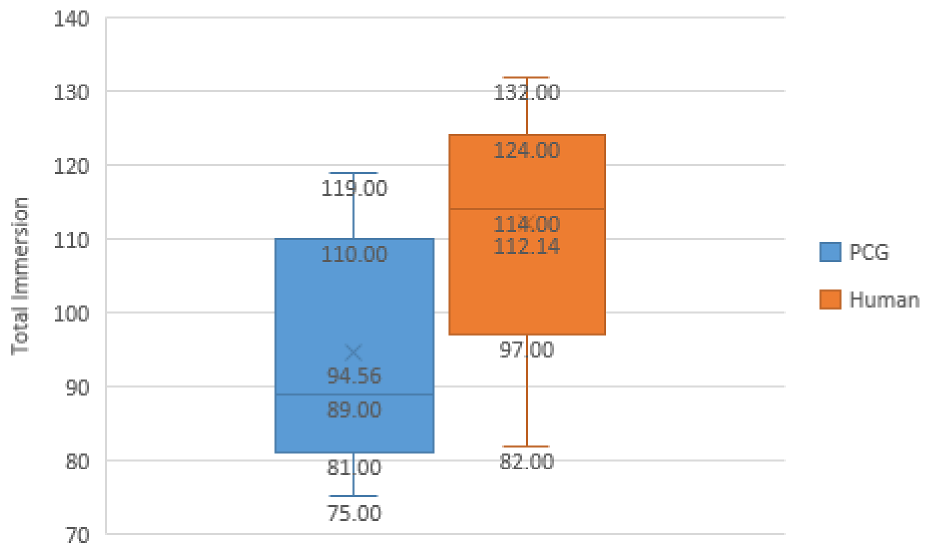


Figure 29 Comparison of total immersion

This figure shows that in general that the total immersion of players of the PCG variant is lower than that of players of the human designed variant. This is confirmed by the data relating to the plots shown in Table 1.

Table 1 Total immersion descriptive statistics

	Human	**PCG**
Mean	112.14	94.56
Median	114.00	89.00
Range	82.00 – 132.00	75.00 – 119.00
Interquartile Range	97.00 – 124.00	81.00 – 110.00

Both the mean and median scores of the PCG group are lower than that of the group who played human designed levels. The difference in mean scores between the two groups is 18, and the difference in median scores is 25. This can be placed into context by considering the number of questions in the questionnaire, and as such a typical difference between individuals of each group might be 1 point on the 5 point Likert scale for just over half the questions. Whilst this difference is small, there is still a need to formally address the null hypothesis through the use of statistical significance testing. This can be achieved using a one sided analysis of variance (ANOVA), an approach previously used by Denisova & Cairns (2015).

Table 2 shows the summary data generated for the ANOVA testing.

Table 2 ANOVA summary data

	Count	*Sum*	*Average*	*Variance*
PCG	9	851	94.55556	249.7778
Human	7	785	112.1429	296.1429

Table 3 shows the ANOVA outcomes generated by using the Excel single factor ANOVA analysis.

Table 3 ANOVA results

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1217.9206	1	1217.9206	4.5167	0.0518	4.6001
Within Groups	3775.0794	14	269.6485			
Total	4993.0000	15				

Based on an α-value of 0.05, this shows that there is no statistically significant differences between group means as determined by one-way ANOVA (F(1,14) = 4.5167, p = 0.0518). This is somewhat contrary to the data presented in Table 1 and Figure 29 where a clear difference is visible, suggesting firstly that the actual distribution of values is not sufficiently distinct for the differences to be clear. It also suggests that the sample size may not be sufficiently large for there to be high confidence in the outcomes.

Given that the p-value is only marginally greater than the α-value, and that the F-value is smaller than F_{crit}, the outcomes of the ANOVA are definitely borderline, as might be expected from the visual interpretation of Figure 29. It is therefore important to not over-stress the outcomes of the statistical analysis, particularly given the limited sample size.

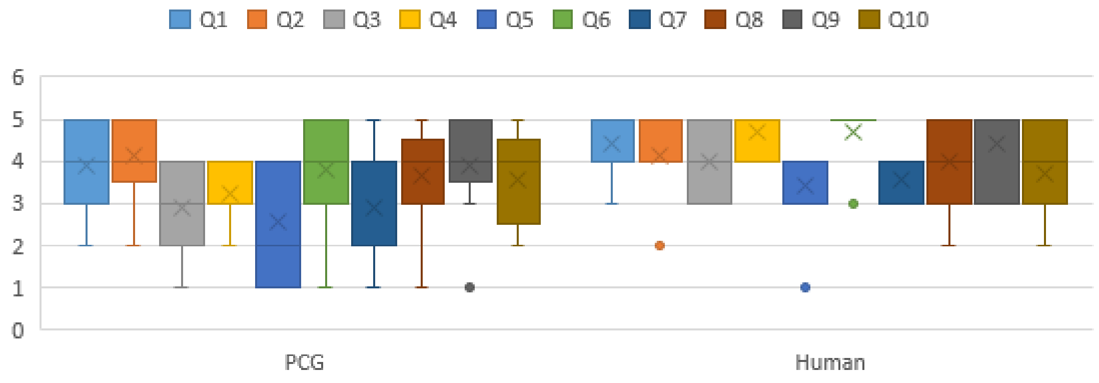
Whilst the one-way ANOVA is considered a robust test against the normality assumption, which implies that it tolerates violations to its normality assumption rather well, this robust only applies to skewed or kurtotic distributions. In particular, with small sample sizes, platykurtosis can have a profound effect.

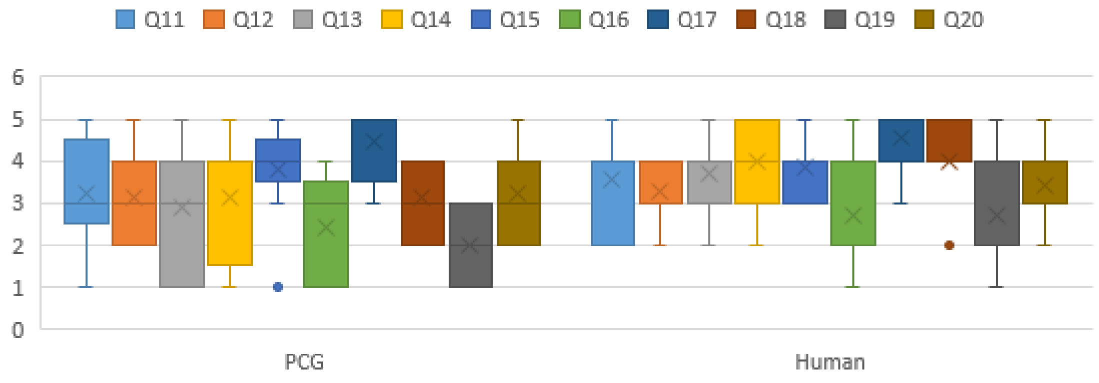
To confirm the null hypothesis, a non-parametric Kruskal-Wallis test is conducted. The Kruskal-Wallis test statistic is approximately a chi-square distribution. Based on the significance level of $\alpha = 0.05$, and the number of degrees of freedom is $df = 2 - 1 = 1$, the rejection region for this Chi-Square test is $R = \{X^2: X^2 > 3.841$. The H statistic ($\approx X^2$) is calculated to be $H = 4.045 > R$ (= 3.841). It is then concluded that the null hypothesis is rejected. Similarly, the p-value is $p = 0.0443$, and since $p = 0.0443 < 0.05$, it is concluded that the null hypothesis is rejected.

The outcomes of the Kruskal-Wallis test contradict that of the ANOVA, however the P-value is still close to the α-value which suggests again a borderline outcome in terms of statistical significance. It is possible that some contribution to the outcomes being borderline are related to the aggregation of the questionnaire results into a single measure of total immersion, therefore the following section analyses the questionnaire responses at a higher level of granularity to identify any hidden detail.

5.4 Detailed Questionnaire Analysis

The Immersive Experience Questionnaire (see Appendix B) consists of 30 questions and the box plots in Figure 30 show an analysis of individual question responses.





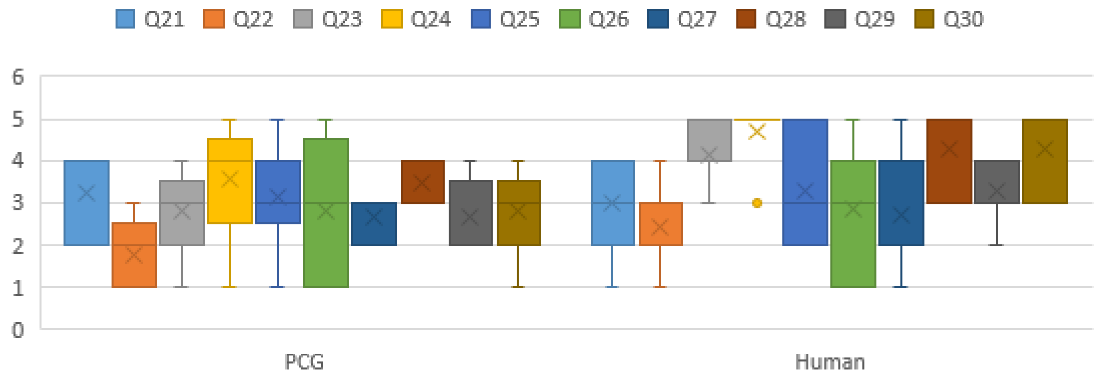


Figure 30 Comparison of individual question responses

The descriptive statistics of the responses to each question are presented in Table 4.

