Designer-driven Procedural Game Content Generation using Multi-Agent Evolutionary Computation

Jan Kruse

A thesis submitted to

Auckland University of Technology

in fulfilment of the requirements for the degree

of

Doctor of Philosophy

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 acknowledgments), 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.

Auckland, 15th May 2019

_____

Acknowledgements

First and foremost, thank you, Jeannine. You started it all. But you are also my soulmate and, by an almost infinite margin, the most important person in the world for me.

Second, thank you Andy and Stefan. You are a fantastic supervision team, allowing me to walk the thin line between Design and Computer Science, guiding me, pushing me, supporting me.

I also thank Lynne Jamneck for proof-reading my thesis. I feel you did an outstanding job.

A special thanks to Greg for supporting me in various ways. You are a great Head of Department.

Thank you everyone else for your help and for making this great journey possible.

Thank you AUT and in particular the Vice-Chancellery for awarding me the Doctoral Study Award, which fully funded the last leg of my journey.

This research has been approved by the ethics committee (AUTEC) under 17/80.

Table of Contents

Table of Figures

Abbreviations

AI	Artificial Intelligence
CS:GO	Counter-Strike: Global Offensive (Game)
DDA	Digital Differential Analysis
DEA	Design Expert Agent
EC	Evolutionary Computation, the field of computer science using biologically inspired search and optimization techniques
GA	Genetic Algorithm, probably the most common Evolutionary Computation algorithm
HBGA	Human-based Genetic Algorithm, a multi-agent system replacing recombination and selection with human and/or computational agents
IEC	Interactive Evolutionary Computation, biologically inspired search and optimization techniques with a human user typically replacing a mathematical fitness function
IGA	Interactive Genetic Algorithm, Genetic Algorithm with a human user typically replacing a mathematical fitness function
MVP	Minimum Viable Product
NPC	Non Player Character
PCG	Procedural Content Generation
SDK	Software Development Kit

Abstract

Creating computer game levels that offer good playability and interesting layouts is a laborious and costly task. In particular, multiplayer game levels necessitate careful balancing of gameplay between teams and clear objectives for players. Expert knowledge that draws on a good understanding of player experience and strong level design skills results in maps that players enjoy and appreciate. The success of new level designs hinges on this expertise.

This study introduces a Multi-Agent System based on heuristics developed from expert level designers, that is able to augment game designers of all levels of expertise and help them create First Person Shooter levels with good playability. An interactive evolutionary algorithm is employed to provide designers with several level design options. The human user stays in full control of the decisions made during the design process, while the agent system provides suggestions that promise good playability and a positive player experience.

The system has been evaluated using game designers with varying levels of experience, and the results show great promise. A Multi-Agent System in addition to the Interactive Genetic Algorithm outperforms a purely human-centric solution: Game designers of all skill levels draw heavily on the suggestions made by the Multi-Agent System. Recommendations for future developments are given.

Foreword

Before I dive straight into the topic of this thesis, let me explain my motivation for completing this project.

When I handed in the thesis for my Master of Philosophy (MPhil), which explored novel ways for approaching the computational design of virtual environments, I felt that my research was unfinished. This was partly because I discovered that many aspects of the research had to be relegated to a *future research* section, due to the nature and scope of the degree. It was also because I did not realise at the time that my interest in the possibilities at the boundaries of design, visual effects, computer games, and computer science was only part of a much longer research trajectory. Now that I am writing this introduction, I feel the same yet again. The work that I am about to present herein is only a small puzzle piece, part of a longer quest with the aim of making machines help us work smarter in design processes, or in this specific case, creating machines that can help us work smarter when designing computer game content. I do not buy into the idea that machines will eventually replace us in all employment areas. I strongly believe that we are simply going to continue using machines to make our work a bit easier.

In this thesis, I will touch on a small number of ideas carried over from my MPhil. I will, however, also consider how we *play* games, and what it means to design a 'playable computer game', in a bid to better understand how cognitive models can help designers make game content-creation somewhat easier. The focus is on player-centric design with the help of computational tools. To this extent, this thesis addresses the relationship between game designers and players through the game itself. It also discusses an attempt to address the conceptual and concrete development of collaborative design tools that help create and evaluate procedural game content. As would be expected of research in a rapidly developing field, this body of work has a purposefully narrow scope, and the assumptions made are relatively simple abstractions. These abstractions, however, are not only required by the scope of this study, but also provide useful confines that may help to generalise some of the findings, so that these contributions can be rendered applicable to a wider range of evaluation problems in game level design.

It goes without saying that this thesis is only a partial reflection of what the overall journey had to offer. This document describes a consistent research project that makes a novel contribution to knowledge, but there is much more to these results than what is presented in this single research project. I learned a great deal about knowing nothing, and realised how little I knew in relation to subjects I considered myself to be well-informed about. This is both a profound and yet very mundane experience that many PhD students likely have. I also learned to expand and develop algorithms in a much deeper way than ever before. At the start of this journey, I experimented significantly. I tested agents that could drive virtual cars in a game engine, and neural networks that learned to fly spacecraft and perform manoeuvres without any user interaction or interference. I also made and played many test game levels with a number of friends, colleagues, and students in order to understand the difference between an awfully designed, virtually unplayable level, and a prototype level that was actually enjoyable despite its simplicity. I also now understand why a solid foundation in theory is the best grounding for successful practice – although I will never claim to have reached a sufficient theoretical level, as I am more aware of my own limitations the more I read and learn. However, despite all these minor (or personally, major) achievements, I feel that this journey is only the beginning of a longer trajectory. I hope this study serves as a solid stepping stone for future research (ad)ventures.

1 Introduction

This thesis investigates the performance of a multi-agent system that augments a human user in a game level design task, based on an interactive genetic algorithm. The agents assess procedurally generated levels and make suggestions to the user in order to accelerate the speed of the process. The work takes an in-depth look at how designers think and feel about the interaction with the system.

1.1 Problem Statement

Immersive games, movies, and virtual realities necessitate convincing environments. Creating high-quality levels (or maps) for computer games requires substantial skill and time, and is therefore subject to high costs and typically numerous iterations as part of the design cycle. In fact, the cost and complexity of computer game development has been identified as being at least as high, if not higher, than any other software development project (Musil, Schweda, Winkler & Biffl, 2010). Cost from one generation of games to the next has repeatedly doubled (Edwards, 2006), and a recent survey found that top-end mobile titles require $5m to $20m[1] investment, while AAA console and PC releases go up in cost tenfold every decade (Koster, 2018). Large computer graphics models from areas such as architecture and biology necessitate complex detail and a large number of small building blocks, which further intensifies the cost issue. In addition to their generation, models and game levels require testing as a whole to address any problems (bugs), and to ensure their playability, prior to being released to the customer, which adds cost and time. At the centre of these undertakings is the game designer, or in larger project, a specific game level designer. The game level designer seeks to take all aforementioned factors into account in order to produce levels that are not only playable, but enjoyable for players. My research aims to augment the designer by automating some of the repetitive tasks during the design phase of FPS levels. The study seeks to reduce the time required by putting the designer into a team of software agents, and creating a hybrid human computational multi-agent system. Moreover, the study seeks to capture the (human) designer's intent in the automated process; instead of rationalising designers out of the system, they become a more significant and integral part of it. Hence,

[1] All in US Dollar

this study attempts to create procedural generation and evaluation of artefacts, which closes the feedback loop of the design cycle.

1.2 Research Question

This study is driven by the following research question:

> How can intelligent agent systems be employed to generate, and more importantly, evaluate procedural game content? Specifically, can agents based on cognitive models evaluate computational designs in computer games?

The experimental design aims to serve as a research platform for allowing me to respond to this question, and ultimately, to make a contribution to existing knowledge by testing computational cognitive agents as part of a multi-agent system, which augments a game level designer while conducting design tasks.

1.3 Research Approach

In this study, a mixed methods research approach is adopted. At the core of this research rests a qualitative inquiry using semi-structured interviews to understand the designer's views in relation to the multi-agent system employed in this study. Expert accounts of their design process and similar literature have been used to develop heuristics that guide the evaluation of computational design within the multi-agent system. In order to triangulate the qualitative findings and possibly arrive at a more rigorous and defendable result, quantitative data from various sources such as designer mouse and keyboard inputs, as well as gaze-tracking data is used for the evaluation of this study.

1.4 Contributions

This thesis makes a number of contributions to knowledge and as a result provides insight to where further research could be valuable. The main contribution of this study is the development and evaluation of a multi-agent system with human designers (as human agents) in full control of the overall design process. These systems show great promise as a tool for augmenting the designer's abilities, while allowing them to pursue an individual goal, as opposed to simply following a generic algorithmic pattern. It may also help designers focus on tasks they actually prefer to do, instead of spending significant time on administrative tasks and testing. Laborious and costly, repetitive testing and iterations can at least to some extent be replaced by computational tools.

The study also collects a wealth of qualitative data in form of interview responses from expert game designers, which provides an insight into how experts think and feel about semi-automated design systems.

The use of cognitive models based on heuristics derived from expert designers, has recently been identified as representing a gap in the literature (Zhu, Zhao, Fang & Moser, 2017). This thesis makes an attempt to fill this gap. The workflow for creating these models is straightforward but quite effective and can likely be used not only in this specific case, but also in other design contexts.

Another contribution can be found in the overall design system, which adds agent-based evaluation to the procedural content generation process. This effectively closes the design cycle and opens up new possibilities for computational game creativity. Although there have been recent attempts to create evaluators for computational game content, evaluation of design elements has been acknowledged as an area of ongoing research (Liapis, Yannakakis & Togelius, 2014). Even the advent of highly capable computational content creators such as generative adversarial networks (GAN) using deep neural networks (Giacomello, Lanzi & Loiacono, 2018) present limitations, given that GANs are not able to determine the quality of their created content using computational means. This thesis makes a contribution to this field by presenting a novel way for modelling evaluation agents in computational game level design systems.

While not an original contribution in itself, the use of an eye tracking device (and additional mouse and keyboard input recordings) highlights and confirms the necessity for triangulating qualitative data. Experts provided interview responses, which were reasonably consistent across the board. However, quantitative data indicates significant contradictions among verbal statements. This is particularly true for experts who have a long-running track record. It seems that learned behaviour and confidence based on long-term professional experience may cloud the judgement of what an individual relies on during their design processes, and this can in turn lead to potentially questionable interview responses. Subsequently, it can also potentially lead to questionable design decisions. Recordings of what individuals actually look at and do while they are performing a given task can only be reliably obtained through triangulation. Potential biases can easily be identified through the use of eye tracking and input recordings.

While most of the building blocks that are being employed in this study can be found in the form of existing software libraries, the prototype was largely purpose-built. While it could be argued that doing so represents a significant and time-consuming undertaking, it proved to be much easier than using existing libraries, due to the necessity for enabling these building blocks – including several computational agents, a human user, and a genetic algorithm – to interact with each other, and to exchange information in real-time.

1.5 Structure of this Thesis

The purpose of this first chapter is to contextualise the research and outline the research strategy. It frames the research problem and research question, introduces the applied research methodology, and indicates the limitations of this study.

Chapter 2 provides an overview of relevant work conducted in the areas of game design, procedural content generation, and evaluation in computer games. This chapter also discusses playability and player experience goals.

Chapter 3 unpacks the methodology and research design of this study. Questionnaires, semi-structured interviews and telemetry, as well as eye tracking, are discussed.

Chapter 4 describes the prototype implementation and its fundamental principles. It also presents an overview of the tools used to conduct this research.

Chapter 5 discusses the results and observations of this study, including interview responses and quantitative data derived from eye tracking and user input.

Chapter 6 provides an overall conclusion to this thesis including findings, limitations, and contributions, and indicates possible future research directions.

2 Literature Review

This chapter discusses game design and computer science literature relevant to automated game content generation. Aspects of how the quality of such game content can be evaluated is also discussed. Given that this thesis attempts to understand how designers can be empowered through generative design tools, both the (game) design process, as well as the tools and underlying algorithms, need to be examined.

The study seeks to find tools that can augment the designer's ability by automating some of the time-consuming processes in game level design, allowing the designer to focus on the core task, namely, creating playable and enjoyable game levels. This chapter provides an overview of the existing literature in the areas of procedural content generation and in search and optimisation algorithms. This includes the use of genetic algorithms in purely computational, interactive (human-based), and finally, hybrid form, the latter using both human and computational resources to produce game content.

The literature review further reviews existing hybrid human and computational systems that allow a user not only to interact with an application, but control its outputs by being an active part of its algorithmic system: here, the user has a direct impact on the performance of the algorithm, and becomes an integral part of the solution, rather than merely using a solution produced by a computational process. Existing systems are discussed in section 2.5 of this chapter, which leads to one of the primary contributions of this thesis: a novel multi-agent system specifically created for design purposes, by changing and expanding the existing systems to generate and also evaluate procedural content. The multi-agent system is grounded in Kosorukoff's (2001) human-based genetic algorithm (HBGA). HBGAs are fundamentally multi-agent systems (MAS) in which human and computational agents share tasks or complement each other while undertaking different tasks. This study can be considered a HBGA in human-centric form, focussed on a game content designer. However, given that the literature does not consistently use Kosorukoff's terminology, I will use the term 'multi-agent system'. Additionally, Kosorukoff suggests the user can be part of the selection process and the mutation process; however, in this study, the user is exclusively part of the selection process, and accordingly, I choose to use the more common terminology.

Generative design in general, and procedural content generation (PCG) and evaluation in particular will first be discussed. PCG is the core output of the software prototype used to collect data from participants, which highlights its significance within this research project. This thesis applies a player-centric lens to game design, which drives the choice of literature to some extent. The condensed and focused approach is quite deliberate, predominantly for pragmatic reasons; practically, it keeps the focus of this thesis restricted, which allows for a more in-depth consideration of this specific area. Following the philosophical viewpoints of Tracy Fullterton's player experience-based game design (Fullerton, 2008), explicit aims have been developed and applied to the prototyping of both the design software and the actual resulting prototype game. Player experience-based content generation, also known as adaptive content generation (Shaker, Togelius, & Nelson, 2016b), refers to the paradigm where previous player behaviour or experience shapes content creation. Player experience goals specific to this study are outlined in section 4.1.4.

Further, genetic algorithms are discussed as part of the literature. The genetic algorithm is the core of the multi-agent system and links the individual agents together. Furthermore, agents are examined and specific definitions of what constitutes an agent are reviewed. Finally, a brief consideration of some of the underlying concepts for agent creation and data collection, such as ray-casting and eye tracking, is presented.

2.1 Game Design

An unavoidable truth should be stated at the start of this section: a comprehensive and all-encompassing review of game design literature is neither possible within the scope of this thesis, nor is it desirable, as the current study can only engage with a very narrow part of the broad field of game design. However, this is not necessarily a limitation, but rather a disclaimer, as a full and complete game design literature review is not needed to address the research conducted as part of this study, or to identify the gap in the literature that this thesis seeks to address. This study focuses on procedural game content generation and evaluation. Therefore, a large proportion of this review is dedicated to computational topics. However, a number of important aspects of game design, and in a broader sense, design literature, must naturally be discussed. This section of the literature review takes a brief look at the design process and game design process first, followed by a deeper examination of procedural game content generation and evaluation.

2.1.1 Game Elements

There have been numerous attempts to provide a fundamental theory of (computer) games (for example Freyermuth, 2015; Järvinen, 2009; Jenkins, 2004; Juul, 2011), and a discussion of this in any great depth would be sufficient for the scope of a thesis itself. Therefore, the conversation around definitions of computer games and theories of games will be avoided here. However, to understand what procedural content generation means in the context of this study, a brief consideration of game elements relevant to this study seems appropriate. A detailed overview of the implementation of game elements for this research is provided in section 4.1.1.3.

Computer games can be broken down into game elements, which theorist Aki Järvinen (2009) categorises as *systemic elements, compound elements, and behavioural elements* (see Figure 1, derived from Järvinen, 2009).

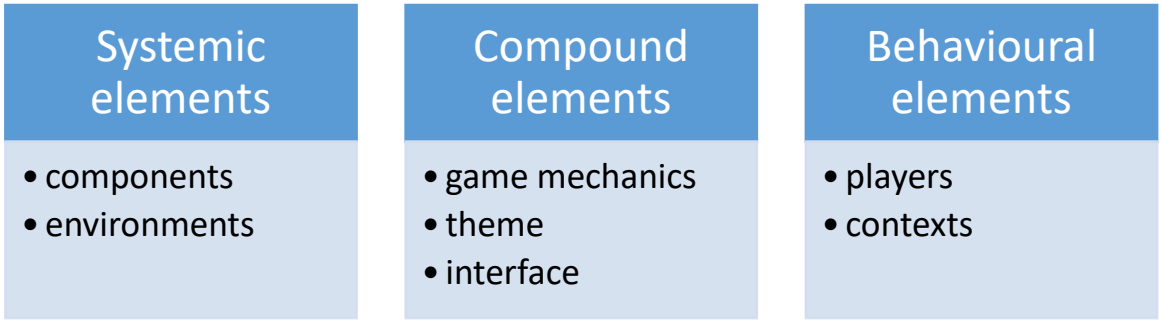


Figure 1 - Game elements (derived from Järvinen, 2009).

Components are objects that can be carried, used, manipulated or exchanged by players or the game system, for example, non-player characters. These may be weapons or tools, coins, or even larger objects such as cars or ships. Generally, these are clearly distinguished from inanimate objects such as buildings or vegetation and other parts of the environment. Components often have additional attributes such as value or power, or simply points value (Järvinen, 2009). This thesis refers to components by employing the more commonly used term *props*. The term *prop* is derived from 'theatrical property', and is widely used in the film industry (Okun & Zwerman, 2010), as well as in game design literature (for example B. L. Mitchell, 2012; Schell, 2014; Thompson, Berbank-Green & Cusworth, 2007). The game prototype used in this study makes use of a small number of props including weapons, and simple items that mostly enhance the environment such as flower pots, chairs, tables, and fences.

‘Environment’ is the physical or virtual space that constrains game play and defines some of its game mechanics (Järvinen, 2009). Within this research, the environment is the game map or game level, which includes streets, buildings, containers, and a terrain. These elements are also the main factors that are influenced by the designer in the design prototype software.

‘Game mechanics’ describes the way in which a player interacts with the game (Schell, 2014). In the case of first-person shooter games, this encompasses movement in the environment using a mouse and keyboard, as well as shooting a gun or taking damage (losing health points). Game mechanics also define the way in which players achieve a goal. In this prototype, this represents not only shooting enemies, but also defending or capturing the flag point. The game mechanics implemented in the prototype, as well as the theme and interface, will be discussed in section 4.1.1. with the latter able to be broken down into design software interface and player (in-game) interface.

The ‘theme’ is the element that defines not only the look and feel of the overall game, but also contextualises the meaning of the game itself. This may entail a specific style and convention, which can be entirely novel or in other instances, derived from popular culture genres such as science-fiction, fantasy, crime, or war (Järvinen, 2009). The theme used in this study is simply a contemporary urban environment, where two teams engage in a (simplified) daylight match to capture a flag point. The prototype intentionally makes no use of any particular style or genre in order to provide the most basic, but also perhaps most generalisable theme possible, as this study is concerned with evolutionary design systems in general, without any thematic or even domain-specific context.

‘Interface’ is the medium through which the player interacts with the game system (Järvinen, 2009). In most computer games, this is either a mouse, keyboard, or game controller, or a combination of these. While many alternative and experimental inputs are noted in the literature, a deeper discussion of any unconventional interfaces is beyond the scope of this study.

Defining players poses another dilemma. Intuitively, it seems most simple to state *what* a player is, yet many theorists have struggled to find a common definition that suits all theoretical contexts. Juul (2011) views the player as a trans-medial consumer, a notion that, for example Järvinen (2009) agrees with; others such as Fullerton (2008) and Salen and Zimmerman (2004) link the definition of a player more strictly to games, where a person only

becomes a player if they submit to the rules and constraints of games (Fullerton, 2008). This thesis will not attempt to enter this discussion and instead employs the term 'player' simply as the user of the computer playing the game. Players influence the game system through inputs based on their actions and decisions, and I am more interested in player behaviour observed in computer games than in general game studies and ludology. I will therefore apply a simpler, narrower view, and focus on the elements that drive gameplay, including mechanics, assets, and layouts.

Järvinen (2009) defines context elements as the "time and place where the game takes place" in its most basic form. Therefore, context elements are very closely related to thematic elements in games, and according to Järvinen, can lead to numerous debates around cultural backgrounds. This thesis follows his stance, and assumes a very practical game design approach, as the focus of this study is to produce new knowledge in procedural game content generation and evaluation, and not a better definition of theories of games.

2.1.2 Generative Design

Generative design is the field of study concerned with algorithmic form and shape-finding (Bohnacker, 2012). It is also often referred to as 'procedural design', highlighting its algorithmic origins using computational procedures. I have discussed the core differences between manual design supported by computer software and automated design conducted by computer software in previous work (Kruse, 2014). Therefore, a repetition of the same matter seems unnecessary here. However, a number of important aspects that address the enhancement of the designer's capability had been omitted in this earlier work; therefore, a brief look at some aspects will be conducted here. Specifically, literature concerned with computer (or video) game design was not part of the previous study, and thus requires additional attention.

Ernest Adams (2013) argues in his often-cited book, *Fundamentals of Game Design*, that the creation of a great video game necessitates solid game design. This emphasises the impact of design on the quality of computer games. Accordingly, a short discussion of the overall design process embedded in the design cycle will be given in the following section.

It also seems important to point out that this study considers solutions that explore the fine line between fully automated design and manual content generation. The core aim of this study is the investigation of software tools that support designers in their endeavour to

explore new design solutions. This research does not seek to replace designers, but to empower them. To achieve this, concluding the design cycle through content evaluation is a necessity. The following subsections will show that there is a gap in existing knowledge, in particular with respect to domain-specific content.

2.1.3 Design Cycle

There have been many attempts in the literature to define the design process and its cyclic or iterative nature (for example, Adams, 2013; Lawson, 1997). Some of these definitions take a very domain-specific approach, whereas others seek to provide generalised denotation. This study will not enter into a discussion as to whether the former or latter is a valid approach, as doing so will be far beyond the scope of this research. However, to understand the interrelation between procedural content generation and evaluation, a definition that is found in computer game literature will be given. I am aware that it is both simplified and very general, but it serves the purpose of highlighting the basic iterative nature of many design approaches, including game content generation. Furthermore, in the literature, the design cycle is often depicted as a single iterative cycle with four main elements, namely, ideation, implementation, testing, and evaluation (for example, Lawson, 1997). In other cases it is simplified as a three stage process that includes concept, elaboration, and tuning stage (Adams, 2013). The design cycle employed by this study employs two iterative elements, one for major revisions of fundamental design flaws that required re-thinking and new ideation, and another, smaller cycle for minor iterations only concerned with mild modifications and re-testing (see Figure 2). This approach follows Fullerton (2008). It should be acknowledged that the terms ‘design cycle’ and ‘design process’ are dissimilar, and predominantly used in different fields (design disciplines and engineering). Alternatively, the ‘design cycle’ may have a number of components that describe each of the three or four main stages. Most design processes have a much more detailed breakdown and therefore, a higher number of basic elements. It is also worth noting that design processes exist in linear and iterative formats, reflecting their algorithmic heritage and application, whereas design cycles (as the name suggests) are usually of a cyclic, iterative nature. With all possible caution in mind, it can be suggested that their interchangeable use is likely a discipline-specific problem. A deeper discussion of this hunch has to date, however, not led to a more defined thesis, and may be conducted in future work.

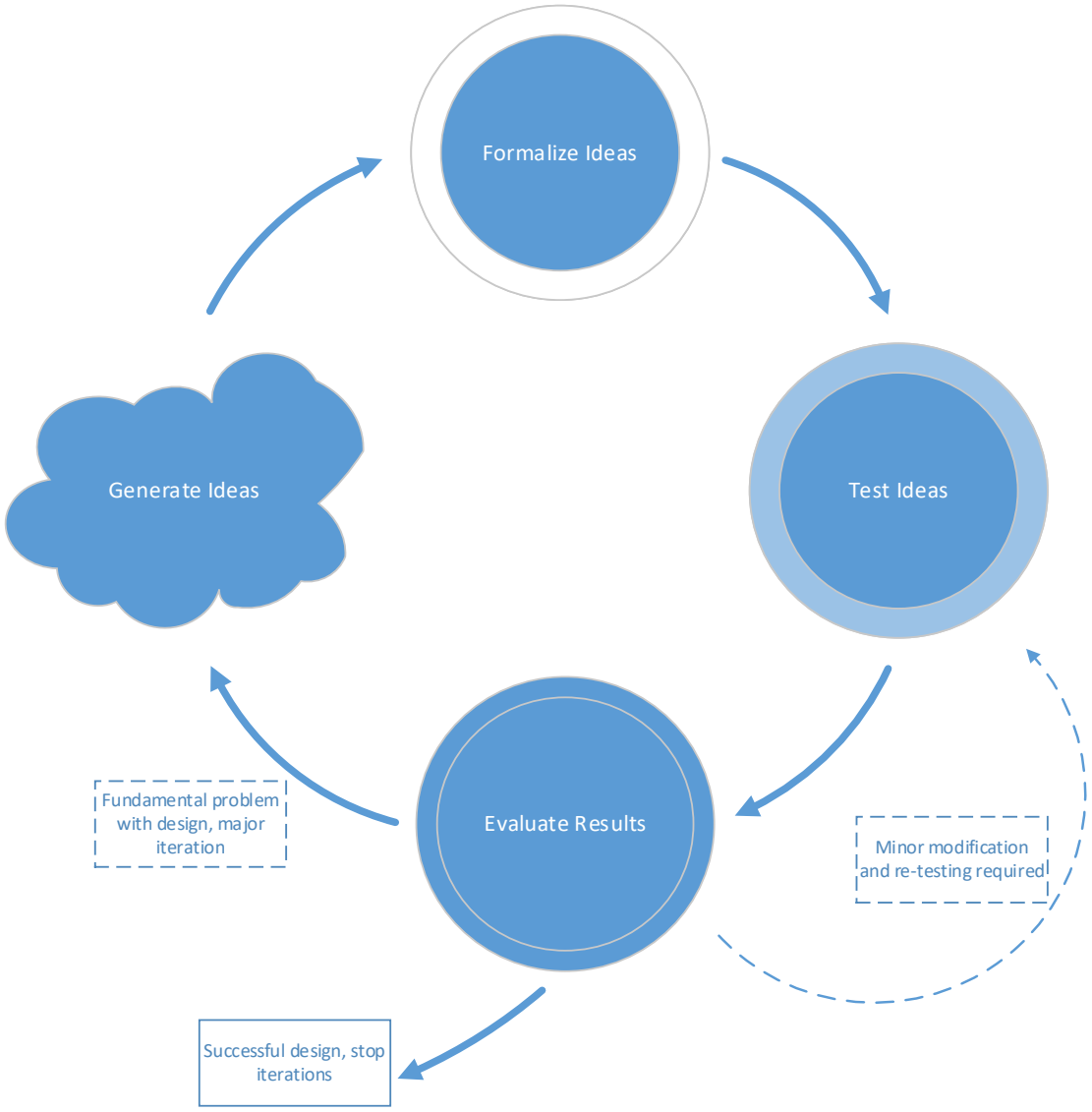


Figure 2 - Design cycle as per Fullerton (2008).

Game content generation, like many other design tasks, starts with idea generation (Fullerton, 2008) or ideation. These ideas are normally driven by the overall design goals, which define the intended player experience, including game concepts and game mechanics that may serve said player's experience goals. Fullerton (2008) uses the term 'brainstorm', whereas Macklin and Sharp (2016) refer to this as 'concept'. This indicates the open and broad approach that can be taken in order to achieve new solutions that address problems with an existing design. It is the phase in which to generate ideas and concepts as a starting point for the iterative design process that follows.

The next main step within the design process (Figure 2) is the formalisation of ideas (or systems), which is the step of creating a testable prototype that embeds all or at least some of the previously conceived ideas for addressing player experience goals (Fullerton, 2008).

Following the formalisation of ideas or systems, these ideas are tested (Figure 2). As indicated above, two prototypes are employed in this study to evaluate the quality of the multi-agent system in order to generate a response to the primary research question. Testing ideas (or play-testing) is the process of benchmarking an idea or system against the original player experience goals (Macklin & Sharp, 2016).

Resulting findings are evaluated and ranked in order to arrive at the conclusion of whether the design was successful. Fundamental flaws and significantly negative results may necessitate another iteration, which is initiated through a new ideation step, essentially closing the design cycle. Otherwise, if results indicate improvements, and no fundamental issues are unveiled – but minor modification and re-testing is needed – only a small iteration is induced, as shown in Figure 2. Should the results be positive and the design appears to be successful, the design process is terminated and no additional iterations are required.

Chapter 3 will show that in the current study, there are actually two different prototypes, which both follow the same iterative design process, but target vastly different user groups, that is, designers and players. At the core of this study is the designer, who engages with the design prototype. In this particular case, there are no player experience goals, but rather, designer experience goals: ideas that can potentially lead to a more effective or more enjoyable design experience. The second prototype follows the notion of player experience and resembles a simplified first-person shooter game, with essential elements such as weapons, the game level layout itself, damage models, sound, and other aspects. A detailed discussion of both prototypes is provided in the following chapter.

2.1.4 Computational Creativity

Liapis, Yannakakis and Togelius (2014) argue in their position paper (which coins the term ‘computational game creativity’), that an extensive exploration of computational creativity has led to a significant number of autonomous generative systems covering several disciplines in the arts, entertainment, maths, and engineering. They highlight the potential of automation in game design, not only to advance game research itself, but also as a path leading to innovation in computational creativity. The paper depicts games as the “killer

application for the study of Computational Creativity" (Liapis et al., 2014). The current thesis follows this bold statement, as its veracity is argued in a strong and convincing manner. Unlike the majority of many current advancements and achievements in computational creativity, which are concerned with only a single particular aspect of creativity, such as music, sound, images, visual style, narrative, or interaction, computer games are highly interdisciplinary, multifaceted, and diverse. A number of different domains are employed within a single output, a feature that allows for study of the topic, while enforcing consistency between its contributing creative domains. For example, a game – similar to a movie or a theatre play – necessitates consistency between visual language, narrative, and sound in order to allow the audience to reach a certain level of immersion (Adams, 2013) , or what is known as *suspension of disbelief* (Brown, 2012). Yet unlike film and theatre, video games add an additional important facet: interaction.

In film, missing interaction highlights the lack of audience agency. Agency is understood as the audience being an integral part of the state of the game, that is, being able to manipulate and change the direction the game takes. In essence, it means to be the player of a game (Brown, 2012).

The absence of interaction in theatre underlines another important role that an audience plays when playing computer games, which is authorship (Brown, 2012). The narrative of the game is co-created by the inputs of the player. In particular, in puzzle and adventure games, authorship of the player is an integral aspect that shapes the course of the story.

Liapis et al. (2014) also point to an important distinction of different forms of computational game creativity that needs to be mentioned in context of the present research. They identify several gaps in the literature, for example, fully automatically generated games. The current research does not attempt to pursue this avenue; rather, another gap in knowledge defines the foundation for this study, which is the evaluation of generated game content. This study also embraces the human designer as part of the process, and views content evaluators as a useful tool, not a threat to human design work.

In conclusion, it can be stated that computer games appear to be extremely suitable for computational creativity research, which is one of the reasons that computer games have been selected as the field of study for this thesis. Computational game creativity also

identifies the main research concern of this study, namely procedural content evaluation, which is the subject of discussion in the following section.

2.2 Creating computer game content

Thus far, a short overview of what constitutes a computer game, in general terms, has been given. This leads to the creation of computer game content, and what this means in the context of the present research. This section will take a brief look at traditional approaches, procedural content generation, and evaluation.

2.2.1 Traditional Approaches

Game content development is a multi-faceted and highly diverse undertaking that is constantly developing, and it is as difficult to define as it is to state precisely what a game is (Freyermuth, 2015). Game content includes game levels and maps, rules, assets, props, characters, and music (Togelius, Shaker, & Nelson, 2016), and a broad overview of these is far beyond the scope of this literature review. Accordingly, in this review, I will focus on game levels, and more specifically, on FPS maps and the processes that are required for creating this type of game level.

An important consideration that needs to take place before the game level creation process begins is the visual language that is to be created through models, layout, and any other visual elements in the game environment (Schell, 2014). The visual language of the game expresses environment setting, location, and theme. Examples of setting include urban, forest, and space. Location may be New York City, (an undefined) cabin, or the Eiffel Tower in Paris. Possible themes are abandoned, steam-punk, post-apocalyptic, or paranormal, among others (Galuzin, 2016). In addition to a successful visual design, which includes a consistent visual language, there appears to be no need to use narrative or other means to communicate setting or theme to the player. This highlights the importance of careful visual language design.

Game level creation generally starts with the selection of essential building blocks such as terrain and large objects and landmarks that define the theme and scale of the level (Galuzin, 2016). These assets can either be manually created using digital content creation software such as Autodesk Maya (Murdock, 2017) and Blender (Fisher, 2014), or they may be procedurally generated, as is discussed in section 2.2.2. Consideration must be given to

themes that are created through shape, textures, colour, and the composition of individual three-dimensional models, but also the aforementioned collective thematic appearance (or visual language, sometimes called 'the look' in popular media) of the asset pack, as it defines aspects of the game such as period, cultural association, and visual style (Galuzin, 2016).

Once the basic building blocks have been acquired through manual or computational means, they need to be placed in the three-dimensional environment of the game. A number of considerations need to be taken into account, including the shapes and silhouettes that placement creates (Schatz, 2017).