Table 4 Summary of questionnaire results by question

	PCG		
	Mean	Median	SD
Q 1	3.89	4.00	1.10
Q 2	4.11	4.00	0.99
Q 3	2.89	3.00	0.99
Q 4	3.22	3.00	0.63
Q 5	2.56	2.00	1.34
Q 6	3.78	4.00	1.23
Q 7	2.89	2.00	1.29
Q 8	3.67	4.00	1.15
Q 9	3.89	4.00	1.20
Q 10	3.56	4.00	1.07
Q 11	3.22	3.00	1.23
Q 12	3.11	3.00	1.10
Q 13	2.89	3.00	1.45
Q 14	3.11	4.00	1.37
Q 15	3.78	4.00	1.13
Q 16	2.44	3.00	1.17
Q 17	4.44	5.00	0.83
Q 18	3.11	3.00	0.87
Q 19	2.00	2.00	0.82
Q 20	3.22	3.00	1.03
Q 21	3.22	4.00	0.92
Q 22	1.78	2.00	0.79
Q 23	2.78	3.00	0.92
Q 24	3.56	4.00	1.26
Q 25	3.11	3.00	1.10
Q 26	2.78	3.00	1.55
Q 27	2.67	3.00	0.47
Q 28	3.44	3.00	0.50
Q 29	2.67	2.00	0.82
Q 30	2.78	3.00	0.92

	Human		
	Mean	Median	SD
Q 1	4.43	5.00	0.73
Q 2	4.14	4.00	0.99
Q 3	4.00	4.00	0.76
Q 4	4.71	5.00	0.45
Q 5	3.43	4.00	1.05
Q 6	4.71	5.00	0.70
Q 7	3.57	4.00	0.49
Q 8	4.00	4.00	1.07
Q 9	4.43	5.00	0.90
Q 10	3.71	4.00	1.03
Q 11	3.57	4.00	1.05
Q 12	3.43	3.00	0.49
Q 13	3.71	4.00	0.88
Q 14	4.00	4.00	1.07
Q 15	3.86	4.00	0.64
Q 16	2.71	2.00	1.28
Q 17	4.57	5.00	0.73
Q 18	4.00	4.00	0.93
Q 19	2.71	2.00	1.28
Q 20	3.43	3.00	0.90
Q 21	3.00	3.00	1.07
Q 22	2.43	2.00	0.90
Q 23	4.14	4.00	0.64
Q 24	4.71	5.00	0.70
Q 25	3.29	3.00	1.28
Q 26	2.86	3.00	1.36
Q 27	2.71	2.00	1.28
Q 28	4.29	5.00	0.88
Q 29	3.29	3.00	0.70
Q 30	4.29	5.00	0.88

In terms of the mean responses to each question, the data related to participants playing the human designed game variant is consistently at least equal to that for participants playing the procedurally generated levels. The one exception to this is for

Q21 (“How well do you think you performed in the game?”). In terms of the median responses, a number of questions have produced data that is comparable. These include Q2, Q8, Q10, Q12, Q14, Q15, Q17, Q20, Q22, Q25 and Q26. Of these questions, of particular interest are Q2 (“To what extent did you feel you were focused on the game?”), Q22 (“To what extent did you feel emotionally attached to the game?”) and Q25 (“Were you in suspense about whether or not you would win or lose the game?”) where the standard deviation of the data related to the procedurally generated level is equal to or smaller than that of the data related to the human designed variant. This suggests that there is more coherence and agreement in the responses to these questions.

To be able to further confirm where there are similarities between the responses of the two groups, a one-way ANOVA has been conducted on the responses for each individual question. This analysis is presented in full in Appendix B, and Table 5 presents the P-values for each test.

Table 5 ANOVA P-values for responses to individual questions

Question	ANOVA P-value
Q 1	0.3122
Q 2	0.9535
Q 3	***0.0375***
Q 4	***0.0002***
Q 5	0.2064
Q 6	0.1137
Q 7	0.2340
Q 8	0.5887
Q 9	0.3686
Q 10	0.7832
Q 11	0.5829
Q 12	0.7370
Q 13	0.2344
Q 14	0.2071
Q 15	0.8789
Q 16	0.6866
Q 17	0.7693
Q 18	0.0873
Q 19	0.2247
Q 20	0.7009
Q 21	0.6821
Q 22	0.1720
Q 23	***0.0072***
Q 24	0.0598
Q 25	0.7878
Q 26	0.9214
Q 27	0.9245
Q 28	***0.0404***
Q 29	0.1566
Q 30	***0.0077***

Based on an α-value of 0.05, analysis of these p-values suggests that there is a statistically significant difference between the responses from the two groups for Q3 ("How much effort did you put into playing the game?"), Q4 ("To what extent did you lose track of time?"), Q23 ("To what extent were you interested in seeing how the game's events would progress?"), Q28 ("How much would you say you enjoyed playing the game?") and Q30 ("Would you like to play the game again?").

Cross-referencing these questions to the descriptive statistics in Table 4 would suggest that the human designed game levels have achieved a higher level of immersion in the dimensions related to these questions when compared to the procedurally generated game. Whilst the median and mean for other questions are higher for the human designed game players, the spread of responses to these questions is such that the difference is not statistically significant.

5.5 Discussion of Results

The outcomes observed from the results above need to discussed and put into context in order to identify the value and significance of the outcomes.

5.5.1 Genetic Algorithm

First, it should be noted that these results are related to a specific implementation of a Genetic Algorithm that produces game levels, in relation to the implemented game itself, and should not generalised for the PCG methods at large. The specific fitness, crossover, mutation and evaluation functions that have been implemented are what is being examined here.

In this research, the fitness function, which was responsible for selecting the best levels, only looks at; whether the level is playable, and the ratio between space and the total area of the level. In conjunction with the falling rectangle function, the GA is able to generate levels that can be completed. In its current implementation though, the GA implemented here cannot achieve an immersion score equal or higher than levels designed by a human.

The notions that formed these two functions assume that the ability to complete a level and the ratio between the space the level's dimensions are qualities of good level design. This is just an assumption in this research. In a broader context that takes into consideration player experience and game design, these qualities may be outweighed by other qualities of the level. These qualities are subject to the specific game design in question.

In light of this outcome, it can be anticipated that the success of a PCG method would implement from a mapping between game design and a positive player experience. The results from this research highlights the importance of this. In this case there is no mapping between player experience and good level design. To describe this in the schema established in section 4.1.4, there is no effective mapping between the current game design of this research and engaging spatial mechanics, implemented in the fitness function of the GA.

5.5.2 Observation of differences between PCG content and Human content

In discussing these results, it would be a disservice to not visually examine the actual levels from the variants. Figure 31 shows a level from each variant, with (a) being the procedurally generated level and (b) being the human designed level.

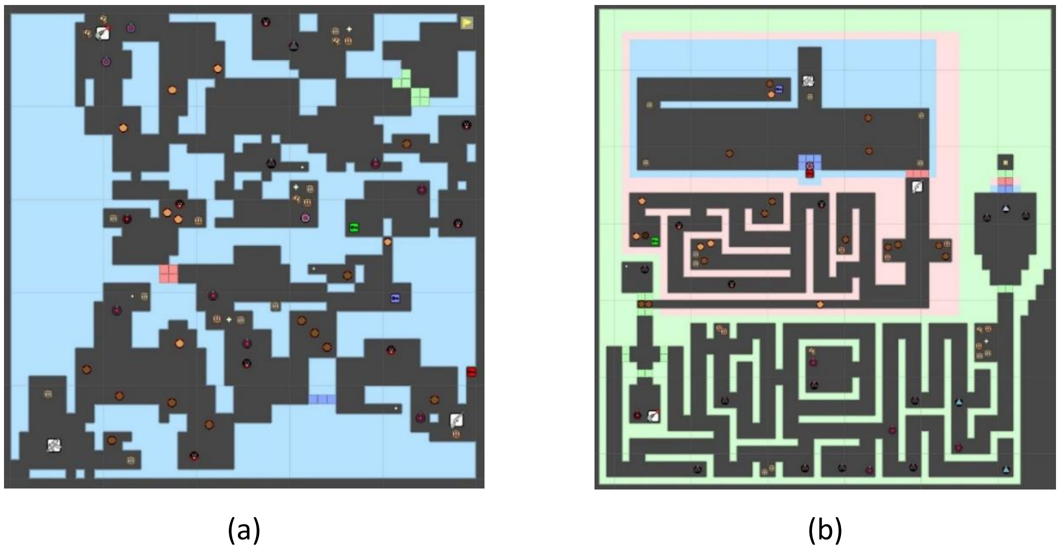


Figure 31 Sample levels

As it can be seen, the design between the GA variant and the human variant is quite different. There is a striking difference in the way things are laid out between the two. This can be seen as the GA starts at the bottom left, and moves towards the top right, a limitation of the current implementation of the GA. While the human level starts at the top and through a maze, delivers the players to the end on the right.

The first obvious feature is that the GA level is generally chaotic while the human level is straightforward. Specifically, hallways are not always straight and rooms are not usually well defined. There is also noise, instances where tiles or gaps may seem misplaced. Regardless of the improvement between the random level generator and falling rectangle generator, there still are chaotic features of the level.

The human designed level has clear intent in its design. There are clear distinctions of different parts of the level. Hallways are straight and rooms are well defined. The topology is not too noisy in relation to the GA variant.

While the GA does provide opportunities for different rooms and hallways, and for gameplay elements to be placed in interesting configurations, its chaotic features may make all of these features feel meaningless to a player. From this observation, the main features that are evident, is noise; the perceived misplacement of tiles, and the lack of distinction; ambiguity between hallways and rooms. However, it remains hard to ascertain directly why the human variant had the higher immersion rating.

5.5.3 Individual Questions

As described in the results section, four questions were more consistent in identifying the human variant as more immersive. Those questions were:

- Q3 ("How much effort did you put into playing the game?")
- Q4 ("To what extent did you lose track of time?")
- Q23 ("To what extent were you interested in seeing how the game's events would progress?")
- Q28 ("How much would you say you enjoyed playing the game?")

The questions in the original paper by Jennett et al (2008) were organized into sections. Basic Attention ranges from question 1 – 4. Both Q3 and Q4 fall into this section. This implies that GA variant failed to hold attention in comparison to the human variant.

The noise identified earlier may be responsible for this. As the excess of noise in the level may result in the level feeling undirected and natural. The human variants may direct the users' attention via the placement of the topology, while, the "natural" effect of noise may influence users to not regard those aspects of the topology.

Q23 is related to emotional involvement. The entire game was designed to avoid this by remaining abstract. Regardless, it appears that the human variant was able to achieve this to a higher degree than the GA variant. As this is the case, a quality in the human designed levels that is currently not in the generated content may be responsible for this. Perhaps this is due to structured nature of the human designed variant. Players could be lead in direction throughout the levels due to the placement of level topology, thus, influencing the insuring series of the events. At the very least, this could be argued to be some sort of narrative. The lack of distinction may also be responsible for this, as the human variants tendency to use room and hallways in the layout of the level, may contextualise the game to in some respect.

The last question can be seen as a one of the summary questions in the questionnaire as it relates to general enjoyment. It would not be a leap to assume that the GA variant was less enjoyable than the human variant, both from the results and the discuss of the questions above.

While these questions identify the most apparent instances where the human variant may achieve a higher immersion, the resulting discussion is ultimately speculation.

6 Conclusion

This research has evaluated the effect of a Procedural Content Generation method on the perceived immersion of a game. A relationship between the use of a PCG method implemented in this research, and the immersion rating of the game has been identified. The results analysed from the play test experiment shows that the human levels of the game had a higher immersion rating than the levels produced by the PCG method. However, acknowledging the limitations of the sample size, the difference between the total immersion of the two groups was not found to be statistically significant using a one-way ANOVA test.

A game was designed and implemented for this study. Two variants of the game were produced. One of those variants featured levels designed by a human designer. The other variant featured levels that were evolved using a specific genetic algorithm implementation. In this regard, this thesis represents a significant undertaking and one of the first attempts to consider how procedurally generated content is received by play testers in comparison to equivalent non-procedurally generated content. As such it makes a significant contribution to the existing literature in this field.

To arrive at the conclusion stated above, the scores of the individual questions in the player immersion questionnaire had to be examined. The average sum immersion scores between the human variant and the PCG variant is not of significant difference to identify a relationship. Examining the individual questions revealed a significant difference in favour of the human variant of the game.

This indicates that the PCG generated levels is not able to produce levels that can achieve the same level of immersion as the levels featured in the human game variant. This identifies an inverse relationship between the PCG method implemented in this and the perceived immersion of the game. However, this study was conducted with a small sample size. A larger sample size may confirm the outcome identified here.

6.1 Potential Limitations

There are a range of the limitations in this research that need to be noted.

6.1.1 Play test experiment

As it has been mentioned, there are a variety of factors from the play test experiment that can be considered limitations of the research. The amount of the participants is the first obvious limitation. The sample size of the study is small. In examining the results from this research this needs to be considered. Recruiting and organizing individuals for a 30 minute play test in the scope of this research was the primary cause of this.

The amount of time for the play test is may be another limitation. Participants were only given roughly 30 minutes to play. Although, depending on the state of the game, participants were allowed to play for a bit longer if they were close to finishing. This was done to ensure their experience of the game was not interrupted. However, some participants were able to finish the game in the 30-minute period. 30 minutes may not be long enough to ensure the immersion rating is accurate. In this case, to evaluate the immersion rating with higher clarity, a longer play test time would be needed.
The scope of this research limited the play time to 30-minutes.

6.1.2 Limitation in the GA

The GA implementation only evolves level topology. Game play elements (picks up, doors, keys, enemies) are hand placed after the generation of a successful solution. This is intentional. Research into utilizing player experience in the PCG is beyond the scope of this study. As such, for the clarity of the research, it was decided that only the level topology would be evolved. In this context, the immersion rating differential is only indicative of the level topology of the generate content. However, it needs to be noted that human placed game play elements would have an effect on the immersion rating of the variants. As both game variants have human placed game play elements however, the clarity of this research should not be compromised.

The human designed levels were able to start from any position and end at any position. GA levels could only start at the bottom left and end at top right. This limitation on the GA, or the freedom for the human designer, could be argued to influence the clarity of the research.

The fitness function in the GA evaluated multiple measurements and then combined them into a single score. The various weighing on each score was assigned an arbitrary value. While this is not considered ideal, the scores where kept proportional to their criteria as much as possible. While the fitness function was able to produce playable levels, the arbitrary weights may limit the clarity of evaluating this PCG method. More elaborate GAs, such as a multi-objective GA, may solve this limitation.

6.1.3 Clarity of game design

To evaluate the hypothesis with clarity, the game design would need to be approachable to a variety of players. If the design was too difficult, even if it is the same over both variants, other unintentional variables may be introduced into the experiment, or pre-existing variables exacerbated. While there may already be a multitude of factors that cannot be controlled in a study of this nature, there may be an ideal game design that can minimize these factors to an extent.

As such, the design of the game that is to be tested, needs to be carefully considered in a study of this nature. The game that was developed for this research was intended to be straightforward, using a simple perspective and a simple objective. However, it may be the case that the design of Vektor in this study was not situated closer to that ideal. Section 4.1 described the game design of Vektor, but nuances that are not easily described may also affect the design with regards to this approachability.

6.1.4 Game Design understanding across variants

To avoid a bias in the design of the levels, different individuals were responsible for the design of the human variant levels. The designer of the GA did not design levels for the human variant. This was done to avoid a bias on part of the researcher. A limitation of the current implementation of the GA, is that it only evolves level topology. Gameplay elements were added in after the generation of the level by the designer of the GA, as he was responable . The designers of the human variant were not responsible for this. This means that the understanding of game design behind both variants are different. This may influence the clarity of the research.

6.2 Future Work

The limitations associated with this work should not detract from the contribution this thesis makes, indeed in many cases the limitations arise as a direct result of pragmatic choices made to ensure the work could be completed in a feasible timescale. The limitations are therefore an indication of directions for future work.

The immersion ratings for the GA variant were slightly lower than the immersion ratings from the human variant. The exact reasons for this outcome are hard to ascertain.

Discussion as to why this is the case, may identify notions about aspects of the produced levels (from both variants) that can be explored in later research.

There are also refinements and improvements that can be made to the Genetic Algorithm developed in this research. Player experience research and discussion about game design can inform future research.

6.2.1 Design of the GA

The assumptions about level design that formed the fitness function and falling rectangle function was not able to create levels with an equal or higher immersion rating then that of a human designer.

Expanding these assumptions to include more accurate notions of immersive level design would be a natural continuation of this research. The obvious first approach would be to incorporate ideas from player experience research into these assumptions. The questionnaire that was used originates from that research area, but was only used to measure the perceived immersion.

Refinement could be made to Genetic Algorithm implemented in this research. The current implementation of the fitness function may not treat all levels that are close to having a playable space the same. As the function scores how close the main space is to the start and end using Pythagoras, spaces along the middle line might be given a higher score then the spaces that are along edges of the levels. This may exclude levels that are just as playable because their spaces are more along the edge.

The crossover in the GA implementation is standard, and was not designed from notions of contiguousness, like the rest of the GA. While different crossover implementations were designed, all of them were too destructive toward the population of game levels to be evaluated. Game levels were that evolved were unable to become playable as features of the levels that could contribute to the playability would be overwritten during the crossover operation. So a standard crossover function was implemented as it was able to evolve playable game levels. Implementing a crossover function that incorporates notions of contiguousness is still of interest, as the game levels are fundamentally a contiguous arrangement of topology.

This research only evaluates the PCG method implemented here, and hence the results can only be applied to the particular game and PCG method implemented in this research. Performing the same evaluation on other PCG and EC methods would be considered an interesting continuation of this research.

Currently the GA as a whole only evolves the topology of the game level. Gameplay elements, like enemies and pickups, are hand placed in the level after it is evolved. Expanding the GA to evolve levels with game play elements would be a continuation of this research. It is anticipated that this would expand the GA implementation considerably.

6.2.2 Game Design Schema and PCG

However, player experience is not the only area of research that would inform this goal. There is a larger academic discussion about what games are, and how best to describe them. The MDA framework (Hunicke et al., 2004), is a part of that discussion. While the MDA framework was helpful for contextualising the role of the GA in the design of Vektor, it remains difficult to accurately describe the nuances in the GA variant and human variant without resorting to speculation. This is no fault of the MDA framework, it provides a novel schema for the design and analysis of games.

The goal of PCG would benefit from a formal schema that can accurately describe games. The ability to account for the nuances in game design would inform the conception and design of PCG systems and allow them to target specific aspects in the interaction between the player and the game.

In section 5.5.2, the two features visually observed between the GA level and the human level, was how the human levels could communicate a distinction between different areas simply from its topology and the amount of the noise in the level. Accounting for these features in future GA or PCG implementations would be of value. Controlling the amount of noise in the generation process so that there is a balance is one notion that can be examined.

The other notion, is how to make areas of the level feel distinct and defined without an explicit definition. While 'hallways' and 'rooms' was used to discuss this in section 5.5.2, this approach may be limiting, as it defines aspects of a level against a pre-existing structure, while a level is simply a contiguous arrangement of topology. Potential configurations that do not conform to pre-existing structures would not be conceived. While the falling rectangle function in this research does have a defined shape, mutation and crossover do not, and changes to levels through generations have the potential to significantly change the shape over time.

6.2.3 AI assisted game design with GA method

While the current GA implementation resulted in lower immersion, it is able to produce game levels that are playable. As the difference in immersion between the two variants are relatively small, there is potential to use this method in conjunction with human authorship to a larger extent. This would situate the method toward AI assisted game design.

6.2.4 Further testing

The sample size for this study was due to the small scope of the research. Pursuing other avenues for testing the two variants is one way to counter this limitation. As the game developed for this research was made with the Unity game engine in the year 2016 (*Unity*, 2016), the game can easily be compiled to a web format and put online. This can generate much more data then conducting an onsite play test. This may be a good option to confirm the results here.

6.3 Summary

This research has evaluated the effect of a Procedural Content Generation method on the perceived immersion of a game.

A game was developed for a play test experiment, in which participants played two variants of a game. A human designed variant and a PCG designed variant. The PCG variant uses a specific genetic algorithm implementation that produces playable games levels.

Using a player immersion questionnaire, the immersion scores of each game variant was recorded. It was found that the human variant had on average a higher immersion rating. However, there was not a significant difference between the two groups. On further examination of individual questions from the questionnaire, a significant difference was found in favour of the human variant. The PCG method was not able to produce levels that had an equal or higher immersion score then the human variant. Because of this, an inverse relationship between the specific PCG method used in this research and the perceived immersion of the game has been identified. However, the sample size was small and further research may clarify this outcome to a larger extent.

6.4 References

Ashlock, D., Lee, C., & McGuinness, C. (2011). Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 260–273.

Blow, J. (2004). Game development: Harder than you think. *Queue*, *1*(10), 28.

Blizzard North. (1996). Diablo [PC] Blizzard Entertainment & Ubi Soft Entertainment.

Boggus, M., & Crawfis, R. (2009). Procedural creation of 3d solution cave models. In *Proceedings of the 20th IASTED International Conference on Modelling and Simulation* (pp. 180–186).

Bourassa, M., & Massey, L. (2012). *Artificial Intelligence in Games* (Technical Memorandum No. DRDC Ottawa TM 2012-084). Retrieved from http://cradpdf.drdc-rddc.gc.ca/PDFS/unc120/p536670_A1b.pdf

Bourke, P., & Shier, J. (2013). Space Filling: A new algorithm for procedural creation of game assets. In *Proceedings of the 5th Annual International Conference on Computer Games Multimedia & Allied Technology*.

Boyle, E. A., Connolly, T. M., Hainey, T., & Boyle, J. M. (2012). Engagement in digital entertainment games: A systematic review. *Computers in Human Behavior*, *28*(3), 771–780.

Brown, E., & Cairns, P. (2004a). A grounded investigation of game immersion. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 1297–1300). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=986048

Brown, E., & Cairns, P. (2004b). A Grounded Investigation of Game Immersion. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (pp. 1297–1300). New York, NY, USA: ACM. https://doi.org/10.1145/985921.986048

Cardamone, L., Loiacono, D., & Lanzi, P. L. (2011). Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 395–402). ACM.

Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM*, *50*(4), 31–34.

Compton, K., & Mateas, M. (2006). Procedural Level Design for Platform Games. In *AIIDE* (pp. 109–111).

Connor, A. M., & Gavin, J. (2015). Reinventing the arcade: computer game mediated play spaces for physical interaction. *EAI Endorsed Transactions on Creative Technologies*, *2*(5), e4.

Connor, A. M., & Shah, A. (2014). Resource allocation using metaheuristic search. In *Fourth International Conference on Computer Science & Information Technology*.