An important aspect to note is that each step, from model creation to level layout has a major impact on game mechanics and therefore, on playability and enjoyment of the game. Additionally, the entire process is extremely time consuming and increasingly costly, as mentioned in section 1.1. One possible solution to mitigate the risk of game production is the use of computational tools, not only to manually create assets and place them, but also to automate a significant proportion of the entire undertaking. This process is called procedural content generation (Shaker et al., 2016b; Short & Adams, 2017; Togelius, Kastbjerg, Schedl, & Yannakakis, 2011) and the following section discusses some of the literature in this field.

2.2.2 Procedural Content Generation

Procedural content generation in computer games can be defined as the automatic creation of game content using algorithms (Togelius et al., 2011). Other definitions are more liberal and define procedural content generation as game content creation with no user input, or limited user input (Shaker et al., 2016b). This research uses the latter definition and understands PCG as game content creation with limited user interaction. Therefore, some of the content within the design phase of this research was created by computational measures only; however, a small proportion of user input is always required.

To understand this definition of procedural content generation in greater detail, it can be broken down into smaller components. In computer games, *content* can refer to many different possible elements that are part of a computer game (Macklin & Sharp, 2016). Some of the *components* identified above may be included in these elements. For example, levels (or 'maps'), sound and music, narratives, and most importantly, items such as buildings, cars, vegetation, and props (smaller items like tools or weapons) are all part of the content of the

game. Accordingly, some of it may be procedurally generated, or in other terms, computationally created, with little or no user interference.

The terms *procedural generation* and *generative content* indicate the computational, algorithmic nature of an artefact. However, Short and Adams (2017) remind us that generative content differs from procedurally generated content, as the latter is bound by gameplay constraints, whereas generative content (or art) is free from these restrictions. The current research focuses specifically on procedural game content and takes gameplay into strict consideration.

Procedural content generation can take a number of different forms including search-based (Togelius & Shaker, 2016), agent-based (Shaker, Togelius, & Nelson, 2016a), grammars and L-systems (Nagle, Wolf, & Riener, 2016; Togelius, Shaker, & Dormans, 2016), and hybrid approaches to generate multiple facets of games (Cook & Colton, 2011; Cook, Colton, & Gow, 2017). There is significant praise for procedural content generation, and a number of game designers have applied this technique in successful games. Examples include *Rogue* (G. Smith, Gan, Othenin-Girard, & Whitehead, 2011; Toy, Wichman, Arnold, & Lane, 1980), a game that virtually created an entire genre, *Dwarf Fortress* and *Diablo* (Hendrikx, Meijer, Van Der Velden, & Iosup, 2013). However, there are important considerations that have to be taken into account when procedural methods are used to generate game content. First, quality assurance may be an issue, due to the incoherent content that testers will be faced with, that is, reviewing every possible output of a content generator is nearly impossible, and not economical (Short & Adams, 2017). This highlights that procedural content generation cannot simply stand on its own, and that content evaluators are also required. As this issue is at the heart of this thesis, we will take a deeper look at said evaluators in section 2.2.3. Second, while content generators are often assumed to provide efficiency and time-saving, this is not guaranteed, given the complexity of some generators, which consume significant software development time (Short & Adams, 2017). Furthermore, Short and Adams (2017) argue that designers often wish to create a highly authored experience for the player. In my opinion, this implies that there is a need to shift current attempts from a developer-centric approach to a design-centric method. This presents a gap where procedural content generation can be inclusive of a designer, and augment their abilities, rather than replace them. This contrasts

computational game creativity (Liapis et al., 2014), which aims to build systems that generate games by themselves.

The two most encompassing contemporary pieces of literature that have undergone a rigorous peer-review process are likely the books *Procedural Content Generation in Games* (Shaker et al., 2016b) and *Procedural Generation in Game Design* (Short & Adams, 2017). I am aware that there are numerous publications on this topic related to conferences and in journals; however, these two books represent extremely useful and complete collections of current issues and provide a broad overview of the topic. For example, Short and Adams (2017) make a case for well-adjusted procedural content that strikes a balance between repetition, which causes boredom on one hand, and chaos as a result of randomness without designed structure on the other. They argue that procedural generation may be mistaken for being a replacement for content *design* (Short & Adams, 2017), when instead it should be considered a replacement for content *creation* (Short & Adams, 2017). They continue to state that unexpected results do not necessarily contest "pleasantly surprising results of high quality" (Short & Adams, 2017). The emphasis to aim for content design rather than simply content creation reflects the primary motivation behind the present study. Design as a process of creation and evaluation necessitates a measure for quality, thereby enabling content assessment (see section 2.1.3). Furthermore, while computational creativity also seeks to create and evaluate, this study seeks to take a designer-centric approach, one that is much more closely related to how procedural content generation literature, such as the two aforementioned books, treat the link between game designer and player (see also section 2.3.3). This is amplified by the idea that computational creativity seeks to find a multi-faceted solution to automated content (Liapis et al., 2014), whereas procedural content often simply addresses one aspect of a game (Short & Adams, 2017). The present study focusses on game level design and ignores other facets such as, for example, sound and character design. It is therefore grounded in procedural content generation, and not in computational creativity.

Procedural content generation literature often assumes common programming paradigms such as functional and object-oriented programming (for example Bohnacker, 2012; Greenberg, Xu & Kumar, 2013; Pearson, 2011). The prototype for this study is written in object-oriented Java, and borrows only high-level concepts such as messaging between agents from the agent-oriented programming paradigm, as introduced by Yoav Shoham

(1993). Section 2.5.3 discusses the details of this approach and introduces human-based genetic algorithms. Section 3.5 illustrates that the core of the design prototype follows a traditional object-oriented programming model, whereas the resulting levels use mechanics of the Unity 3D game engine, a simple, event-based system (Okita, 2015). Furthermore, Shoham (1993) points out that there are significant overlaps in object-oriented and agent-oriented programming. In order to keep this argument clear, and to focus simply on the essential elements of this study, a deeper examination of agent-oriented programming seems unnecessary here. This study also follows the assumption that an object-oriented paradigm is being employed for the study's procedural content generator.

In summary, a number of elements in a computer game may be created using computational means, and the current research study employs procedural content generation as part of the design process, in particular, its level design.

2.2.3 Procedural Content Evaluation

The discussion of the design cycle for computer games (Figure 2) highlighted not only the need for content creation, but also its evaluation in order to establish whether additional iterations are required, and if so, whether a substantial re-design (major iteration, new ideation), or a simple 'tweak' (minor iteration and subsequent re-evaluation) is needed.

Procedural content evaluation is the field of research that is concerned with the quality and assessment of procedurally generated content. It seeks to quantify how well a particular game element fulfils its purpose (Shaker, Smith, & Yannakakis, 2016). In this context, a game element does not have to be an asset, such as a three-dimensional model, an image (sprite), or a sound. It can also be a game mechanic, a narrative, or a game level (layout). Creating procedural content is fundamentally simple. Even random methods seem to have a certain appeal (Connor, Greig & Kruse, 2018), and more sophisticated methods provide complex gameplay experiences. However, assessing whether a particular asset, mechanic, or full game is of good quality or not, is still subject to investigation.

Cook et al. (2017) created a computational creative program called *ANGELINA* which seeks to generate games that are then assessed by players for instance through online comments. While it considers a designer perspective by applying general design principles, it does not model designer feedback into system. It strength seems to be the ability to generate content and full games based on simple and sparse themes.

A number of researchers are actively working on solutions that allow for a computational evaluation of content. Investigations have been conducted into assessing 2D rogue-like games (Liapis, Yannakakis, & Togelius, 2013b), aesthetics for aiding computational creativity (Galanter, 2012), and assessing platformers by using *Super Mario* as a case study (Summerville, Mariño, Snodgrass, Ontañón, & Lelis, 2017) to name a few. Other works have looked at creating and benchmarking 3D FPS games (Giacomello et al., 2018) or creating a framework for generation and evaluation based on player perspectives (Liapis, Yannakakis, & Togelius, 2013a). I believe that these attempts are important stepping stones toward computational domain expertise that is able to analyse complex game content, such as FPS levels. However, there remains significant work to be done in this area.

A number of researchers argue in favour of generic methods for evaluating content such as game levels (Liapis et al., 2013b). However, experts in game level design have highly domain-specific skills, and a major part of their ability to assess the quality of a level and give a founded statement about playability or other metrics about a game level, is based on this domain specificity. While it is an interesting goal to create computational solutions that are domain independent, I believe that a capable domain-specific solution should be a first logical step as part of this study.

2.3 Cognitive Modelling

This thesis is concerned with how game level designers think, and the processes they utilise to create interesting, playable, and immersive game levels. This thesis considers game level design within a player-centric approach, assuming that designers do not make games only for themselves, but also for others. The assumption is that game level designers consider what players may enjoy, and how they approach all the different areas and elements of a game level. Therefore, section 2.3.1 discusses the literature focused on player experience goals, playability, and the heuristics that designers use to translate the abstract concept of playability into pragmatic, applicable design choices.

2.3.1 Player Experience Goals, Playability Heuristics, and Playability

The literature discusses a number of terms that describe very similar ideas. First, there is ‘player experience’, which can be viewed as the user experience when playing games (Sánchez, Simarro, Zea, & Vela, 2009). Player experience is created by subsequent ‘player experience goals’ (Fullerton, 2008). Other researchers conclude playability as a combination

of both gameplay and the GUI (Paavilainen, Korhonen, & Saarenpää, 2012). Zhu et al. (2017) highlight that these definitions do not provide a clear distinction between 'playability' and 'usability'; rather, they attempt at creating an encompassing review of these two categories within games, with the aim of ultimately developing heuristics that are able to aid game designers in defining clear goals for their game development. The following sections aim to provide an overview of these elements without entering into a discussion about providing a clear definition. Following a pragmatic approach that allows the researcher to conduct this study without the need to provide a bullet-proof definition of playability (or player experience goals or playability heuristics), this research emphasises gameplay, including a sophisticated GUI, and not the overall user experience. My interest lies in developing gameplay elements through level design. However, for the purpose of completeness and in order to ensure that the development of game levels, based on player experience goals, is grounded in existing literature, brief consideration will be given to player experience goals and playability here.

2.3.2 Player Experience Goals

This study uses a player-centric game design approach. This is not only reflected in my own thinking in terms of constructing the prototype for the designer, but more importantly, in the resulting game levels that the prototype tool was supposed to design.

Player experience goals are paramount to player-centric design, as they set goals that foster the experience the designer wishes the player to have (Fullerton, 2008).

Defining player experience goals may seem straightforward, but some researchers argue that they can easily be confused with goals resulting from game mechanics (Järvinen, 2007). Järvinen (2007) argues that game mechanics can be split into three categories, *game-defining mechanics*, which often characterise a specific game (e.g. the now proverbial 'jump' and 'run' mechanic in *Super Mario*), *submechanics*, which have supporting functions such as manoeuvring to a specific location to be able to perform a primary mechanic, and finally, *modifier game mechanics*, which may only be temporary or locally relevant. Taking these distinctions into account, the main global player goals may not always line up with every single mechanic, particularly where *submechanics* or *modifier game mechanics* are concerned (Järvinen, 2007). For example, a point system may drive the main player experience goal, whereas going to a specific location within a game level may not directly result in the accumulation of points, but subsequently contribute to the player experience goal, if said

location offers a better position for the player from which to gain points through kills, or to obtain a boost that allows the player to perform the primary task faster. This highlights the need to analyse submechanics in some cases, instead of relying only on player experience goals, given that, for example, in game level design, the location (for a submechanic) may offer the player a significant advantage for achieving the overarching goals, which means that players will exploit these submechanics. Therefore, in the design phase, consideration also needs to be given to these submechanics.

2.3.3 Playability Heuristics

Nacke et al. (2009) distinguish between the playability of a game as a measure for evaluating the game design itself, and player experience as metrics, which can lead to improved gaming. Figure 3 illustrates how they see the relationship between player, game and design. They argue that player experience should only be evaluated if a game has good general playability, as any underlying game design issues need to be removed in order to provide a sufficient platform for an individual game experience assessment. Their suggested methods include expert reviews and heuristics (Nacke et al., 2009). The current research aims to employ both reviews by expert designers and heuristics that are derived from secondary data, namely expert designer accounts of map level design. 'Expert designers', at least in this study, are considered designers who have extensive domain knowledge and potentially even some FPS game play expertise, which helps them to identify fundamental problems within the game design that can potentially hinder general playability. These heuristics are discussed in section 4.1.5.1.

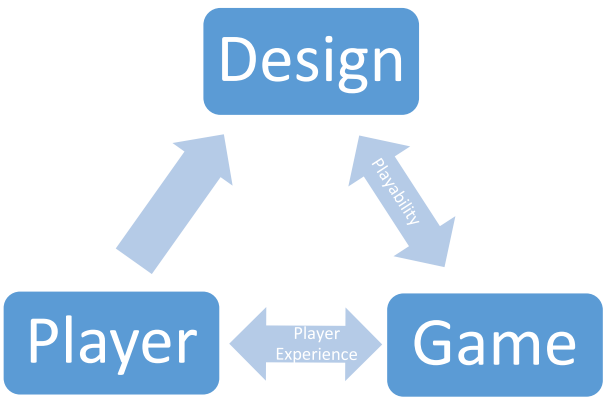


Figure 3 – Game Design cycle, as per Nacke et al. (2009).

Nacke et al. (2009), as well as Zhu et al. (2016), both point to a gap in the literature and suggest that research into playability heuristics remains ongoing, and needs further investigation. They emphasise that there have only been a few attempts to develop playability heuristics (Nacke et al., 2009). There are, however, some examples in the literature, which offer a few interesting pointers.

Desurvire, Caplan and Toth (2004), for example, researched playability heuristics developed from the literature that is specific to computer and board game evaluation, and found that, firstly, playability heuristics can be useful for improving game design. They tested their results in an evolving game design, which highlights the relevance of their study to this thesis. The group also found user testing to be a necessity for validating playability heuristics used to evolve games, as a designer will not be able to fully predict player behaviour (Desurvire et al., 2004).

Others, such as Pinelle, Wong and Stach (2008), derived their heuristics from online game reviews. They confirm that experienced designers will be able to predict a number of playability issues based on their experience; however, many designers will nonetheless benefit from support through formal heuristic inspection and evaluation to identify issues early on (Pinelle et al., 2008). This thesis implements (playability) heuristics as part of the design cycle, and offers the support identified as important by Pinelle et al. (2008) to designers, by making it an inherent feature of the overall level generation process.

Another similar approach to heuristics development was fairly recently published by Zhu, Zhao, Fang and Moser (2017). This group of researchers also considered a number of game reviews, but rather than utilising large parts of the review texts and analysing them using qualitative coding methods, effectively categorising the problems with the game as identified by the online reviewer, Zhu et al. (2017) used a methodology called the ‘lexical approach’, which derives personal traits from the use of adjectives. The work in their 2017 paper expands on previous research by also considering nouns in order to create a more complete model of the personality traits reflected in game reviews. They found that using adjectives alone will lead to an insufficient reflection of subjects and context in the resulting playability heuristics; however, by adding nouns to their analysis, a number of heuristics that serve as useful design guidelines have been developed. The authors make an interesting remark in their final

conclusion, when pointing to the limitations of their own study, where they suggest that enhanced playability requires game designers to find a good balance between novelty and familiarity. They suggest that future research should gather data from experienced designers on how to achieve this balance, in order to deliver a good player experience (Zhu et al., 2017). Addressing this gap in the literature, by deriving heuristics from experienced designers, is one of the primary contributions of this thesis.

2.3.4 Applications of Cognitive Models

Cognitive models form an important part of current research into computational generation and evaluation of several different aspects of computer games, for example, player experience models for procedural content generation (Yannakakis & Togelius, 2011), adaptive games based on player models (Charles et al., 2005), custom game mechanics reflecting player personality traits (Nagle et al., 2016), and personalised elements and mechanics with a focus on user-centric design (Ferro, Walz & Greuter, 2013). I wish to avoid a wider discussion of cognitive science (e.g. Boden, 2006), however, given that my area of interest is reasonably narrow. I am keen to test whether it is possible to model specific traits of a decision-making process in the form of heuristics, as suggested by Zhu et al. (2017), in order to find a response to my research question. Therefore, the following discussion of the literature will only touch on a few aspects of cognitive modelling and is in no way meant to represent a full taxonomy of the field.

Finding a definition of 'cognitive models' may seem trivial; however, there is no consistent definition to be found within the literature. A number of different terms have been proposed, depending on the domain in which these models are applied. Examples include player experience models (Shaker, Shaker, Abu-Abdallah, Al-Zengi & Sarhan, 2013), player behaviour models (Bakkes, Spronck & van Lankveld, 2012), as well as more generic player models (Charles et al., 2005; Togelius, Shaker & Yannakakis, 2013; Yannakakis, Spronck, Loiacono & André, 2013). There is also research focusing on models that are not strictly confined to players or their behaviour or experience. This includes the use of the terms 'mental model' (Seidel, Berente, Lindberg, Lyytinen & Nickerson, 2018) and 'cognitive model' (Bohil & Biocca, 2007; Fum, Missier & Stocco, 2007) as more generic definitions and applications of human models in computer game contexts. Given that the focus of this study is indeed not on players

but on designers, I will simply refer to ‘designer models’ or ‘cognitive models’, where appropriate.