Connor, A. M., & Shea, K. (2000). A comparison of semi-deterministic and stochastic search techniques. In *Evolutionary Design and Manufacture* (pp. 287–298). Springer.

Csikszentmihalyi, M., & Csikszentmihalyi, I. S. (1992). The Flow Experence and its significance for humanpsychology. In *Optimal Experience: Psychological Studies of Flow in Consciousness* (pp. 15–35). Cambridge University Press.

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

De Carli, D. M., Bevilacqua, F., Pozzer, C. T., & Cordeiro d'Ornellas, M. (2011). A survey of procedural content generation techniques suitable to game development. In *Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on* (pp. 26–35). IEEE.

Denisova, A., & Cairns, P. (2015). The Placebo Effect in Digital Games: Phantom Perception of Adaptive Artificial Intelligence (pp. 23–33). ACM Press. https://doi.org/10.1145/2793107.2793109

Dennaton Games. (2012). Hotline Miami [PlayStation 4, PlayStation Vita, PlayStation 3, Android, Microsoft Windows, Linux, Mac OS] Devolver Digital.

Dormans, J. (2010). Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the 2010 workshop on procedural content generation in games* (p. 1). ACM.

Dragert, C., Kienzle, J., Vangheluwe, H., & Verbrugge, C. (2011). *Generating extras: Procedural AI with statecharts*. Technical Report SOCS-TR-2011.1.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (Vol. 1, pp. 39–43). New York, NY.

Eberhart, R. C., & Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization. In *International Conference on Evolutionary Programming* (pp. 611–616). Springer.

Ebert, D. S. (2003). *Texturing & modeling: a procedural approach*. Morgan Kaufmann.

Edwards, R. (2006). The economics of game publishing. *IGN Entertainment Inc*.

Eladhari, M. P., Sullivan, A., Smith, G., & McCoy, J. (2011). *AI-Based game design: Enabling new playable experiences*. Technical Report, UCSC-SOE-11.

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, *19*(1), 43–53.

Evins, R. (2013). A review of computational optimisation methods applied to sustainable building design. *Renewable and Sustainable Energy Reviews*, *22*, 230–245.

Farnell, A. (2007). An introduction to procedural audio and its application in computer games. In *Audio mostly conference* (pp. 1–31).

Fernández-Vara, C., & Thomson, A. (2012). Procedural generation of narrative puzzles in adventure games: The puzzle-dice system. In *Proceedings of the The third workshop on Procedural Content Generation in Games* (p. 12). ACM.

Folmer, E. (2007). Component Based Game Development–A Solution to Escalating Costs and Expanding Deadlines? In *International Symposium on Component-Based Software Engineering* (pp. 66–73). Springer.

Frade, M., de Vega, F. F., & Cotta, C. (2010). Evolution of artificial terrains for video games based on accessibility. In *European Conference on the Applications of Evolutionary Computation* (pp. 90–99). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-12239-2_10

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, *20*(4), 74–94.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Greuter, S., Parker, J., Stewart, N., & Leach, G. (2003). Real-time procedural generation ofpseudo infinite'cities. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (p. 87–ff). ACM.

Haywood, N., & Cairns, P. (2006). Engagement with an interactive museum exhibit. In *People and Computers XIX—The Bigger Picture* (pp. 113–129). Springer. Retrieved from http://link.springer.com/chapter/10.1007/1-84628-249-7_8

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 (TOMM)*, *9*(1), 1.

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI* (Vol. 4, p. 1). Retrieved from http://www.aaai.org/Papers/Workshops/2004/WS-04-04/WS04-04-001.pdf

ID Software. (1992). Wolfenstein 3D [PC] Apogee Software

ID Software. (1993). Doom [PC]. GT Interactive

ID Software. (1996). Quake [PC]. GT Interactive

Ingber, L., & Rosen, B. (1992). Genetic algorithms and very fast simulated reannealing: A comparison. *Mathematical and Computer Modelling*, *16*(11), 87–100.

Isla, D. (2005, March 11). GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI. Retrieved from http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php

Jennett, C., Cox, A. L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T., & Walton, A. (2008). Measuring and defining the experience of immersion in games. *International Journal of Human-Computer Studies*, *66*(9), 641–661. https://doi.org/10.1016/j.ijhcs.2008.04.004

Jennings-Teats, M., Smith, G., & Wardrip-Fruin, N. (2010). Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (p. 11). ACM.

Johnson, L., Yannakakis, G. N., & Togelius, J. (2010). Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (p. 10). ACM.

Jörnmark, J., Axelsson, A.-S., & Ernkvist, M. (2005). Wherever Hardware, There'll be Games: The Evolution of Hardware and Shifting Industrial Leadership in the Gaming Industry. In *Proceedings of the Digital Games Research Association International Conference (DiGRA) 2005, Vancouver, Canada*.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., & others. (1983). Optimization by simmulated annealing. *Science*, *220*(4598), 671–680.

Kruse, J., & Connor, A. M. (2016). Multi-agent evolutionary systems for the generation of complex virtual worlds. *EAI Endorsed Transactions on Creative Technologies*, *2*(5).

Kushner, D. (2002). The wizardry of id [video games]. *IEEE Spectrum*, *39*(8), 42–47.

Lantz, F. (2004). Foreward. In K. Salen & E. Zimmerman, *Rules of play: Game design fundamentals*. Boston, MA: MIT press.

Lee, Y.-S., & Cho, S.-B. (2011). Context-aware petri net for dynamic procedural content generation in role-playing game. *IEEE Computational Intelligence Magazine*, *6*(2), 16–25.

Mourato, F., dos Santos, M. P., & Birra, F. (2011). Automatic level generation for platform videogames using genetic algorithms. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology* (p. 8). ACM.

Nelson, M. J. (2011). Game Metrics Without Players: Strategies for Understanding Game Artifacts. In *Artificial Intelligence in the Game Design Process*.

Nelson, M. J., & Mateas, M. (2008). Recombinable Game Mechanics for Automated Design Support. In *AIIDE*.

Nichols, N. (2014). *The Video Game Business*. London: British Film Institute.

Ong, T. J., Saunders, R., Keyser, J., & Leggett, J. J. (2005). Terrain generation using genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (pp. 1463–1470). ACM.

Pedersen, C., Togelius, J., & Yannakakis, G. N. (2010). Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, *2*(1), 54–67.

Pizzi, D., Lugrin, J.-L., Whittaker, A., & Cavazza, M. (2010). Automatic generation of game level solutions as storyboards. *IEEE Transactions on Computational Intelligence and AI in Games*, *2*(3), 149–161.

Raffe, W. L., Zambetta, F., Li, X., & Stanley, K. O. (2015). An Integrated Approach to Personalized Procedural Map Generation Using Evolutionary Algorithms. *IEEE Transactions on Computational Intelligence and AI in Games*, *7*(2), 139–155. https://doi.org/10.1109/TCIAIG.2014.2341665

Roberts, J., & Chen, K. (2015). Learning-Based Procedural Content Generation. *IEEE Transactions on Computational Intelligence and AI in Games*, *7*(1), 88–101. https://doi.org/10.1109/TCIAIG.2014.2335273

Rollings, A., & Morris, D. (2003). Game architecture and design: a new edition.

Simpson, C. (2014, July 17). Behaviour Trees for AI: How they work. Retrieved from http://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php

Singer, M. J., & Witmer, B. G. (1998). Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence: Teleoperators and Virtual Environments*, *7*(3), 225–240.

Smelik, R. M., Tutenel, T., de Kraker, K. J., & Bidarra, R. (2011). A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics*, *35*(2), 352–363.

Smith, A. M., & Mateas, M. (2011). Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 187–200.

Smith, A. M., Nelson, M. J., & Mateas, M. (2009). Computational Support for Play Testing Game Sketches. In *AIIDE*.

Smith, G., Gan, E., Othenin-Girard, A., & Whitehead, J. (2011). PCG-based game design: enabling new play experiences through procedural content generation. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games* (p. 7). ACM.

Smith, G., Whitehead, Ji., & Mateas, M. (2011). Tanagra: Reactive Planning and Constraint Solving for Mixed-Initiative Level Design. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 201–215.

Smuts, A. (2005). Are Video Games Art? *Contemporary Aesthetics*, *3*.

Squire, K. (2002). Cultural framing of computer/video games. *Game Studies*, *2*(1), 1–13.

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. N. (2011). What is procedural content generation?: Mario on the borderline. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games* (p. 3). ACM.

Togelius, J., Nelson, M. J., & Liapis, A. (2014). Characteristics of generatable games. *Intelligence*, *9*, 20.

Togelius, J., Preuss, M., & Yannakakis, G. N. (2010a). Towards multiobjective procedural map generation. In *Proceedings of the 2010 workshop on procedural content generation in games* (p. 3). ACM.

Togelius, J., Preuss, M., & Yannakakis, G. N. (2010b). Towards multiobjective procedural map generation. In *Proceedings of the 2010 workshop on procedural content generation in games* (p. 3). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1814259

Togelius, J., Shaker, N., & Nelson, M. J. (2016). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer.

Togelius, J., Yannakakis, G. N., & Liapis, A. (2013). Sentient Sketchbook: Computer-Aided Game Level Authoring. In *Proceedings of the 8th Conference on the Foundations of Digital Games* (pp. 213–220).

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172–186. https://doi.org/10.1109/TCIAIG.2011.2148116

Unity. (2016). (Version 5). Unity Technologies. Retrieved from https://unity3d.com/

van der Linden, R., Lopes, R., & Bidarra, R. (2014). Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, *6*(1), 78–89.

Weibel, D., & Wissmath, B. (2011). Immersion in computer games: The role of spatial presence and flow. *International Journal of Computer Games Technology*, *2011*, 6.

Williams, D. (2002). Structure and competition in the US home video game industry. *International Journal on Media Management*, *4*(1), 41–54.

Woodcock, S. (2000, November 1). Game AI: The State of the Industry. Retrieved April 29, 2016, from http://www.gamasutra.com/view/feature/131975/game_ai_the_state_of_the_industry.php

Yannakakis, G. N. (2012). Game AI revisited. In *Proceedings of the 9th conference on Computing Frontiers* (pp. 285–292). ACM.

Yannakakis, G. N., & Togelius, J. (2015a). A Panorama of Artificial and Computational Intelligence in Games. *IEEE Transactions on Computational Intelligence and AI in Games*, *7*(4), 317–335. https://doi.org/10.1109/TCIAIG.2014.2339221

Yannakakis, G. N., & Togelius, J. (2015b). Experience-Driven Procedural Content Generation (Extended Abstract) (pp. 519–525). Presented at the International Conference on Affective Computing and Intelligent Interaction (ACII).

Yoon, D., & Kim, K.-J. (2012). 3D game model and texture generation using interactive genetic algorithm. In *Proceedings of the Workshop at SIGGRAPH Asia* (pp. 53–58). ACM.

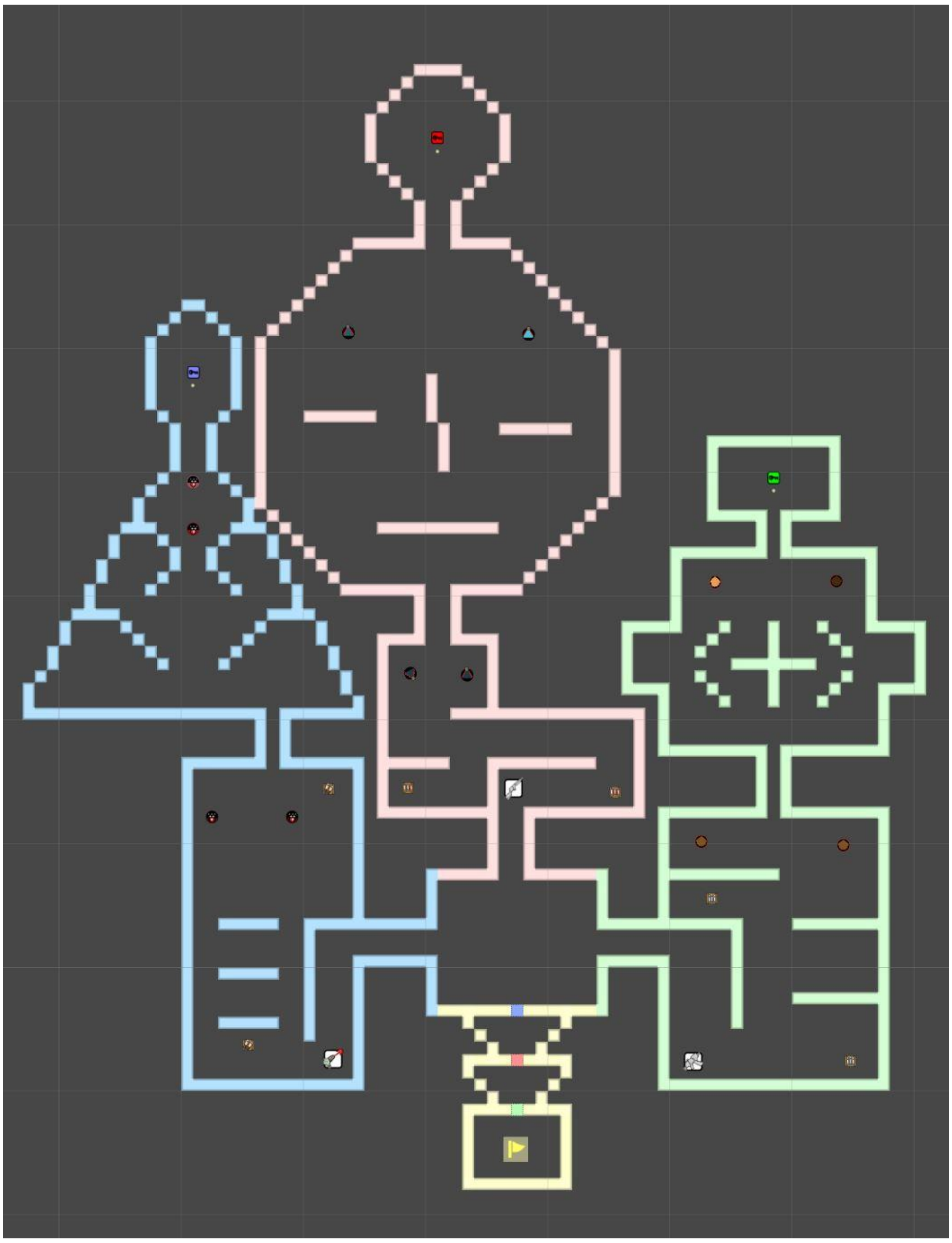
Yue, B., & de-Byl, P. (2006). The state of the art in game AI standardisation. In *Proceedings of the 2006 international conference on Game research and development* (pp. 41–46). Murdoch University.

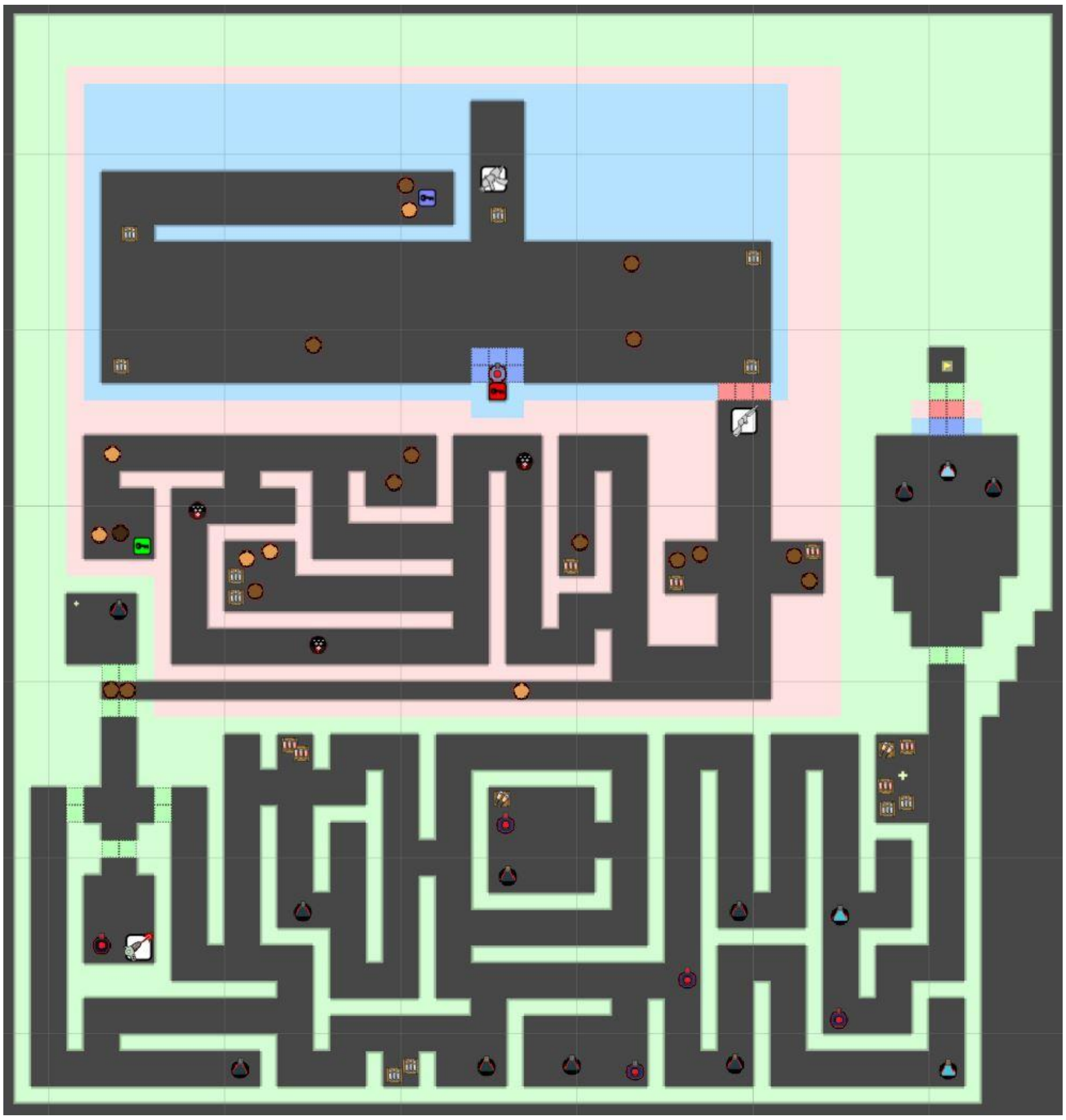
Yuret, D., & De La Maza, M. (1993). Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *The Second Turkish Symposium on Artificial Intelligence and Neural Networks* (pp. 208–212). Citeseer.

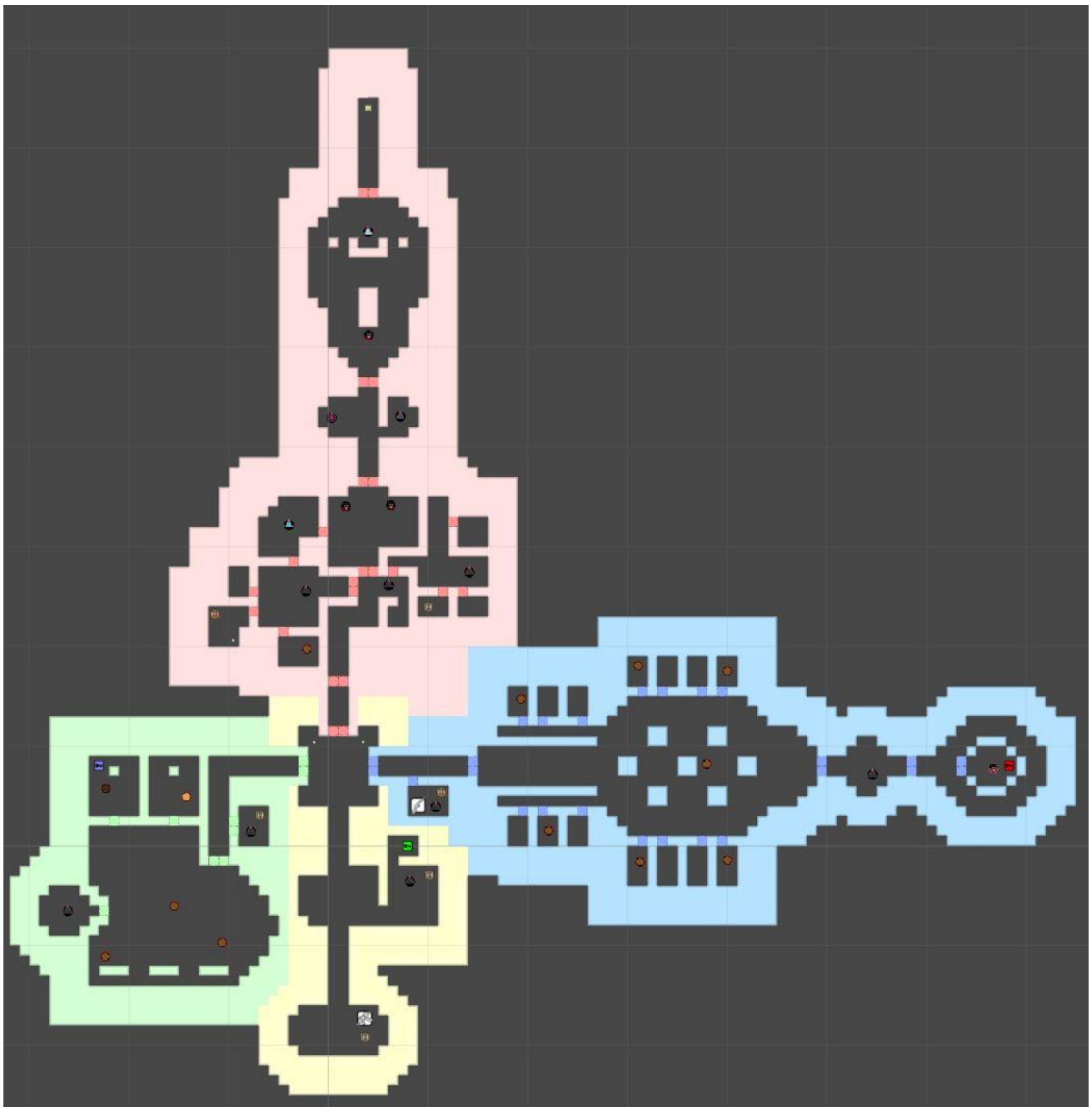
Zook, A., & Riedl, M. O. (2014). Automatic game design via mechanic generation. In *AAAI* (pp. 530–537).