While player models are not a focus of this thesis, they still deserve a short overview here. Additionally, I will briefly provide the reasons why I believe they are important in the wider context of this study. First, in the initial phase of this study, a number of different avenues were explored, and one focus area was player models. It was later deemed to be more important to emphasise research focusing on thought processes and the practice of designers, a motivation that has recently been confirmed by a study pointing to a significant gap in the area of player experience research (Zhu et al., 2017). Player models, however, are still somewhat relevant to this thesis, as designers in the literature, and those with whom interviews were conducted as part of this thesis, clearly emphasised a player-centric design approach, which in turn is based on several factors, among them what designers believe is important to players, and what designers think players will do and prefer in certain situations within a game. This was particularly highlighted as an important design aspect for multiplayer games (Ølsted, Ma & Risi, 2015), which have limited narratives and often reasonably simple gameplay. These games are predominantly driven by players, which gives careful consideration of player behaviour even higher priority than in, for example, platformer or puzzle games. Finally, player models are deemed to be important to this thesis, as they may offer possible ways forward in game level and game asset evaluation, which can potentially contribute to advancements in computational game creativity.

A recent position paper suggests employing active learning to model player behaviour, that is, using unsupervised methods, but with limited labels, in order to achieve an efficient and potentially more accurate model than supervised methods with a limited, known set of labelled instances (Togelius et al., 2013). This may be a promising area of future research, especially in the context of large self-learning neural networks, which have come into focus in the past few years due to the accessibility of vast computing resources in the form of general purpose GPU computing.

Finally, cognitive modelling in the context of computer games can also be approached from the perspective of designers rather than players, and the following section discusses literature in this particular area.

2.3.5 Cognitive Designer Model

One of the main challenges of this study is the construction of cognitive models for the design process. From a general perspective, a cognitive model is a system (computational/artificial or natural) that simplifies complex processes and features by forming an abstraction of such processes, and reducing non-essential features (Fum et al., 2007). Dawson (2003, p. 6) refers to a (cognitive) model as "an artefact that can be mapped onto a phenomenon", to help us gain a better understanding of the (human/behavioural) phenomenon. According to Dawson, the model needs to be "useful" (Dawson, 2003, p.6), which implies that a model allows us to work easier, or to understand a modelled phenomenon more easily than simply considering the phenomenon on its own (without a model). This does not only apply to cognitive models in particular (Boden, 1977; Feigenbaum & Feldman, 1963), but to the process of modelling human traits in general; for example, in mathematical and statistical models of human decision-making or group behaviour (Pentland & Liu, 1999; Sadilek, 2012; Ziebart, Maas, Bagnell & Dey, 2009).

Dawson also highlights the difficulty of defining models due to their diversity, and interestingly, compares this problem to the issue of finding a commonly accepted definition of 'games' (Dawson, 2003). He highlights three common advantages and disadvantages of modelling that are relevant to this study. These include the precision of terms, formalisation, and communication. The precision of terms requires the careful consideration of definitions, considering that, for example, different researchers may use the same definition in different ways. This is important, as the modelling process may draw information from a number of diverse sources, where different individuals may use the same term in various ways. My models will have to ensure consistency, even when data sources are derived from different fields and a variety of people.

Dawson (2003) further argues that "one potential problem with formalisation is that the process requires the researcher to make design decisions". This implies that these decisions have an impact on the behaviour of a model. Dawson continues to describe a simple machine learner that had been successful in someone else's experiment, but turned out to converge into local minima when applied to his particular classification problems. A possible way to mitigate this risk is to test it against a benchmark, so that potential formalisation issues can be identified. In my study, I will have to construct a way to obtain baseline data in order to

pitch the cognitive model against it. I am also conscious that a complex solution is not necessarily the best possible solution. This thesis follows the mantra that simplicity should not be dismissed simply because it 'feels' simple, but should be tested first, before turning to more complex solutions. For example, instead of going straight to the computationally expensive training of a deep neural network, which also requires significant training data, and which may be difficult to obtain, simpler solutions such as expert systems should be considered first.

The class of models that this research is concerned with are computational cognitive models, which are models implemented in software, for example, as an agent. While some of the definitions for 'cognitive model' also applies to statistical and mathematical models, the main difference is that cognitive models reproduce the behaviour of the subject that is modelled, whereas mathematical models simply describe the behaviour (Fum et al., 2007). This makes cognitive models directly applicable as agents in a multi-agent system. The way in which the cognitive model is implemented is based on expert systems. The following section gives an overview of relevant literature on expert systems, and implementation is described in section 3.3.3.

2.4 Artificial Intelligence

This section of the literature review provides a brief look at the current understanding of intelligence, and moreover, artificial intelligence. A brief consideration of agents is presented; the current research project is positioned in this wide field of study to set the context for the algorithms and paradigms explored in the procedural design software prototype.

2.4.1 Machine Learning

Prior to providing a definition of 'artificial intelligence' that is suitable for this research, a brief look at the difference between adaptive and statistical approaches seems appropriate. This will help contextualise this study in the field of artificial intelligence and also draw a clear distinction between artificial intelligence and machine learning.

The previous sections have shown that artificial intelligence is located within the wider field of general intelligence. Figure 4 shows that it constitutes a number of different subfields such as evolutionary computation, agents, and other areas. This study can be positioned as a hybrid between traditional artificial intelligence approaches and optimisation methods. It draws

predominantly from the two highlighted areas in the figure below, agents and genetic algorithms (Figure 4). The details of how they interact within the procedural design prototype are discussed in section 4.

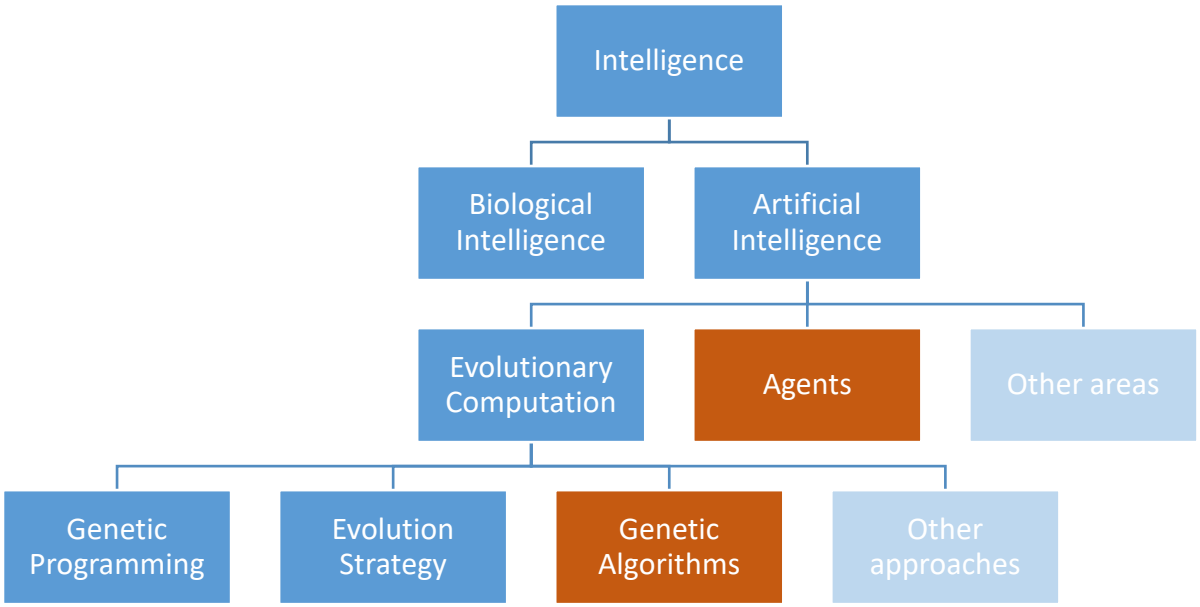


Figure 4 - Positioning agents and genetic algorithms within the wider field of artificial intelligence.

2.4.2 Definitions of Artificial Intelligence

The majority of literature places artificial intelligence as a subfield of general intelligence, and indicates it as attempting to simulate intelligent behaviour in the same way humans do. This notion of human equivalent behaviour can be traced back to Alan Turing, who proposed his famous benchmark test (the 'Turing Test'), in which a player interrogates a computer and a human in a blind test and attempts to establish which responses are human, and which ones are made by a machine (Turing, 1950). While the test has been widely criticised in the literature, it has also become an integral part of artificial intelligence philosophy (Russell & Norvig, 1994; Saygin, Cicekli & Akman, 2000).

Contrasting the idea of the imitation and simulation of human behaviour, David Fogel (2006) argues in his book, *Evolutionary Computation: Toward a New Philosophy of Machine*

Intelligence, which details evolutionary computation, that the term 'artificial intelligence' may itself be unfortunate and misleading. He suggests that any form of intelligence, whether it is expressed by living beings, systems, or machines, should be treated equally. That is, any form of intelligence should be understood, explained, and applied in the same way. Fogel claims that if all processes aimed at generating or applying intelligence follow the same principles, these processes become fundamentally similar, and follow the same physics (Fogel, 2006).

Consequently, Fogel seeks to define intelligence independent of human behaviour. Instead of simulating, or in Turing's words, 'imitating human behaviour' (which he refers to as 'the imitation game'; see Turing, 1950), Fogel defines intelligence as "the capability of a system to adapt its behaviour to meet its goals in a range of environments" (Fogel, 2006). In the researcher's view, this appears to be a very limited definition when compared to what other philosophers of artificial intelligence propose.

Russell and Norvig (1994) advocate a separation between systems that simply imitate human thinking processes, and systems that model human behaviour and consequently, act similar to humans. The former is predominantly concerned with human thinking and behaviour, and is located in the field of cognitive science, an interdisciplinary field of study that draws on methods from computer models, experimental psychology, and neuroscience (Russell & Norvig, 2003). Its main goal is to gain an understanding of how the human mind works, and subsequently, imitate its thinking processes. The second type of system is less concerned with the actual thinking processes of human minds, and more focussed on constructing a mathematical or computational model of human behaviour. A well-known example of such systems is the aforementioned Turing test (Turing, 1950). The main aim is to expel behaviour similar to that of humans, rather than attempting to exactly imitate what the human mind does. Turing suggests that this is the only way to deceive the interrogating player and proposed that human behaviour should also be reflected in the unclear responses and mistakes that the machine might make.

A recent and most prominent example for a machine that models human behaviour is IBM's 'Watson', a supercomputer that combines several methods to successfully compete in *Jeopardy*[2] games, where humans have to respond to statements (answers) in form of a

[2] TV trivia programme

related question (Ferrucci, Levas, Bagchi, Gondek & Mueller, 2013). Watson will not be able to capture all necessary existing knowledge in a database, due to computing and storage limitations. It will be a virtually impossible task to do so, and response times will be unacceptably slow using contemporary hardware. Instead, Watson uses a number of approaches to generate and apply its 'knowledge'. One of Watson's core modules is based on natural language processing (NLP), which allows the machine to understand and contextualise the answers in order to find the relevant question (Lally et al., 2012). Furthermore, Watson models several human traits to learn (reinforcement learning) while it plays *Jeopardy*, in order to understand existing knowledge that it acquired in previous games, and combines this knowledge with new, previously unknown contexts (Prager, Brown & Chu-Carroll, 2012).

2.4.3 Agents

Agents are at the core of this study, and their application with the aim of augmenting a human designer represents one of the main contributions of this thesis. Therefore, a closer look at the core properties of agents relevant to this study will be presented in the following sections. First, however, some clarification for what 'agent' means in the context of this research is required.

Agents, and more specifically, autonomous agents, have been the subject of many research projects, with a number of definitions being proposed. It is paramount to establish the difference between 'software' in general and 'agents' as a particular type of software. In their taxonomy of autonomous agents, Franklin and Graesser (1996), examine ten different definitions of autonomous agents, and attempt to answer the question of what defines an agent, and what is merely a (software) program.

Probably the most common and broadly used definition in the literature derives from Russell and Norvig's (1994) textbook, *Artificial intelligence: a modern approach*, which simply states that an agent needs to be able to independently examine and act within its environment. Other researchers add attributes such as independence and persistence, which ultimately contribute to the notion of agents being autonomous and acting without human intervention (Smith, Cypher & Spohrer, 1994; Wooldridge & Jennings, 1995). This summarises the particular perspective this study takes, as agents are considered as independently acting within the space of possible solutions to a game level design. This study does not follow

narrow definitions such as the requirement of agents to be embodied or situated within a real and physical environment (Brustoloni, 1991; Franklin, 1997).

Three different computational agents are used in this study, which work in conjunction with the human designer in a multi-agent system. The first agent provides analysis and statistics of game levels to the multi-agent ecosystem. It uses a number of techniques to collect metrics about the population and its members. These metrics are relayed to other agents via a simple message system. The analysis is independently performed and the resulting information is used by other agents in different ways, which led to the decision of implementing it as an agent in the current thesis. The second agent is based on a cognitive model of game level designers and implemented as an expert system. In some ways, it acts similar to a fitness function by ranking levels according to their prospective playability and enjoyability, using the aforementioned designer model. Finally, a diversity agent is added to the selection process from the breeding pool, by selecting two game levels that have properties contrasting the levels with the highest fitness. Literature relevant to these different agent implementations is discussed in the following subsections. Design decisions and details of agent implementations are discussed in chapter 4.

2.4.3.1 Raycasting using Digital Differential Analysis

In the context of the multi-agent system, one of the design decisions was to implement the level design prototype outside of a common game engine. The reasoning behind this is discussed in chapter 4 of this thesis, but it seems important to nonetheless point out here that the decision to implement the majority of this work outside common tools was simply driven by a curiosity to understand the inner workings of some frequently used algorithms, and to gain a level of abstraction in the hope of an easier generalisation of the findings. Rather than binding most of the logic to a particular game development environment with its respective tools and constraints, I wanted to keep everything as accessible and flexible as possible. This led to a Java implementation of the Digital Differential Analysis (DDA) algorithm, which is used to provide a form of 'sight' and 'hearing' to the level analysis agent in this prototype.

Digital differential analysis was developed in the mid-20th century, at the dawn of computing (Sprague, 1952), and has seen a number of derivates and iterations since. It was adapted to digital logic from its original analogue conception (Elshoff & Hulina, 1970), and is a common

computer graphics algorithm employed to find intersections between voxels (in three dimensions) or cells (in two dimensional applications), and a ray that has been cast from defined points within the coordinate system. Ken Museth, who is considered the father of VDB[3], which is a highly efficient volumetric data structure used in computer graphics, (Museth, 2013), and who also received a Technical Academy Award for this work, has successfully adapted digital differential analysis to traverse voxel spaces in OpenVDB files (Museth, 2014). The algorithm is closely related to Bresenham's line algorithm (Bresenham, 1965), which was developed for efficient line approximation in grid structures (rasterisation). This thesis uses a simplified variant of digital differential analysis to traverse grid cells from one point to another, and to detect any intersections with objects in its path. Digital differential analysis offers the efficiency of conducting analyses in complex spatial environments such as computer game levels, and using a number of obstacles per level, as well as several levels, as part of the population of the genetic algorithm applied in this study.

2.4.4 Expert Systems

Expert systems have been a subject of research for nearly five decades, first merely as theoretical exercises starting in the 1950s, and later as actual implementations in various fields, when computers became available to researchers. Edward Feigenbaum is considered one of the fathers of expert systems, who not only contributed significant early work in the field in the 1980s and 1990s, but was also acknowledged for his contributions by having been awarded the ACM Turing Award in 1994, one of the highest recognitions in computer science (a discipline that is not recognised with a Nobel Prize category). Feigenbaum delivers a definition of expert systems that is still widely accepted in the literature:

> An expert system is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners in the field. (Feigenbaum, 1980, p. 2)

[3] VDB is a voxel space data structure

This study follows a slightly more concise and updated variant of the above definition, as suggested by Peter Jackson (1998) in his often cited introductory textbook to expert systems. *Introduction to Expert Systems* constitutes an expert system as "a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice" (Jackson, 1998, p. 2). Breaking this concept down, Jackson argues that expert systems have to possess knowledge, as opposed to simply employing an algorithm to process information. Instead of simply hosting a number of questions and answers that are merely processed by the computer program, similar to a database query, an expert system simulates human behaviour by "reasoning over representations of human knowledge" (Jackson, 1998, p. 3). This definition links directly to what is discussed in section 2.3.5 of this thesis regarding cognitive models, and the requirement that models reproduce the behaviour of the modelled subject. Interestingly, Feigenbaum used the same notion of simulating behaviour and the actual term 'model' in his original definition, as shown above.