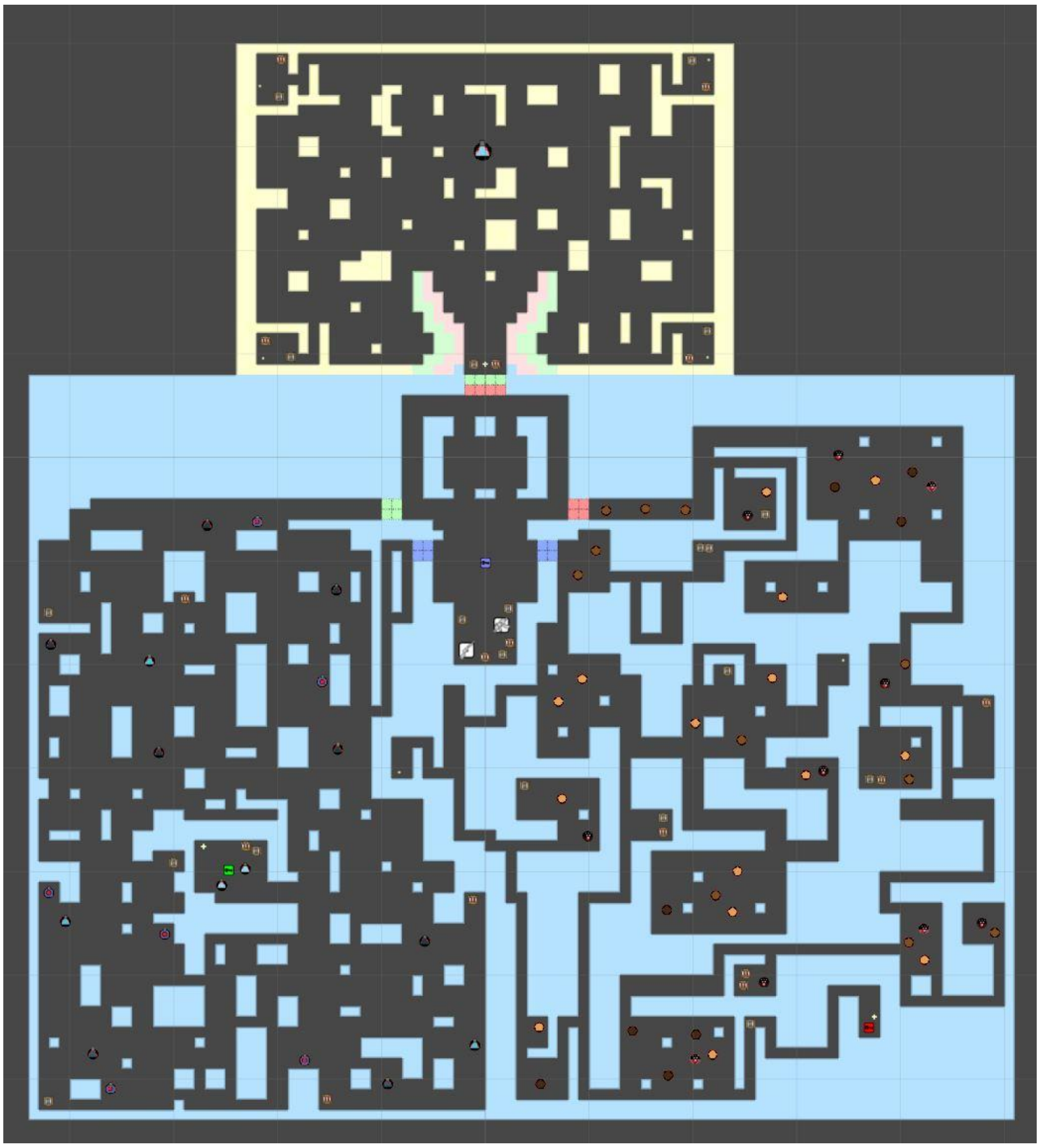
Appendix A: Evolved Levels and Human Levels

A.1 Human Designed Level Maps

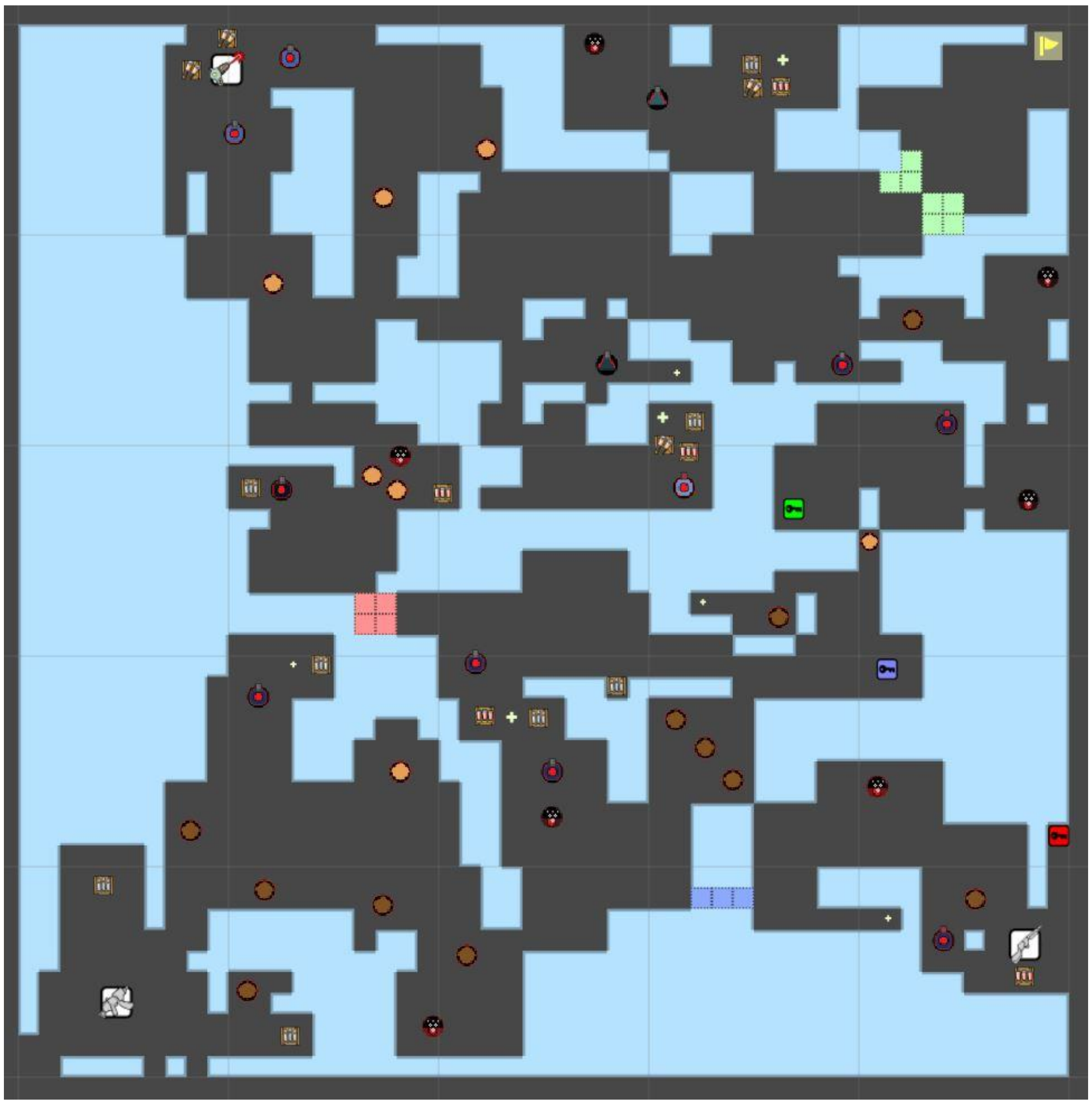


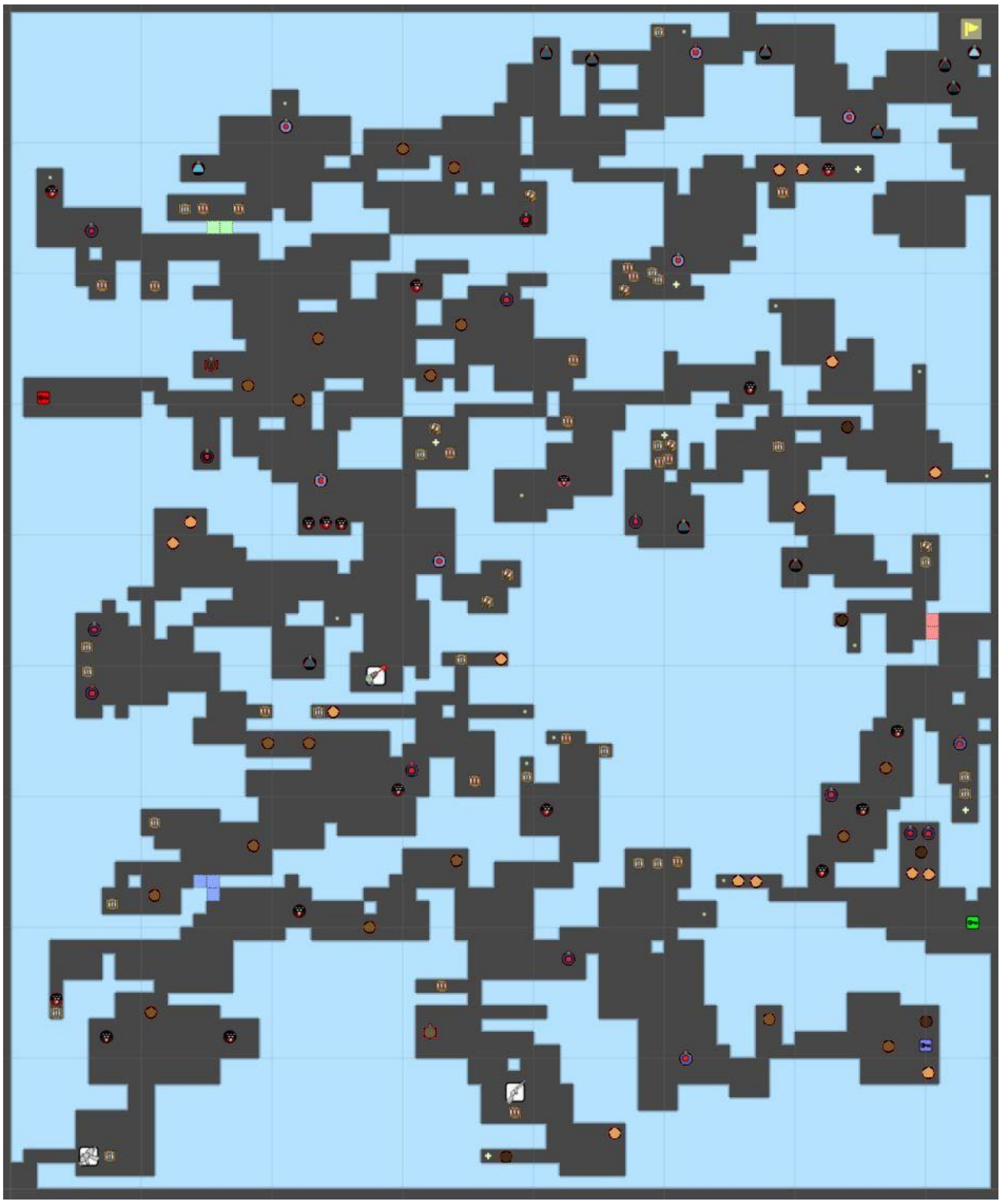


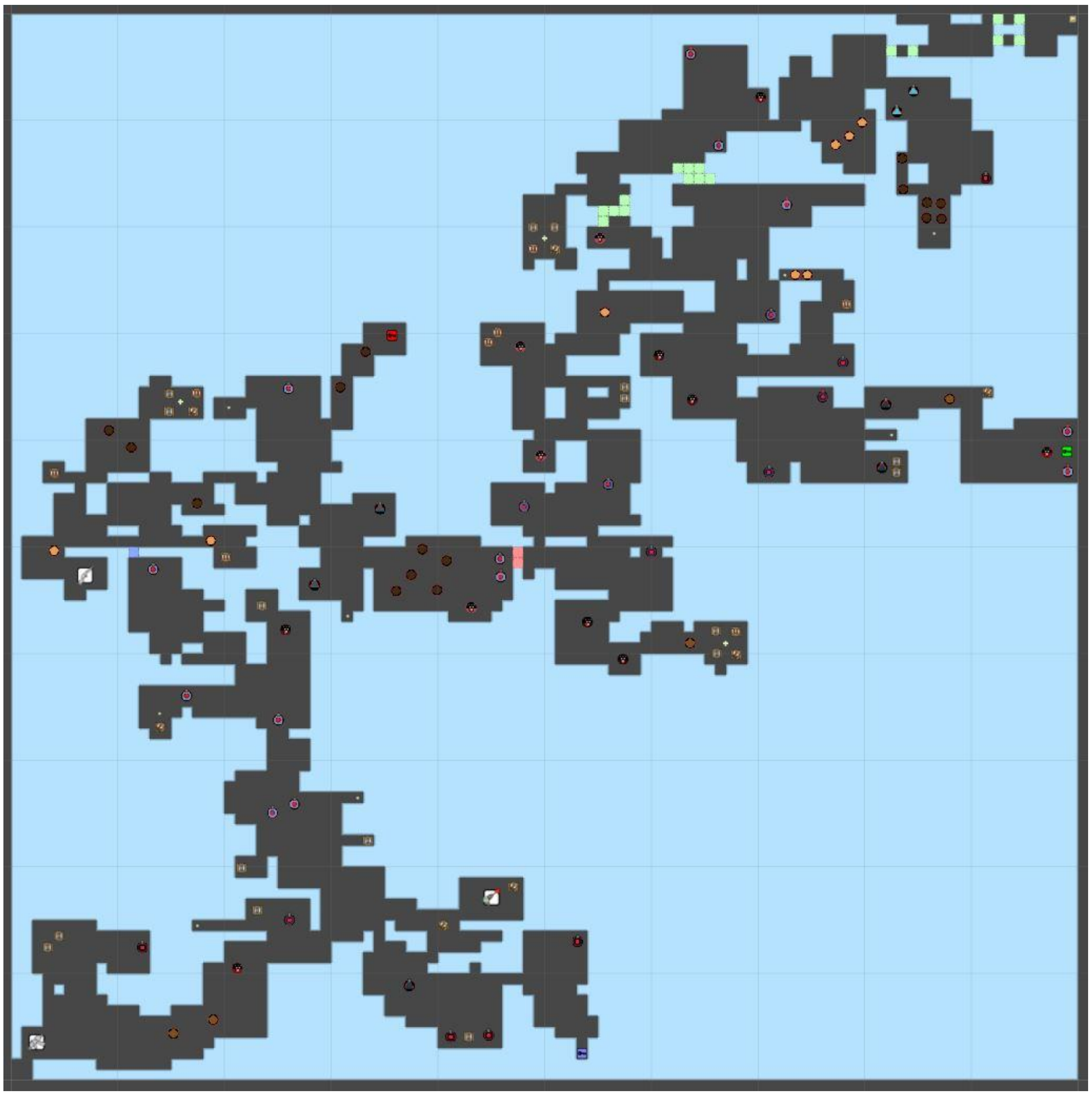


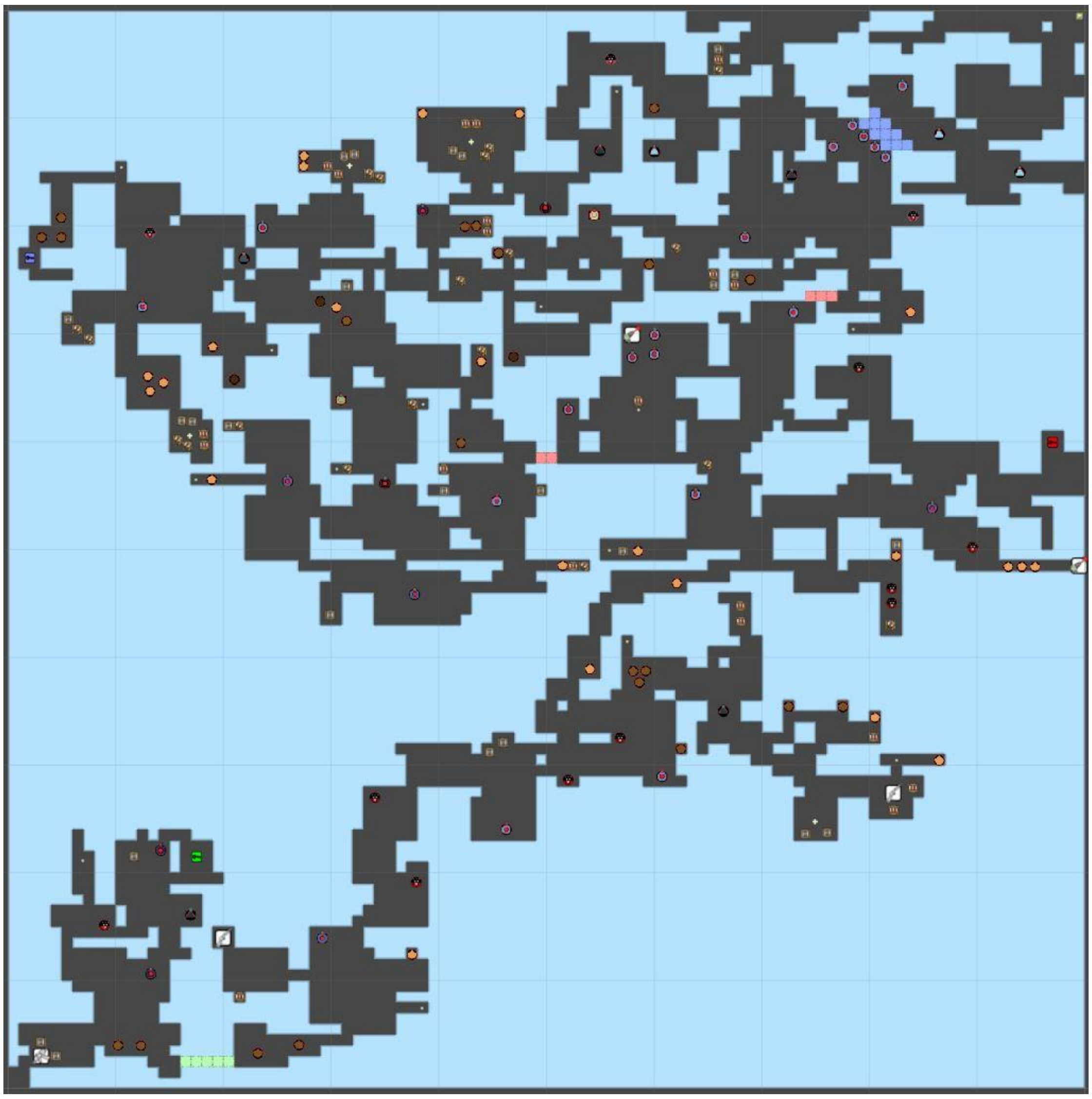


A.2 Procedurally Generated Level Maps









Appendix B: Questionnaires

B.1 Pre-Participation Questionnaire

SECTION 1: PERSONAL DETAILS					
1.1	Please indicate your age range				
	○ 18-21	○ 26-30	○ 36-40		
	○ 22-25	○ 31-35	○ 41-50		
	○ 50+				
1.2	Please indicate your gender				
	○ Male	○ Female			
SECTION 2: GAMING EXPERIENCE					
2.1	How many years have you been playing video games?				
	○ 0-2	○ 3-4	○ 5-6		
	○ 6-8	○ 8-10	○ 10+		
2.2	What platforms do you play on? Choose all that apply.				
	○ Xbox One	○ PS3	○ PS4		
	○ Xbox 360	○ Wii	○ Wii U		
	○ PC	○ Other (Please specify)			
2.3	How often do you play video games?				
	○ Daily	○ Weekly	○ Monthly		
	○ Rarely	○ Never			
2.4	How long do you play games in each play session?				
	○ 0-15 minutes	○ 15-30 minutes	○ 30-60 minutes		
	○ 1-2 hours	○ 2-3 hours	○ 3+ hours		
2.5	To what extent do you enjoy playing video games?				
	Not at all				Very Much
	1	2	3	4	5
	○	○	○	○	○

2.6	Do you play more than one game?		
	○ Yes	○ No	
2.7	What game are you currently playing the most?		
2.8	What genres of games are you most familiar with? Choose all that apply.		
	○ Action	○ Shooter	○ Action-Adventure
	○ Adventure	○ Role Playing	○ Simulation
	○ Strategy	○ Sports	○ Arcade
	○ Mystery	○ Other (Please specify)	
2.9	What is your favourite genre of game? Choose just one.		
	○ Action	○ Shooter	○ Action-Adventure
	○ Adventure	○ Role Playing	○ Simulation
	○ Strategy	○ Sports	○ Arcade
	○ Mystery	○ Other (Please specify)	
2.10	What is your favourite game of all time?		