Expert systems also have to be domain-specific (Duda & Shortliffe, 1983). Information has to be relevant, organised, and integrated. That is, information is not random and unrelated information stored in a data storage, but organised in a way that provides meaningful data relevant to a problem. Going even further, Jackson (1998) points out that the expert system must have the ability to solve problems directly. Instead of simply reproducing knowledge and information, which data storage is able to do, an expert system should be capable of applying knowledge and actively performing the task at hand. Therefore, while the system will certainly be able to store information, it also actively acts on the data and solves problems. According to Jackson, this is generally done using heuristics, which are rules of thumb that contain the necessary knowledge for solving a problem in a specific domain. Heuristics are simply approximations, and not necessarily optimised or perfect solutions (Jackson, 1998). The advantage is that they can act on incomplete and imperfect data, and they generally provide a weighted response that reflects the certainty (the likelihood of being accurate) of their outputs.

Another important and highly relevant property of expert systems is their ability to address problems of realistic complexity in a manner that matches human abilities and expertise (Buchanan, Randall & Feigenbaum, 2006). Many intelligent programs are essentially simply optimisation and search algorithms (as discussed in the section on genetic algorithms), and

many of these solutions are based on statistical methods and perform well on abstract mathematical or simplified problems. In comparison to any of these statistical search and optimisation tools, expert systems perform well on genuine scientific and commercial issues (Jackson, 1998). An exception – a popular one to the aforementioned simple AI systems – are deep neural networks, which have gained new popularity in recent years. These complex 'neural nets' show great performance on real world problems in many domains, and while a detailed discussion of these networks are beyond the scope of this study, their exceptional performance nonetheless warrants a brief mention.

The literature on expert systems also highlights the need to validate a system in terms of assessing whether domain knowledge is actually contributing to a software systems performance increase (Prieto-Díaz, 1990), an issue that is highly relevant to the current study.

Finally, we need to distinguish between 'knowledge-based systems' and 'expert systems', two terms that the literature sometimes use synonymously. To illustrate the difference using an example, we would call a system that is able to 'talk' about the stock market simply because it holds knowledge of it and is able to reproduce said knowledge in a conversation, a knowledge-based system. An expert system goes beyond this ability, and will be able to apply the stock market knowledge, make predictions about its development, and provide forecasts for specific stocks. This differentiation is relevant to the research discussed in this thesis, as the system employed in the design process is able to apply expert knowledge to game level design, and actively perform analyses of selected properties in each level, as well as actively make suggestions regarding best practice, and most playable levels.

2.4.5 Multi-Agent Systems

Wooldridge (2009) defines multi-agent systems as entities that comprise "multiple interacting computing elements, known as agents" (M. J. Wooldridge, 2009, p. xi). The focus of his textbook, *An Introduction to MultiAgent Systems,* is on agents as computational autonomous entities, similar to Russell and Norvig (2003). The same holds true for a genetic algorithm controlled by a multi-agent system, known as a multi-agent genetic algorithm, or MAGA (Zhong, Liu, Xue & Jiao, 2004). There are variants of multi-agent systems; however, these consider hybrid teams of human agents and computational agents, for example, in human-based genetic algorithms (Kosorukoff, 2001), which will be discussed in more depth in section 2.5.3.

Multi-agent systems, no different to single agents, need to provide a way for agents to understand the environment they are in, and they need to be able to make autonomous decisions without any user interaction. However, not all agents need to have the same sensory system; rather these can be complementary (Wooldridge, 2009). Furthermore, agents have to have a way in which to communicate with one another, for example through messaging.

While the literature provides a vast breadth of different approaches and techniques for designing multi-agent systems, this study utilises a very simple and small system, comprising three autonomous agents and a human designer. Therefore, most of the facilities that have been investigated for multi-agent systems are not needed in this discussion. The agents of this study follow foundational definitions of agents (Franklin & Graesser, 1996; Russell & Norvig, 2003; Wooldridge, 2009) as simple, autonomous, independently acting entities that generate outputs (suggestions) based on simple sensing systems, in order to observe their environment (the breeding pool and population of the genetic algorithm; see section 2.5.1).

2.5 Evolutionary Computation

Evolutionary computation is the field of research within artificial intelligence that encompasses approaches based on the concepts of natural evolution. A common assumption among researchers is that only computational solutions, generally based on a mathematical fitness function, are included in this category. However, as shown in section 2.5.2, this also includes evolutionary computation based on user interaction.

All of these computational approaches borrow from Darwin's theory of evolution, which is based on the idea that individuals (or members) of a population increase their chances of survival and reproduction by way of natural selection. The selection process combines properties of each individual and passes on a variation of the breeding originals to the new generation through inheritance. Darwin's theory and Mendel's concept of genetics formed what the field of biology refers to as 'modern evolutionary synthesis' (Keeton, 1996).

Pimpale and Bhande (2007) provide a simplified breakdown of biological evolution, which highlight some of the main concepts that have been translated into evolutionary computation:

- DNA (deoxyribonucleic acid) is a double-helix molecular structure that encodes genetic information in each cell of all living organisms.
- Chromosomes are subsections of the DNA string.
- Genotypes represent the hereditary information encoded in the DNA. They are the encoding of information.
- Phenotypes are the observable results based on the genotypes. Phenotypes are therefore the visible implementation of the encoding.
- Reproduction (or recombination) is the creation of offspring by two parents, combining parts of each parent's DNA.
- Crossover is the underlying synthesis of an offspring DNA string, resulting in a newly combined chromosome.
- Mutations only happen occasionally and are small, unexpected changes in the resulting offspring DNA. Mutations have a very low probability of occurring, and may lead to small variations, in addition to significant changes induced by recombination. Recombination can be understood as the intentional part of reproduction, whereas mutations are unintended mishaps.

One of the core ideas highlighted by the theory of biological evolution is the concept of 'survival of the fittest'. This means that only the strongest properties encoded in the DNA will be sustained over several reproduction cycles. Therefore, only the 'fittest' offspring's chromosomes will survive over a large number of generations. Weaker properties will eventually be eliminated over many breeding cycles. The 'survival of the fittest' concept is often used synonymously with the term 'evolution' throughout computational literature (Pimpale & Bhande, 2007).

Survival of the fittest (or simply, evolution) also exemplifies what evolutionary computation is at its core: systems that seek to find optimal solutions. Most evolutionary algorithms are search and optimisation algorithms. While they borrow concepts and ideas from biological evolution, they are simply abstractions of biological synthesis, and seek to emulate intelligent or emerging behaviour. Significant examples of such abstractions are genetic algorithms (Holland, 1975), evolution strategies (Beyer & Schwefel, 2002), and genetic programming (Koza, 1992). All of these abstractions borrow from the principles of natural evolution, but vary significantly in their application, and the extent to which the principles are applied.

Evolution strategies, for example, are predominantly designed to optimise solutions in technical areas (Negnevitsky, 2004). This poses a very different application to genetic programming, an idea first introduced by John Koza (1992). Genetic programming attempts to generate computer programs by using evolutionary computation. These evolved computer programs seek to address the actual problem at hand. What Koza essentially proposed (and later implemented) is an algorithm that programs computers to solve problems, rather than directly applying evolutionary principles to optimise a solution. Genetic programming, while considered somewhat rare and less popular than other evolutionary approaches, have been extremely effective and successful in some areas, for example, for addressing engineering problems. Reportedly, this approach led to the first two patentable designs generated by a machine (Koza et al., 2003).

There are several other similar (and varying) concepts of evolutionary algorithms and approaches in the literature; however, an in-depth discussion of all evolutionary computational algorithms is beyond the scope of this thesis. Therefore, only the most relevant concepts are highlighted in the following sections of the literature review.

2.5.1 Genetic Algorithms

Genetic algorithms (GA) are heuristic search algorithms that can be viewed as the most fundamental and earliest algorithms within the area of evolutionary computation. They have been well-researched and are accordingly, well-understood, and are likely popular due to their relative simplicity (Fogel, 2006). Genetic algorithms are at the core of this research for a number of reasons. First, they are incredibly simple to implement, yet deliver potential solutions to reasonably complex design problems (Renner & Ekárt, 2003). Second, they have been successfully applied to design work that seeks to capture the designer's goals, while finding a solution to fairly complex designs, such a urban structures and layouts, and artistic patterns using colour and shapes (Anderson, Buchsbaum, Potter & Bonabeau, 2008; Kelly & McCabe, 2007; Kruse, 2014). Additionally, genetic algorithms have been used in various other fields of research, for example, engineering (Connor, 1996) and computer animation (Marks, 2006), thereby indicating the capability of these algorithms. This study seeks to generate procedural game levels, effectively trying to find at least one solution that captures the designer's intent within a virtually infinite number of possible variations, given the basic terrain and assets being employed in this study.

2.5.1.1 History of Genetic Algorithms

The aforementioned Alan Turing was likely the first scientist to propose a link between computation and evolutionary principles in his famous article, "Computing Machinery and Intelligence" (Turing, 1950). Turing did not go as far as proposing a prototype genetic algorithm, but set the foundation for this idea, which was picked up by Fraser and Burnell in their book, *Computer Models in Genetics* (Fraser & Burnell, 1970). Their computational model effectively entails a genetic algorithm. However, most of the computational literature credits John Holland (1975) as the father of genetic algorithms, which is likely justified, given the depth and scope that he provides in his foundational book, *Adaption in Natural and Artificial Systems* (Holland, 1975). Other notable academics and scientists that expanded on Holland's ideas, and critically examined a number of modern derivations of Holland's original 'vanilla' genetic algorithm include David Goldberg (1989), Melanie Mitchell (1998), and David Fogel (2006). Goldberg provides a fundamental overview and discussion in *Genetic Algorithms* (1989), which appears unmatched not only in terms of its popularity (with more than 70,000 citations on Google Scholar to date), but also regarding the breadth of Goldberg's review. Goldberg does not only introduce genetic algorithms and a number of derivatives, but also observes their application as search and optimisation tools, as well as their use in machine learning.

David Fogel's critical voice on the topic of machine intelligence is discussed in the first section of this chapter. David Goldberg's and Melanie Mitchell's fundamental introductions to genetic algorithms (D.E. Goldberg, 1989; M. Mitchell, 1998), on the other hand, will serve as foundational contexts to highlight the features and traits of a canonical genetic algorithm in this section.

2.5.1.2 Canonical Genetic Algorithm Overview

First, chromosomes are abstracted into binary digits. These binary strings are passed from one population to the next, following a genetics-inspired breeding process. Breeding entails crossover and mutation of the binary chromosome strings. The fitness of each offspring chromosome is then evaluated, using a fitness function to establish the performance of each chromosome toward the overall optimisation (or search within the solution space) goal. Chromosomes with poor performance have a high probability of being removed from the breeding process, whereas chromosomes with high fitness are likely to be selected for

reproduction. This selection is driven by a crossover operator, which mixes parts of two chromosomes into a new offspring, based on their respective fitness. The final step in the process is mutation, where some chromosomes may be randomly mutated, based on a mutation probability factor. This mutation factor is generally set very low, to avoid a high number of randomly occurring changes in the offspring. The main goal is typically optimisation through selection, not random diversity through high mutation. This process is repeatedly applied over many generations, often in the thousands or hundreds of thousands, but entirely driven by the problem to be solved and the solution space (the number of possible solutions). After a number of generations, or when a specific numerical fitness is reached, the process is stopped. Due to the large number of iterations and the combination of targeted, fitness-based selection and random mutation, seemingly complex behaviours emerge, and highly complex problems can be solved; at the very least, results that converge towards an optimal solution can be found (Negnevitsky, 2004).

2.5.1.3 Convergence

Establishing fitness parameters in genetic algorithms to ensure a fast and efficient convergence to the global optimum has been discussed in the literature since their inception (D.E. Goldberg, 1989; M. Mitchell, 1998; Mühlenbein, 1992; Rudolph, 1994). To illustrate the issue at hand, which is one of the significant reasons for why this research takes a few steps beyond the canonical genetic algorithms, we assume the following simple instance following, Goldberg's (1989) example.

We consider the following function:

$$f(x,y) = e^{-(x^2+y^2)}$$

which results in a single 'hill' when plotted using a three-dimensional plot (see Figure 5).

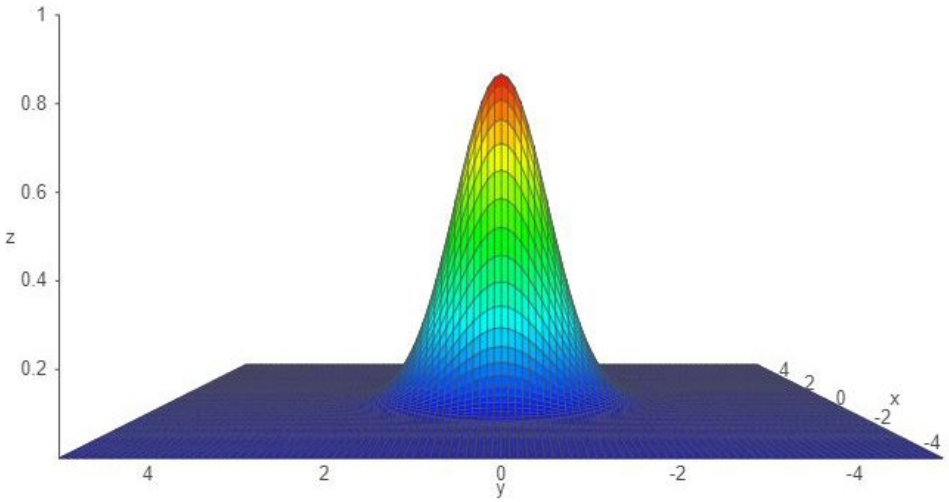


Figure 5 - Plot of $f(x,y) = e^{-(x^2+y^2)}$ resulting in a single hill.

There is only a single maximum in this surface, and a simple hill climbing algorithm will be suitable for finding this maximum: A hill-climbing function, effectively simply comparing values and favouring the higher over the lower neighbour, will eventually converge toward the single hill.

Now, we consider the following function, resulting in two maxima, when plotted:

$$f(x,y) = e^{-(x^2+y^2)} + 2e^{-((x-1.5)^2+(y-1.5)^2)}$$

Figure 6 shows a three-dimensional plot of the above function.

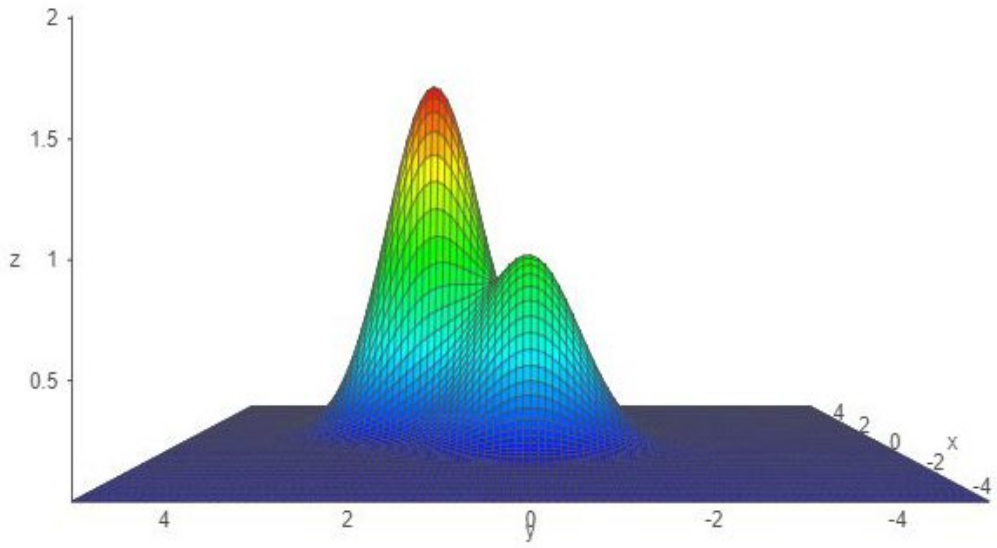


Figure 6 - Plot of $f(x,y) = e^{-(x^2+y^2)} + 2e^{-((x-1.5)^2+(y-1.5)^2)}$ resulting in two local maxima.

It can be imagined that, first, a simple hill-climbing algorithm may fail to find the global maximum, which would be considered the optimum in this search, and a genetic algorithm may or may not exceed the first (local) maximum in order to arrive at a suitable global solution. This depends on the balance between selection and mutation operators, as research has demonstrated (Rudolph, 1994).

Given that the present research is concerned with search spaces that are much more complex than the aforementioned relatively simple mathematical functions, the possibility for the genetic algorithm to get stuck in a local maximum, and terminating the optimisation (search) process prematurely, is significantly higher. This circumstance has been mitigated in this research using a number of techniques, including a careful balance of selection and mutation operators through empirical means, but moreover, by adding a diversity agent to the multi-agent system that controls the genetic algorithm. This will be presented in Chapter 4.

2.5.1.4 Encoding

In genetic algorithms, encoding describes how the parameters of each candidate are translated into data, or simply put, how chromosomes are represented as data types. This is an abstract concept, and does not necessarily reflect how computers handle data at a basic level (binary representation); nonetheless, it relates to the data types generally employed, which may be binary, numeric, or other value types (M. Mitchell, 1998). This is important, as one can imagine that genetic operators may need to act differently, depending on the underlying data type. For example, if the parameters to be encoded reflect colours, which in

turn are represented through three colour channels, red-green-blue (RGB), with a range of 0 to 255 for each colour channel, it is clear that these colours can be treated in at least two different ways: as binary and numeric (integer or float) values. If the data is treated as binary, a change of a higher bit (or a bit of higher significance) will result in a significant change in colour value, which in turn poses a significant change to the colour itself. Accordingly, if the mutation or crossover operator changes not the float or integer value itself, but the underlying bits, an unintended significant change can be triggered. It would, at the very least, be difficult to maintain consistency between different mutations, and in turn, the mutation rate may be invalid.

There are many different approaches to encoding in the literature that attempt to avoid some of the issues outlined above (M. Mitchell, 1998). Though an in-depth discussion of all known encodings, their common use cases, and their relationship with mutation and crossover is beyond the scope of this thesis, it seems important to highlight a few common examples, namely binary encoding, value encoding and permutation encoding.

Binary encoding is one of the most common variants, and is likely also the most fundamental form of encoding, given that it is used in canonical genetic algorithms (D. E. Goldberg, 1989). The entire chromosome is encoded as a binary string.

2.5.1.5 Fitness Function

The fitness function is at the core of a computational genetic algorithm (as opposed to interactive genetic algorithms; see section 2.5.2). It assesses the performance of candidates in a population, based on criteria addressing the issue that the algorithm seeks to optimise or solve. When implementing a fitness function, the goal is to find the global maximum (or minimum, depending on the start condition of the genetic algorithm), with a large possibility of success, and in the shortest possible time (Bentley, 1999). A number of test functions to verify the implementation of the genetic algorithm and the fitness function in particular have been established. Examples include the aforementioned hill functions (section 2.5.1.3). I will not go into great detail about them, as these approaches are addressed in-depth in the existing literature (Bentley, 1999; D. E. Goldberg, 1989; M. Mitchell, 1998), and they have little relevance to this study, given that an interactive evolutionary system is employed to drive the evaluation of individual populations. The following section (2.5.2) highlights the

differences between a computational approach, employing a mathematical fitness function, and using an interactive method, using a human user to assess the population.

2.5.2 Interactive Genetic Algorithms

Given that this study is concerned with computational support for design tasks, an examination of a special class of genetic algorithms is required. Canonical genetic algorithms are simple to implement and extremely effective in basic search and optimisation tasks, as shown above. However, one of their predominant features, the fitness function, poses challenges in terms of its implementation when issues such as aesthetics, appeal, and attractiveness come into play (Takagi, 1998). These features are not easily captured with a mathematical function, and instead of trying to describe the problem mathematically in order to create a suitable fitness function, interactive genetic algorithms (IGA) employ the help of a human user to perform the selection task. This removes the requirement for a fitness function and allows for addressing (mathematically) complex issues and applying the benefits of genetic algorithms to domains that are not based on a numerical foundation, such as engineering and computer science. IGAs are a subclass of algorithms used in the field of interactive evolutionary computation (IEC). The notion of employing humans to perform some tasks in evolutionary computational systems was originally formed by Richard Dawkins in his foundational book, *The Blind Watchmaker*, in the book's third chapter (Dawkins, 1986). Dawkin's 'biomorphs' are based on Darwinian theories and require human users to evaluate fitness (aesthetics and appeal) interactively (Dawkins, 1986). This idea was extended by Karl Sims (1992), who assumed that human users were able to perform evaluations, more specifically, selections within an evolutionary system, without any knowledge of the underlying processes (Sims, 1992). This is similar to our understanding of IGAs, in that the genetic operators are fully independent and the selection operator does not need to have any access to what the recombination operators are acting on. Takagi (2001) coined the term 'interactive genetic algorithm', and performed a number of investigations into their application in various fields, but in particular, within a design context. He found that IGAs are not only a vehicle for performing optimisation tasks in design, and that they often outperform pure genetic algorithms and manual computer-supported design, but that IGAs also offer a high level of flexibility, and that evaluations (the selection process) can be altered while the task is performed. If a (static) mathematical function is used, this is simply not possible. This

feature of IGAs eventually led to their recognition as ‘novelty generators’ (Gu, Xi Tang & Frazer, 2006).

It is worth noting that the human user often makes selections based on an abstract representation of DNA, rather than interacting directly with either the genotype or the phenotype. For example, instead of making decisions by looking at a bit string that encodes pixels (genotype), or even the associated colour values of each pixel (phenotype), the user simply selects from a number of images formed by a large number of individual pixels. In this way, the underlying bit strings and associated colour values are hidden from the user, but selected through the choice of full images for breeding within the genetic algorithm. Based on the user selection, the genetic algorithm creates a new population, including the crossover and mutation of bit strings (which represent the colour values of pixels). The new generation is then presented to the user in the form of a number of rendered images. This highlights an important implication that the developer needs to decide about: if low-level crossover is performed on bit strings, this may lead to odd, inconsistent, and potentially invalid colour values, which in turn leads to images that may have invalid pixels. Bit string length, as well as crossover point, both play an important role in the recombination process. It is also possible to perform crossover on higher level abstractions and manipulate the phenotype instead of the genotype, for example, recombining larger parts of the chromosome to keep legal colour values intact, as a means for ensuring that pixels always receive a valid colour property.

Another important point to discuss is the impacts that interactive genetic algorithms may have on users. While it is an extremely elegant way to avoid having a mathematical fitness function, which in some creative contexts may be virtually impossible (although modern deep neural networks perform astonishingly well as a statistical solution to some creative problems), interactive genetic algorithms can lead to user fatigue if a larger number of iterations is required to solve certain tasks (Takagi & Iba, 2005). There have been attempts to counteract this issue, and I have myself contributed work in this area (Kruse & Connor, 2015). Adding an agent to detect user preference by analysing user input shows some promise; however, the results have been moderate to date, and more research is needed to improve this weakness of the approach.

IEC has been successfully used in PCG studies and the literature offers a number of relevant cases. Examples include interactive systems to generate city models in real-time (Greuter,

Parker, Stewart, & Leach, 2003) instead of using non-real-time generators (Kruse, 2014; Parish & Müller, 2001). Other work looked at game map generation to offer personalised choices to players (Raffe, Zambetta, Li, & Stanley, 2015).

2.5.3 Human-Based Genetic Algorithms

Kosorukoff (2001) takes the idea of IEC further and suggests that not only the selection operator, but either the selection or recombination operator can be assigned to a human or computational agent (he also introduces the idea of a multi-agent system in the context of genetic operators). Kosorukoff takes an organisational standpoint and assumes that both these tasks will be 'outsourced' from the algorithm to agents. The genetic algorithm is essentially the organisational framework, or shell, and the computational and human agents are the contractors (Kosorukoff, 2001). Treating genetic algorithms as multi-agent systems has several advantages over traditional genetic algorithms. First, it allows for combining human intellect and computational performance into a single evolutionary framework. Second, it addresses the issue of human limitations in a more flexible and possibly broader manner. Human and computational agents are able to complement each other in both selection and recombination tasks. Finally, it offers flexibility that is not found in traditional genetic algorithms, in that it allows for combining several selection agents and multiple recombination agents into a single system. Either of these can be human or computational, or even a hybrid.

2.6 Summary

The literature review for this study touches on a variety of topics that highlight the interdisciplinary nature of this research. While the review of design, and specifically game design literature is nowhere near complete nor very broad, a gap in knowledge has been identified, and some of the most relevant parts of the literature have been addressed. This study seeks to enhance the game designer's ability and effectiveness, not to replace them. This study also explores the boundaries of human and computational design agents as part of an overall design process for computer game levels. In doing so, it is hoped that this exploration will also make computer game level design a little bit easier, and therefore, more accessible to a broad range of game designers.

3 Methodology and Research Design

I apply a pragmatist approach in this study. Pragmatism is a world view that arises from actions and consequences. It is also a research paradigm that is problem-centred, and orients itself toward real-world practice (Creswell, 2014). I believe this is not only the most appropriate paradigm for this particular study – as it aims to construct a novel workflow and is located in an applied context – but also reflects what I, as a researcher, am best at. I am as much a practitioner as I am an academic. Relating my research back to my practice in a problem-oriented manner feels most natural and consistent to me. The study is conducted using a mixed-method approach, including semi-structured interviews and questionnaires, in-game metrics, and observations. This section unpacks the methodology into the components employed in this study.

3.1 Research Questions

As indicated in section 1.2, this research is guided by the primary research question below:

> How can intelligent agent systems be employed to generate, and more importantly, evaluate procedural game content? Specifically, can agents based on cognitive models evaluate computational designs in computer games?

I will unpack this highly condensed question in this section and provide two sub-questions that I have used in order to break the overall project down into manageable sections. These questions address issues that evolve from engaging with the research question, despite the fact that they are not tightly aligned with the specific terms of the main question.

I wished to capture the views of experienced game designers as they related to the proposed content evaluators. This was based on the assumption that it was possible for expert designers to create a cognitive model of decision-making processes in the first place. Therefore, I asked:

> (Q1) Are cognitive agents as game content evaluators considered a useful addition to game level design by experienced game designers?

Furthermore, I was interested in whether cognitive designer models could be created by employing rule-based systems, which guided an additional sub-question:

(Q2) Can a cognitive model be built as an expert system, and can it employ knowledge extracted from secondary sources, such as personal accounts of expert game level designers?

The following sections discuss methods that helped me to obtain meaningful responses to these questions.

3.2 Methodology

Due to the nature of this study, which is concerned with a range of interdisciplinary issues such as game prototype development, participants conducting design tasks, as well as statistical data collection and analysis, the methodology borrows a number of different approaches from several cross-disciplinary methodologies. It is driven by a pragmatic research philosophy using a mixed methods approach.

The research in this thesis follows an interpretivist paradigm for its epistemological position. That is, understanding the world through the interpretation of research data, or as Bryman (2012) puts it, through the examination and interpretation of participant responses.

Furthermore, a constructivist ontology drives the researcher's view of how sociological factors are formed, and how our general understanding of meaning is generated. Subsequently, these constructions of meaning are thought to be reflected in participants' responses, which in turn assists in articulating a general understanding of interactions between individuals, and to an extent, the interactions between machines and humans, too.

The overarching methodology can be described as practice-led (Candy, 2006). This study seeks to form new workflows, and its core output is a novel method that uses human and computational agents to drive procedural content-generation for games. Practice-led research derives from practice-based methods, and shares a number of similarities with them (Dean & Smith, 2009). The most significant difference is probably the research output itself. While practice-based methods generally result in an artefact, practice-led research seeks to enhance knowledge and understanding of processes and practice itself. Practice-led research advances techniques, methods, and practices, and this study presents a new way of creating game content. Its goal is therefore to advance workflow.

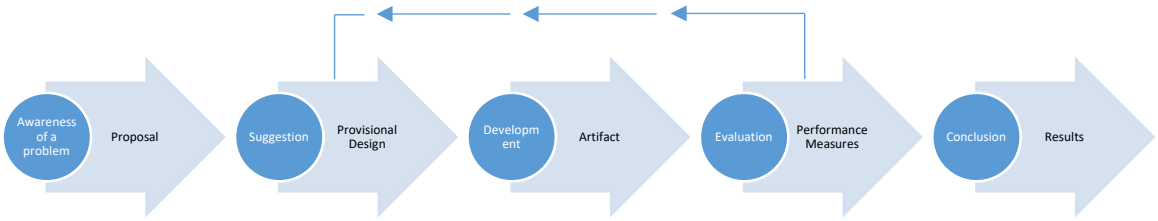


Figure 7 - Design science research process diagram.

Advancing and improving workflow is also at the core of design science research (Hevner, March, Park & Ram, 2004), which aims to widen knowledge about human behaviour, and make predictions about the consequences of such behaviour. It is used to improve human and organisational performance through the creation of prototypes and artefacts, and presents itself as a five-step iterative process. Vaishnavi (2008) links each of these five steps directly to an expected outcome (Figure 7). The first step is to become aware of a problem within the chosen field of research, or in a specific discipline. The outcome is a formal or informal research proposal. The next step is a suggestion that resembles the initial idea for establishing a possible solution. This idea typically draws on previous knowledge or already existing elements of known solutions to other problems and is generally combined with an additional new element that is likely to improve existing solutions. The resulting provisional outcome is generally only an initial design prototype or tentative solution, which is refined in the third step of the design science research process (Vaishnavi, 2008). This third step, termed 'development' is the core of design science research and necessitates a high level of skill, creativity, and intensive use of software development tools. The aim of this step is to create a viable artefact that promises a sound solution to the initial problem. The next phase evaluates the artefact and obtains performance measures. Based on these performance measures, insights and possibly new ideas trigger a new iterative cycle that restarts at the suggestion phase. This cycle continues until sufficient performance measures are reached, or no additional potential improvements of the artefact can be identified. The cycle completes when the final phase, called 'conclusion' is entered, the purpose of which is to summarise results, often into two groups: results that are firm and results that are considered loose ends. Firm results comprise repeatable outcomes that can be verified; loose ends are anomalies that defy explanation and require future research. Vaishnavi (2008) emphasises that design

science research is generally depicted as a linear process (a notion I follow in Figure 7), and in practice is typically cyclic, as few projects terminate after the first evaluation, and most require a substantial number of iterations to generate meaningful results.

The methodology is essentially a two-part approach, the first part being concerned with methods that serve an iterative design process, and the second part being a qualitative evaluation of the output of this design process. The latter is triangulated by quantitative methods to make the qualitative part more robust. The multi-faceted methodology applied to this research has allowed me to understand and experiment with different approaches through multiple methods, which compliments my future development as a researcher.

While the methodology is somewhat similar to that employed in my master's thesis research, it is expanded and applied in a way that will hopefully foster a better understanding of real-world problems, allowing me to design a potential solution, while expanding scholarly knowledge in the best spirit of practice-led research. The following sections of this chapter deconstruct the individual components (methods) of the methodology used in this study.

3.3 Research Design

The research design employed in this study is at its core an evaluation of the game level design tool prototype with participating game designers. This section of the thesis outlines the experiments, explains the reasoning behind the experimental design decisions, and links them back to the existing literature to highlight where they fit in.

3.3.1 Game Level Design Tool Evaluation

In this subsection, I will break down the evaluation of the prototype game level design tool. The emphasis here is on *prototype*. I had no intent to create a tool that was finished in terms of UX/UI experience for the user. I simply needed a vehicle for evaluating the quality of the multi-agent system.

3.3.1.1 Structure of the evaluation

The evaluation of the prototype tool was conducted with the goal of gaining a basic understanding of the effectiveness of the multi-agent system, and to gather initial feedback from professional game designers in order to form a foundation for future developments. The evaluation followed a three-step process.

First, a pre-participation interview was conducted in order to gain a detailed understanding of who the participant was in terms of their computer game play preferences, their level of experience designing games themselves, and their capacity to create FPS game levels in particular. I also captured basic demographic data to review whether there was any bias in terms of gender and age. The pre-participation questionnaire may seem unusual, as it captures data that will generally be part of the participant selection process. I will discuss the reasoning behind this in section 3.5.1, but in short, this is linked to the small game design community in New Zealand. I required a sufficient number of participants, which hindered me from being too selective. In order to counteract the potential lack of variety in my participant group, I needed to gain a deeper understanding of what their qualities were as it pertained to this study.

The second part of the evaluation was an assessment conducted with the participant, using the game level design tool. The user was asked to use the tool for a number of runs to create several generations of candidate solutions, until a candidate that was deemed a playable and potential enjoyable level was found. Some of these runs had no other agents in the system than the human designer. A number of runs had all computational agents active. By using no agents and letting the designer essentially conduct the entire selection process, without any suggestions and interference by computational agents, I aimed to create a baseline for comparing these runs with the full system being active. This is, of course, based on a number of assumptions and the quality of the results, and some limitations of this process are discussed in later chapters. The fact that the agents were inactive in some runs was hidden from the participant. During all evaluation runs, the eye-tracker and keyboard and mouse telemetry capture was active to observe whether there was any distinction in user behaviour between different participants. All data capture happened transparent to the user in the background. Additionally, the activity of agents was not directly disclosed to the user, neither verbally nor through visual cues.

Strictly speaking, the runs conducted with inactive computational agents were merely simple interactive genetic algorithms (Takagi & Iba, 2005), as only one agent – the human – was part of the system.

The third and final part of the game level design evaluation was a semi-structured interview, conducted with the participant. Given the very different levels of experience of participants,

the choice to conduct semi-structured interviews rather than questionnaires or structured interviews proved to be extremely effective, as I could divert from the script for the interview in order to clarify additional questions that arose during our conversations. I was also able to capture additional comments that I had not considered at the time of designing the interview questions.