B.2 Participation Questionnaire

Please answer the following questions that relate to your game playing experience.

1 To what extent did the game hold your attention?

Not at all				A lot
1	2	3	4	5
○	○	○	○	○

2 To what extent did you feel you were focused on the game?

Not at all				A lot
1	2	3	4	5
○	○	○	○	○

3 How much effort did you put into playing the game?

Very little				A lot
1	2	3	4	5
○	○	○	○	○

4 To what extent did you lose track of time?

Not at all				A lot
1	2	3	4	5
○	○	○	○	○

5 To what extent did you feel consciously aware of being in the real world whilst playing?

Not at all				Very much so
1	2	3	4	5
○	○	○	○	○

6 To what extent did you forget about your everyday concerns?

Not at all				Very much so
1	2	3	4	5
○	○	○	○	○

7	To what extent were you aware of yourself in your surroundings?				
	Not at all				Very aware
	1	2	3	4	5
	○	○	○	○	○
8	To what extent did you notice events taking place around you?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○
9	Did you feel the urge at any point to stop playing and see what was happening around you?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
10	To what extent did you feel that you were interacting with the game environment?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
11	To what extent did you feel as though you were separated from your real-world environment?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
12	To what extent did you feel that the game was something you were experiencing, rather than something you were just doing?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○

13	To what extent was your sense of being in the game environment stronger than your sense of being in the real world?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
14	At any point did you find yourself become so involved that you were unaware you were even using controls?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
15	To what extent did you feel as though you were moving through the game according to you own will?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
16	To what extent did you find the game challenging				
	Not at all				Very difficult
	1	2	3	4	5
	○	○	○	○	○
17	Were there any times during the game in which you just wanted to give up?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○
18	To what extent did you feel motivated while playing?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○

19	To what extent did you find the game easy?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
20	To what extent did you feel like you were making progress towards the end of the game?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○
21	How well do you think you performed in the game?				
	Very poor				Very well
	1	2	3	4	5
	○	○	○	○	○
22	To what extent did you feel emotionally attached to the game?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
23	To what extent were you interested in seeing how the game's events would progress?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○
24	How much did you want to "win" the game?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
25	Were you in suspense about whether or not you would win or lose the game?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○

26	At any point did you find yourself become so involved that you wanted to speak to the game directly?				
	Not a lot				Very much so
	1	2	3	4	5
	○	○	○	○	○
27	To what extent did you enjoy the graphics and the imagery?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○
28	How much would you say you enjoyed playing the game?				
	Not at all				A lot
	1	2	3	4	5
	○	○	○	○	○
29	When interrupted, were you disappointed that the game was over?				
	Not at all				Very much so
	1	2	3	4	5
	○	○	○	○	○
30	Would you like to play the game again?				
	Definitely not				Definitely yes
	1	2	3	4	5
	○	○	○	○	○

.

Appendix C: ANOVA Results for individual questions

Anova: Single Factor (Q1)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	35	3.888889	1.361111
Column 2	7	31	4.428571	0.619048

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.146825	1	1.146825	1.099457	0.312151	4.60011
Within Groups	14.60317	14	1.043084			
Total	15.75	15				

Anova: Single Factor (Q2)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	37	4.111111	1.111111
Column 2	7	29	4.142857	1.142857

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.003968	1	0.003968	0.003528	0.953474	4.60011
Within Groups	15.74603	14	1.124717			
Total	15.75	15				

Anova: Single Factor (Q3)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	26	2.888889	1.111111
Column 2	7	28	4	0.666667

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	4.861111	1	4.861111	5.280172	0.037502	4.60011
Within Groups	12.88889	14	0.920635			
Total	17.75	15				

Anova: Single Factor (Q4)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	29	3.222222	0.444444
Column 2	7	33	4.714286	0.238095

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	8.765873	1	8.765873	24.62261	0.000209	4.60011
Within Groups	4.984127	14	0.356009			
Total	13.75	15				

Anova: Single Factor (Q5)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	23	2.555556	2.027778
Column 2	7	24	3.428571	1.285714

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3.000992	1	3.000992	1.755222	0.20644	4.60011
Within Groups	23.93651	14	1.709751			
Total	26.9375	15				

Anova: Single Factor (Q6)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	34	3.777778	1.694444
Column 2	7	33	4.714286	0.571429

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3.453373	1	3.453373	2.846612	0.113711	4.60011
Within Groups	16.98413	14	1.213152			
Total	20.4375	15				

Anova: Single Factor (Q7)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	26	2.888889	1.861111
Column 2	7	25	3.571429	0.285714

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.834325	1	1.834325	1.546726	0.234042	4.60011
Within Groups	16.60317	14	1.185941			
Total	18.4375	15				

Anova: Single Factor (Q8)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	33	3.666667	1.5
Column 2	7	28	4	1.333333

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.4375	1	0.4375	0.30625	0.58872	4.60011
Within Groups	20	14	1.428571			
Total	20.4375	15				

Anova: Single Factor (Q9)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	35	3.888889	1.611111
Column 2	7	31	4.428571	0.952381

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.146825	1	1.146825	0.863055	0.368627	4.60011
Within Groups	18.60317	14	1.328798			
Total	19.75	15				

Anova: Single Factor (Q10)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	32	3.555556	1.277778
Column 2	7	26	3.714286	1.238095

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.099206	1	0.099206	0.078687	0.783187	4.60011
Within Groups	17.65079	14	1.260771			
Total	17.75	15				

Anova: Single Factor (Q11)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	29	3.222222	1.694444
Column 2	7	25	3.571429	1.285714

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.480159	1	0.480159	0.316045	0.582884	4.60011
Within Groups	21.26984	14	1.519274			
Total	21.75	15				

Anova: Single Factor (Q12)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	28	3.111111	1.361111
Column 2	7	23	3.285714	0.571429

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.12004	1	0.12004	0.117378	0.736985	4.60011
Within Groups	14.31746	14	1.022676			
Total	14.4375	15				

Anova: Single Factor (Q13)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	26	2.888889	2.361111
Column 2	7	26	3.714286	0.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2.68254	1	2.68254	1.544386	0.234378	4.60011
Within Groups	24.31746	14	1.736961			
Total	27	15				

Anova: Single Factor (Q14)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	28	3.111111	2.111111
Column 2	7	28	4	1.333333

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3.111111	1	3.111111	1.75	0.207078	4.60011
Within Groups	24.88889	14	1.777778			
Total	28	15				

Anova: Single Factor (Q15)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	34	3.777778	1.444444
Column 2	7	27	3.857143	0.47619

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.024802	1	0.024802	0.024091	0.878868	4.60011
Within Groups	14.4127	14	1.029478			
Total	14.4375	15				

Anova: Single Factor (Q16)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	22	2.444444	1.527778
Column 2	7	19	2.714286	1.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.286706	1	0.286706	0.169715	0.686604	4.60011
Within Groups	23.65079	14	1.689342			
Total	23.9375	15				

Anova: Single Factor (Q17)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	40	4.444444	0.777778
Column 2	7	32	4.571429	0.619048

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.063492	1	0.063492	0.089457	0.769263	4.60011
Within Groups	9.936508	14	0.709751			
Total	10	15				

Anova: Single Factor (Q18)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	28	3.111111	0.861111
Column 2	7	28	4	1

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3.111111	1	3.111111	3.37931	0.087328	4.60011
Within Groups	12.88889	14	0.920635			
Total	16	15				

Anova: Single Factor (Q19)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	18	2	0.75
Column 2	7	19	2.714286	1.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2.008929	1	2.008929	1.61373	0.224675	4.60011
Within Groups	17.42857	14	1.244898			
Total	19.4375	15				

Anova: Single Factor (Q20)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	29	3.222222	1.194444
Column 2	7	24	3.428571	0.952381

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.167659	1	0.167659	0.153716	0.700913	4.60011
Within Groups	15.26984	14	1.090703			
Total	15.4375	15				

Anova: Single Factor (Q21)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	29	3.222222	0.944444
Column 2	7	21	3	1.333333

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.194444	1	0.194444	0.175	0.682052	4.60011
Within Groups	15.55556	14	1.111111			
Total	15.75	15				

Anova: Single Factor (Q22)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	16	1.777778	0.694444
Column 2	7	17	2.428571	0.952381

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.667659	1	1.667659	2.071655	0.172042	4.60011
Within Groups	11.26984	14	0.804989			
Total	12.9375	15				

Anova: Single Factor (Q23)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	25	2.777778	0.944444
Column 2	7	29	4.142857	0.47619

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	7.337302	1	7.337302	9.865091	0.007221	4.60011
Within Groups	10.4127	14	0.743764			
Total	17.75	15				

Anova: Single Factor (Q24)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	32	3.555556	1.777778
Column 2	7	33	4.714286	0.571429

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	5.286706	1	5.286706	4.193233	0.059827	4.60011
Within Groups	17.65079	14	1.260771			
Total	22.9375	15				

Anova: Single Factor (Q25)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	28	3.111111	1.361111
Column 2	7	23	3.285714	1.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.12004	1	0.12004	0.075302	0.787775	4.60011
Within Groups	22.31746	14	1.594104			
Total	22.4375	15				

Anova: Single Factor (Q26)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	25	2.777778	2.694444
Column 2	7	20	2.857143	2.142857

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.024802	1	0.024802	0.01009	0.921412	4.60011
Within Groups	34.4127	14	2.45805			
Total	34.4375	15				

Anova: Single Factor (Q27)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	24	2.666667	0.25
Column 2	7	19	2.714286	1.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.008929	1	0.008929	0.009309	0.924506	4.60011
Within Groups	13.42857	14	0.959184			
Total	13.4375	15				

Anova: Single Factor (Q28)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	31	3.444444	0.277778
Column 2	7	30	4.285714	0.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2.786706	1	2.786706	5.099326	0.040426	4.60011
Within Groups	7.650794	14	0.546485			
Total	10.4375	15				

Anova: Single Factor (Q29)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	24	2.666667	0.75
Column 2	7	23	3.285714	0.571429

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.508929	1	1.508929	2.24053	0.156637	4.60011
Within Groups	9.428571	14	0.673469			
Total	10.9375	15				

Anova: Single Factor (Q30)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Column 1	9	25	2.777778	0.944444
Column 2	7	30	4.285714	0.904762

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	8.953373	1	8.953373	9.653881	0.007723	4.60011
Within Groups	12.98413	14	0.927438			
Total	21.9375	15				