In hindsight, I would have made some changes to the semi-structured interviews in particular, for example, reducing the number of questions. A detailed discussion of potential improvements of my evaluation of the game level design prototype tool will be presented in the results chapter.

3.3.2 Think-aloud and Observation

According to Charters (2003), "Think-aloud is a research method in which participants speak aloud any words in their mind as they complete a task", which can be traced back to the concept of 'inner speech', originally noted in 1962 by Vygotskiĭ (2012). The premise is that words and phrases captured through the think-aloud approach serve as close representations of inner speech, and are often expressed as incomplete sentences, but as a nonetheless useful reflection of a person's thought processes. Presumably lacking the mental filter that leads to active reasoning and the formulation of spoken sentences as part of a conversation, these verbal fragments are considered a valuable tool for accurately accessing a participant's thoughts, as opposed to a delayed explanation (Charters, 2003).

However, think-aloud data is likely to be incomplete, and its results also may vary between individuals, depending on personality traits and the ability to manage additional cognitive load (Ericsson & Simon, 1993). Ericsson and Simon emphasise two issues in this regard. First, a task with low-level complexity can result in insufficient cognitive load, which in turn does not produce any meaningful outlets of inner speech. Secondly, cognitive overload as a result of choosing a highly complex task can negatively impact a participant's ability to manage both the task and think-aloud function, which may lead to less than ideal results when using this method.

The above factors are generally mitigated through triangulation, for example, by retrospective questioning (Charters, 2003). This study makes use of semi-structured interviews to alleviate the risks highlighted above.

This research also follows Charters' (2003) suggestion of applying a qualitative lens to the data, and concurs with Ericsson and Simon (1993) that a balanced cognitive load needs to be the foundation for this type of data collection. The prototype that participants use to generate game levels provides sufficient complexity to allow for a meaningful release of inner speech, but is also simple enough to assume that none of the participants will struggle with the additional cognitive load of think-aloud methods.

3.3.3 Eye Tracking

Eye tracking is a quantitative method for capturing gaze data from participants. It uses a specialised device to capture the position of the pupils using infrared cameras, triangulates these positions, and allows a very accurate estimate of where the user looked on-screen (Duchowski, 2007). Gaze position is associated with the visual attention of the user (J. H. Goldberg & Kotval, 1999), which provides cues to what the participant is actually interested in. I added this method to triangulate responses from semi-structured interviews.

Neuroscientific research suggests a high degree of dependency between eye-head coordination and visual processing. Rich and dynamic retinal input is gained through constant adjustment of the gaze through head and eye movement (Andersen, Bracewell, Barash, Gnadt & Fogassi, 1990; Einhäuser et al., 2007). This study employs a simplified sensory coding model, using only the resulting gaze. The eye tracking device principally supports both eye and head tracking; however, when the first few data streams where collected from designers, head tracking functionality was not yet implemented in the supporting Software Development Kit (SDK) by the manufacturer and was only added halfway through my study. In order to retain data consistency throughout the study, I decided to continue collecting only gaze data in subsequent trials, rather than undermining consistency by adding the head tracking data. This can be addressed in future studies.

To obtain an accurate gaze position, a calibrated tracker is required. I conducted the calibration process prior to each trial with participants. Given that the resulting data is simply a stream of x and y positions at 60 Hz or 120 Hz, this data requires post-processing. There are several ways to visualise the tracking positions, the most common being the use of heatmaps. These (often colour-indexed) representations show a stronger colour and density where gaze had been focused most of the time, and a weaker colour and lower density for where the user looked at less. The heatmaps can be superimposed onto the original screen material to

visualise the gaze pattern of the user in context. Figure 8 shows an example of such an overlay, visualised on a Wikipedia webpage. The light purple areas represent an area that the user looked at for only a limited time, whereas the bright orange-red areas indicate an area on which the user focused a significant amount of time. The latter can be interpreted as areas of high interest, or areas that capture the highest attention.

Figure 8 - Heatmap of eye tracking on a Wikipedia webpage.

3.3.4 Semi-structured interviews

This research employs semi-structured interviews conducted with game designers to capture their experience of using the game level design prototype. Following a pre-participation questionnaire, which simply captures demographics data such as level of experience and personal preference with regards to computer games in general, the actual experiment was conducted using input recording, eye tracking, and think-aloud methods. Then, a semi-structured interview concluded the participation. This interview was designed to capture any thoughts, ideas, and views about the prototype and the design process that could not be captured by any other means. The relatively unconstrained nature of semi-structured interviews, which essentially use a number of indicative questions, but allows the researcher to divert from protocol and explore other interesting paths, was chosen deliberately. First,

after initial tests and following previous experiences with similar situations, it was clear that every designer would likely have very individualised experience. As such, a general, rigid, one-fits-all interview or questionnaire was deemed too restrictive to capture any additional insights that may arise. Second, experienced game designers are rare, and I was fortunate enough to have had a number of very experienced, seasoned designers available. As such, I wanted to capture their views as completely as possible, as doing so would provide an opportunity to tap into abundant game design experience first-hand. An even more important, but less obvious motivation, inspired the idea of using semi-structured interviews: the opportunity to verify some of the heuristics that had been previously derived from expert game level designers' blogs, books, and journal articles. Asking specific questions that challenged these heuristics, and asking questions that could verify them, the resources for future heuristics development – and therefore for the development of future expert agents – could be expanded and rigorously examined.

The interview questions were intentionally designed to allow the participants room to elaborate and talk about their experience. At the same time, I wanted to ensure that they were made aware of the presence of some computational agents in some of the tests, to see how they responded to them. The full list of questions is listed in Appendix D.

3.3.5 Questionnaires

Questionnaires are often assumed as being quantitative methods of data acquisition, however, they can also be employed in qualitative research, in particular when no statistically valid sample size can be achieved (Bryman, 2016). In this study, questionnaires are treated as qualitative data for this reason. Using them allowed me to gain a general idea of the level of expertise of my participants, and their preferences for genre and specific games. This could also have been achieved through additional interview questions, but I wanted to keep the time of the semi-structured interviews as short as possible, given that I had to conduct them after my participants had completed a presumably fatiguing exercise of using a variant of an interactive genetic algorithm. Asking them to complete a questionnaire alongside the required consent forms was deemed to be the best approach.

The pre-participation questionnaires (see appendix D) included a number of questions with scaled responses, and some questions allowing for open-ended answers. Examples include, 'To what extent do you enjoy playing video games?', with a four-level response ('not at all' to

'very much'). It should be noted here that a Likert scale (Brown, 2012), which typically employs five-level responses, was intentionally altered to a four-scale response in order to remove the 'undecided' middle option. A finer granularity seemed unnecessary for gathering simple demographic data, but a clear result indicating whether participants had a positive or negative bias in their responses to scaled questions was used to make analysis of the data easier and clearer.

3.4 Design Prototype Evaluation

This phase of the study employs primary data sources obtained through telemetry of my software prototypes, as well as questionnaires and interviews. In game design research, this data is also known as 'player experience data' (L. Nacke et al., 2009), and in a wider sense, it represents the data that is based on human-computer interaction through the use of keyboard, mouse, and other controllers. Secondary data derives from sources that have been created by someone else, and that have already been published. In the case of this study, secondary data sources were books and blogs about FPS level design processes, where renowned game (level) designers documented their thoughts and reasoning behind the structure, layout, theme, and any iterations of popular game levels. The methods for collecting information from these secondary data sources are quite different to the collection of original data, and therefore, a description of my methodology is provided in this section.

The origin of the data is not the only foundation for selecting suitable methods for my methodology. The data collection method, and how much of a potential bias had been induced into the dataset by each participant were also important considerations. Subjective and objective measures for obtaining data will also influence the choice of adopted methods (L. E. Nacke & Lindley, 2010). Subjective methods include questionnaires and semi-structured interviews. Objective measures relate to data collected in the background, such as telemetry and eye tracking. All methods are subject to interpretation by the researcher; therefore, each dataset may be biased, based on the rigour of the methods employed. Subjective methods add an additional layer of potential bias, and measures for reducing the impact of such biases must be employed. Triangulation is one such measure (Bryman, 2012).

Primary data collection can be conducted through various means. For this study, pre-participation questionnaires were conducted first for the design prototype evaluation in a bid to gather basic demographic data, such as previous experience and the preferences of each

participant. This data was considered potentially useful for interpreting certain behaviour during the test. For example, a professional game level designer with decades of experience in the industry may take a different approach than an inexperienced recent graduate, with only a couple of years of experience in the field. Alternatively, someone with a long-running passion for adventure games may be inclined to select larger levels that offer more room for exploration than a fan of close-quarter, first person combat games. Second, metrics were collected within the design prototype while the design process was conducted by participants. This includes mouse positions (selections), the structure of each candidate's solution, the ranking provided by the agents, the time each run took, as well as eye tracking data, collected with the device described in section 4.1.9.

In addition to distinguishing between primary and secondary sources, section 3.2 also establishes that we need to consider two different types of methods used for analysing the data, that is, qualitative and quantitative approaches. Therefore, the following subsections are categorised into two groups representing these methods.

3.4.1 Qualitative data analysis

Any qualitative data, derived either from interview responses or secondary sources such as game designer blogs and books, was coded in NVivo.

The literature is divided on how coding should be performed, and moreover, whether the data as a whole should be coded, or only its more significant parts. Some researchers suggest that the rich and nuanced nature of qualitative analysis requires the entire data corpus to be coded (Wolcott, 1990). However, given that the data in this research reflects more of a technical and process-oriented collection of methods, and less of a social setting in which nuanced psychological factors should be examined, the current study adopts a very pragmatic and practice-oriented approach (Bryman, 2016; Creswell, 2014). Therefore, data collected via semi-structured interviews and through the think-aloud method were coded based on a selective, reduced data collection. I believe this approach is justified, considering that qualitative coding is an iterative, cyclic process, in which data is linked to an idea or notion, and ideas are linked back to other data (Saldaña, 2012). Filtering the part of data that may be of value from what may be mere noise (again, given that the semi-structured interviews facilitate a rather loose interview style) can be viewed simply as the first iteration of removing some of the less valuable data. Anything that is even remotely related to the topic made it

into the first iteration of the coding process and may only have lost some significance in subsequent coding cycles, which eventually resulted in a very focussed and reduced set of codes. Establishing the number of codes required is another topic that divides the literature. Some researchers suggest that 20 to 100 codes are appropriate (Wolcott, 1990), whereas others propose the number (of provisional codes) to be roughly five (Creswell, 2014). This study employs six codes that encapsulate the most significant traits identified in secondary data sources, and three codes for primary data.

The analysis of qualitative data in this study employed two of the common qualitative analysis approaches proposed by Hsieh and Shannon (2005). First, conventional content analysis, which applies to studies that start with observations, define their codes during data analysis, and which derive their codes from the data itself (Saldaña, 2012). As a result, most of the codes in this study are descriptive and only a few have been developed using NVivo coding. NVivo coding was used to derive meaningful data from secondary data sources in order to produce the heuristics for the cognitive designer model.

The second analysis method was applied to the primary data collected through observation and interviews with designers during the evaluation of the prototype. This method is called summative content analysis, an approach that assumes that some keywords can be derived prior to data collection and analysis, for example, through the researcher's interest or existing literature (Hsieh & Shannon, 2005). Section 3.4.1.1 describes how these methods were used in more detail.

3.4.1.1 Coding in NVivo

The coding process was conducted in NVivo using a three stage process, as per Saldana (2012). The pre-coding stage is essential for gaining an initial understanding of the data. It serves to form an understanding of phrases that are common, and themes that can be identified across all responses. Tools that can be used include Word Queries, Text Search, and visualisation through word clouds and word trees (Saldaña, 2012).

The second stage of analysis seeks to assign labels to nodes. A common strategy for creating labels is to refer back to the research question (Vaismoradi, Turunen & Bondas, 2013). In the case of the current study, I used sub-questions to develop labels linking my data to the primary research question, and that provide meaningful responses to said sub-questions.

These are reflected in the subsections of chapter 5.2.3. The labels used are ‘usability’, ‘multi-agent system’, and ‘feature suggestions’.

The final stage is the post-coding stage, which drives how findings are presented. There are various ways in which to approach post-coding, including visualisations such as project maps, concept maps and charts, or by presenting the findings as closely related to the wording used by participants (Bryman, 2016; Saldaña, 2012). I chose the latter method, for the simple reason that my participants are experts in the field of game design, which resulted in a fairly consistent choice of terms. Another reason for adopting this approach was the low volume of text this study produced. Identifying clusters or high rates of repetition for specific keywords would only occur where large quantities of text were available. This, however, was not applicable to the primary data of this research.

3.4.2 Quantitative data analysis

This study employs a number of different quantitative methods, as indicated above. Analysing the data is achieved through various means. Some data lend itself to be visually analysed, for example, through heatmaps (see section 3.3.4) rather than numerical analysis. Visualisation can be a powerful tool for making numerical data that is not particularly intuitive more accessible. In other cases, graphs based on statistical data can provide new insights, and provide easier and potentially more accurate interpretations. Initially, I decided to generate click-maps (heatmaps representing the selections) for visualising user input. These are, however, difficult to read, given that user input does not create rich or dense datasets the way that, for example, eye tracking does. Very few datapoints were available (one for each user selection), where a heatmap did not allow for easy interpretation, even when generated from all available runs for a user, or even across all runs of all users. Therefore, after the first attempt to generate meaningful outputs from my telemetry data, I began incorporating RStudio (2015), a development environment (IDE) for the statistical language R, into the process. R offers statistical computing and visualisation in a convenient package, and is open-source and cross-platform, with a high level of flexibility in terms of data ingestion and processing (Verzani, 2011). As most of the telemetry data I was streaming from each run of the multi-agent system was captured in simple comma-separated value (CSV) files, merging and converting data for processing across a large number of files was much easier compared to manual data preparation. I decided to employ RStudio to process all data that could not be

analysed by either qualitative means (for example, NVivo coding), or by visual aids such as mapping to screen space (see eye tracking in section 3.3.4).

3.5 Ethical Considerations

The involvement of participants requires prior approval by Auckland University of Technology's ethics committee (AUTEC). The ethics application for this study includes two pre-participation questionnaires for both game designers and play-testers, indicative questions for the semi-structured interviews conducted with the game designers, and a main (post) questionnaire for play-testers, as well as observation protocols. Forms can be found in Appendix C.

The main considerations include that this research poses minimal (low) risk to any participant. Potential risks have been mitigated; most importantly, participants are able to withdraw from this study at any time without needing to specify reasons for doing so. In the even participants felt uncomfortable, which is rarely the case where screen applications are involved, they were advised to withdraw from the study. Finally, these experiments were conducted in a controlled environment under the supervision of a researcher.

3.5.1 Participant selection

The creation of computer game levels was conducted with professional game designers. Participants were selected through an open call at local game events, social media and through contact networks into the local game developer industry and all levels of experience were considered. Given the relatively small game design industry in New Zealand, I decided not to restrict participants by using a very narrow selection process, but rather, to capture data from novice and expert game designers alike. Furthermore, there was no restriction as to whether they had FPS level design experience in particular; however, through the information sheet, this was indicated as a preference. I also felt that it would be beneficial to gather foundational data on how research into this topic can be conducted, and therefore, decided to remove most restrictions on my participant selection process. The main criterion was that the participant had to be a game designer with game level design experience in a professional capacity. I believed that I could treat this project as an initial, exploratory study for gaining insight into how qualitative research, using quantitative and qualitative methods in the domain of game design, can be conducted. This follows Bryman (2016), who suggests that mixed methods can lead to better results, as opposed to employing purely qualitative

approaches. The literature focusing on procedural game content generation and evaluation also mostly employ either strictly qualitative or quantitative methods, which highlighted the necessity for testing a mixed-methods approach.

3.6 Summary

In this chapter, the methodological approach, research design and ethical considerations have been outlined. The reasoning behind several research design decisions have been given. The next chapter is going to unpack details of the design prototype implementation and in particular the structure and function of the Multi-Agent System, which is one of the main contributions to knowledge of this thesis.

4 Design Prototype Implementation

The main output of this thesis is the evaluation of a multi-agent system. A prototype software tool needed to be implemented for this evaluation, thus enabling human designers and computational agents to conduct level design experiments as a team. Prior to this prototype implementation, resulting FPS game levels of the prototype software were tested with players in order to establish whether these were indeed playable, and despite their simplicity, not a distraction from the core aim that any game has: gameplay.

This section of the thesis discusses both prototypes used in the experiments, while section 3.3 explains how the experiments were designed and conducted.

I believe that it is important to provide a brief explanation of the choice of computer game used in this study, and in particular, to justify the simplicity of the prototype used, which was not born out of necessity or limited by the scope of this study, but a conscious research design choice. First, I do not believe that first-person shooter games are 'bad' or have a negative impact on those who play them, as is often portrayed in popular media and the press. My hope is that by consciously making first-person shooter games the centre of this study, I can make my own small contribution to counteracting the negative image that many readers may have of such games. These games were among the first point-of-view computer games to be developed once the performance of hardware and software allowed for creating three-dimensional worlds in real-time. First-person shooter games are among the most successful Esports games, and draw very large crowds into live venues, as well as large numbers of viewers who watch live streaming events and gameplay. To substantiate this statement, the Global eSports Market Report projected an increase of revenue of more than 40% to USD696 million in 2017 in this area (Warman, 2017). First-person shooters make up 27% of all games watched, representing the second largest viewer group online and on television. Twitch.tv, an online streaming service, registered 2.2 billion views for its top 100 games, with *Counter-Strike: Global Offensive* ranking second among all games watched (Gaudiosi, 2016).

It also feels important to not only consider the simplicity of the prototype's implementations; the notion of 'simplicity' has also been applied to the design prototype itself, based on a design philosophy known as minimum viable product (Moogk, 2012; Münch et al., 2013). This design approach is often used to produce a simple prototype of a potential product, which

allows to trial all fundamental functionality in user testing or early access market research. Instead of implementing all functions desired in a final product, making it ready to go to market, these early prototypes serve as vehicles for confirming or refuting key features, without rendering them unusable by default. These prototypes aim to find a balance between initial investment and time spent on polishing the artefact or product, and provide the ability to rapidly gain insight about their usability (Moogk, 2012). Accordingly, the prototype designed for this research is basic and simple, but includes all the key technologies for serving as an evaluation platform for the multi-agent system that I intended to implement. The same holds true for the resulting game levels. While the assets employed in these levels may be basic, and though significant compromises were made in terms of level complexity – for example, a flat terrain and only reasonably simple cell-based layout options – all the fundamental gameplay mechanics are present, and simple player versus player matches could be implemented.

4.1.1 Resulting Game Levels

Prior to discussing in detail the core of this study, which is the design, implementation, and evaluation of the level design tool in the form of a prototype, I wish to break down the resulting levels in order to render some of the design choices made for the level creation tool easy to understand. I also want to show how the resulting levels relate back to the simplified top-down views that were shown in the prototype design tool to participants. This relationship was also verbally explained to the participants during pre-participation briefings to ensure the designers would fully understand the choices they made. As I do not want to pre-empt any of my findings in this chapter, an explanation of how these pre-participation briefs were conducted, is given in context of the results in section 6.2.5.

4.1.1.1 Genre and Setting

Following a number of current FPS games such as *Counter-Strike: Global Offensive* (2012) and *Insurgency* (2014), as well as *Insurgency: Sandstorm* (2018), I opted for a deserted urban environment without any non-player characters (NPCs). It is not fully post-apocalyptic (overgrown and destroyed, with a lot of debris), but rather, deserted in a state that suggests the occupants of these levels had only recently left. The buildings are intact and the streets are only slightly cluttered from the aftermath of whatever caused the population to leave. The main reason for these design choices had not been simply to follow the popular game

examples noted above; rather, I felt that removing the complexity of debris, strong overlays of foliage in the form of overgrowth, and perhaps piles of burning vehicles or other remainders of recent warfare, would assist in focusing my results on simple mechanics such as cover, choke-points and lines of sight. It was also suggested in early reviews of my proposal that I remove complexity as much as possible in order to receive evaluations without large amounts of noise. The final levels reflect an attempt to balance high visual quality that ensures an immersive experience for the player, while keeping the number of elements that have an impact on game mechanics very small, in order to gain cleaner results, and to be able to draw defendable conclusions from my evaluation.

4.1.1.2 Game Mode

Only one game mode has been implemented at this stage, i.e. a simplified conquest-type game mode with a single flag point. There are a number of reasons for keeping the resulting game simple. The obvious reason is practicality within the scope of this study. Additional game modes require additional complexity for menu selections and more debugging before the actual evaluation of the design prototype could be started. Furthermore, a single focus point for players would help remove unnecessary noise in the data in future play-tests. The single conquest mode also allows for more structured play, based on my own experience. It is worth noting that I use the terminology of the *Battlefield* (2002) series here. A very similar game mode is called 'bomb defuse' in *CS:GO*.

4.1.1.3 Level elements

The resulting levels consist of a few core layers of elements that follow conventional FPS level designs, and which relate directly to game elements as introduced by Järvinen (2009). It is worth noting that most of these layers are directly influenced by the designer using the design tool. These elements are considered important to gameplay and game mechanics, as their design choice has a direct impact on how the mechanics (and the resulting gameplay) work. These comprise the following:

1. Terrain
2. Streets
3. Buildings
4. Flag pole

5. Shipping containers

In addition to these layers of significant importance, a few additional, purely cosmetic layers were added, all of which were assumed to have no impact on gameplay, and only serve to enhance the visual impact of the resulting levels.

6. Plants that offer no cover
7. Street lights
8. Foliage on the ground
9. Small furniture such as bistro chairs in front of some buildings

The list of elements ranging one-through-five had to be represented in my design tool prototypes, as these were the only elements with an impact on fundamental FPS mechanics. This was effected to keep the tool as simple and clear as possible, in order for my participants to easily understand it, while managing all important elements that have an impact on gameplay, so that any game design choices would remain in the hands of the designer while using the tool.

1. The terrain in the final levels was simplified and reduced to a flat playing field. Generally, terrain with small and large elevations provides cover by breaking line of sight between players. While this would have been an option for the prototype, in terms of game engine capability and design approaches found in a number of commercial FPS games, elevated terrain seemed to pose an obstacle to what this study seeks to understand. In order to present game designers with a quick-to-understand, simplified view of an entire game map, with tens or hundreds of corners and possible angles for line of sight and cover, a top down approach was chosen for the design tool. Elevations could in this way be represented in a top-down map by traditional map drawing means, such as elevation lines or colour gradients. Initial tests, however, showed that these forms of cover and line of sight were incredibly difficult to read when a designer had to evaluate a large number of candidates within the generation of a genetic algorithm run. Suddenly, a small set of elevation lines became hundreds of elevation lines, in addition to building corners and other obstacles occupying the game level for visual impact. This would have made for extremely noisy results in the human designers' evaluation, and would have impacted their selection of breeding candidates for the genetic algorithm. As a result, the terrain

was made flat. It is assumed that through obstacles such as buildings and containers (in this particular case), there remained sufficient opportunity for creating a balance between direct line of sight and cover situations for both playing teams. Buildings and shipping containers were easy to grasp and evaluate by the human designers, according to initial testing.

2. Streets are an element that can be considered purely cosmetic, as these elements do not provide any cover or obstacles to players. Parts of the terrain that are not occupied by buildings, containers, or street elements are still fully walkable. However, designers responded to streets for a number of reasons. Primarily, they were used as landmarks. In the case of this prototype, the street was a single path, essentially defining the overall shape of the playable map. Furthermore, players noticed that they could rely on the street being the centre part of the map holding the flag element, which they either needed to capture or defend (depending on which team they belonged to). Thus, even if players decided to 'exploit' pathways around large buildings to gain an advantage over the opposing team by flanking them, they ultimately knew that they needed to return to the street in order to proceed to their objective (defend or capture).
3. Buildings were simplified to a square in the design prototype and in the resulting maps for a single reason: they served as semantic elements for obstacles. Additionally, I also wanted no walkable buildings (with an interior) included, in order to focus gameplay at the street level, not the building level. The street level provided me with much easier evaluation, given that, similar to the aforementioned reasons for a simple terrain, designers would have to face the challenging task of evaluating hundreds or thousands of additional corners in each generation of candidates for the genetic algorithm. Removing this highly problematic task (by removing a large number of additional parameters for each candidate) allowed me to run a larger number of generations, without inducing user fatigue (Takagi & Iba, 2005).
4. The prototype levels only comprised a single game mode, i.e. a variant of the 'conquest mode', with only one flag to capture. One team has to occupy the vicinity of the flag pole to defend it, while the opposing team attempts to get close to it for a specified time of 20 seconds in order to capture it. The area, which either blocks the capture or initiates the capture, is roughly one street element, or one block wide.

Players see an indication of whether they are in the radius of the flag pole or not. The flag pole is in the middle of the street element, and approximately in the middle between both spawn locations, which are located on either end of the street path. The length of the path, as well as its shape varies, sometimes putting the flag pole close to both spawn areas (if the path is short), and sometimes far away from both (if the path is long and relatively straight). The flag can also be next to or even inside a container, which provides additional cover. These choices are fully driven by the DNA of each candidate for the genetic algorithm, and accordingly, initiated at random for the first generation, and potentially mutated for any candidate in subsequent generations.

5. Shipping containers provide small cover in street areas. Stylistically justified by the post-apocalyptic theme of the levels, where random obstacles clutter the streets of an urban area, they spawn primarily on street elements. As indicated in subsection four, they may also occupy the flag pole and the spawn areas of teams. My original testing with buildings showed that an oversimplification of the levels suggested that there was not sufficient cover, and the play experience could be less immersive. The addition of a smaller scale obstacle seems to have rectified this. Shipping containers are simply placeholders and can easily be replaced with Humvee military vehicles, as in *Insurgency*, or thin corrugated iron walls and concrete blocks, as in in *CS:GO*. Functionally, they serve the same purpose as in commercial FPS games, namely, to provide small scale cover to improve gameplay.

Given the cosmetic nature of elements six-to-nine, I will focus on how gameplay-defining elements one-through-five are represented in the level design prototype, as well as the final game level.

Figure 9 shows a top-down view inside the level design prototype. It represents a single candidate of a population of game levels. The large area that appears empty is simple terrain. The algorithm is able to use the grey area to place street elements, buildings, containers, the flag, and spawn points. However, based on the random first generation of candidates, which may have a relatively short path length, as well as the very close proximity of spawn points to one another and the flag pole, the entire play area may only occupy a small part of the entire terrain, as shown in this image. The green and red borders indicate the two team spawn

points. The flag pole is visualised by the round red circle in the middle of a street element. The algorithm always places the flag inside a street element in order to keep the flag in the focus area of the level, and it is surrounded by landmarks such as street markings, buildings, and containers. It is common in FPS games to place the flag in an area that does not have only open areas surrounding it, in order to avoid direct line of sight. Not providing any cover will lead to very poor gameplay, as opposing team members will be able to kill each other at a long distance. Evidence of this potential flaw in game level design – which has been avoided in this instance by keeping the main elements in close proximity – can be seen in older releases of *Battlefield*, *War Planes*, and similar, very open large-scale maps. This flawed design could potentially lead to both teams 'camping', rather than pursuing the intended goal of the game, i.e. to capture the flag.

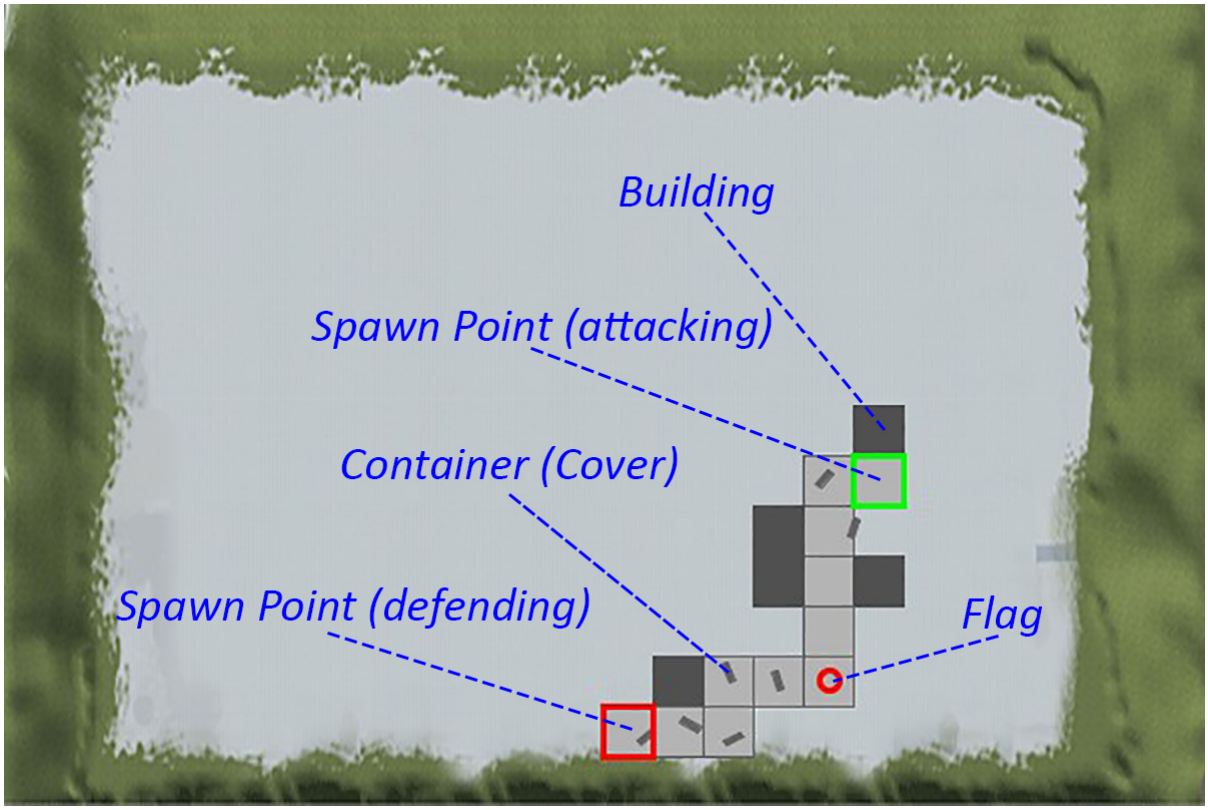


Figure 9 - Candidate level with annotations.

The dark grey squares in Figure 9 represent building placeholders. While the building assets in the final level are not exactly fully square, and do not always perfectly fill a full square, the difference between this simplified view and the resulting game level is marginal. It was deemed not worth the additional visual clutter that a perfect representation would have added to the game level design tool. I wanted to maintain an important goal, i.e. simplicity,

in order to allow the user fast decision-making, and to reduce fatigue as much as possible. If fatigue caused by visual clutter had impacted the results sought by this study, it may have distorted the findings concerning the software agent counteracting such fatigue.

Containers, being an element that provides additional, very effective cover due to their position in the middle of streets, and because of their odd angles with regard to the buildings and relatively linear position of the street elements – are shown as smaller grey rectangles (Figure 9). Figure 10 shows a rendering of the map example to illustrate how the map translates into an actual game level.

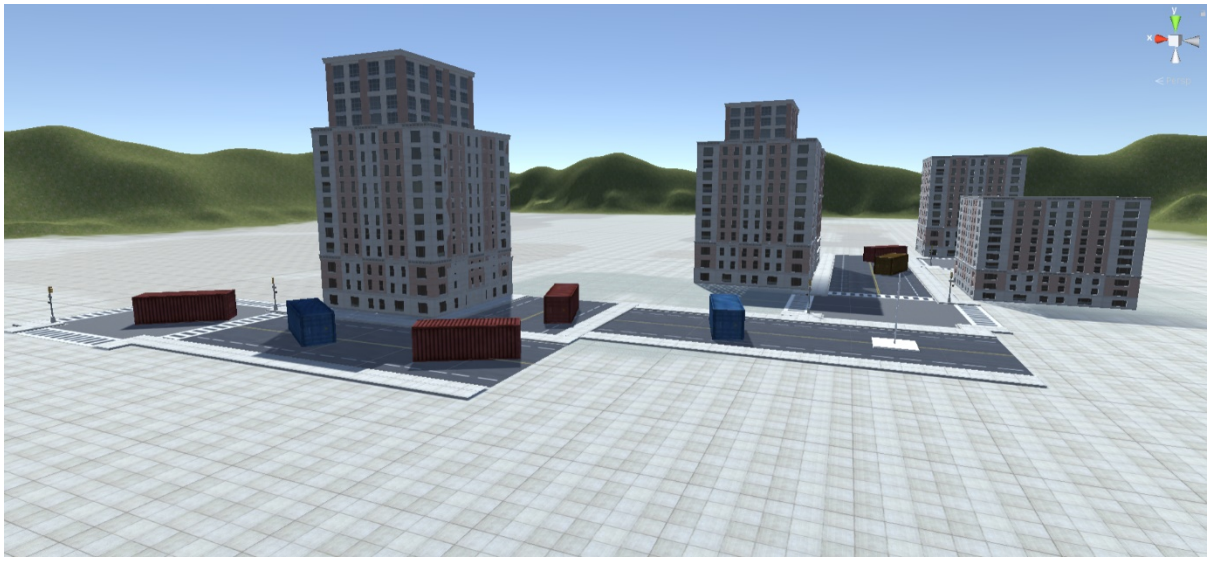


Figure 10 - Three-dimensional rendering of map shown in Figure 9

While the overall cluster of elements may appear reasonably small, it is important to understand that each cell representing a building or a street element is 25m squared. Accordingly, running at fast speed from end to end will take roughly one minute, which makes this example level in Figure 9 a similar size to typical medium-sized levels in *CS:GO* or *Insurgency*. This consideration is important, as the game mode is entirely based on infantry without vehicles, with a maximum of eight players in the map, which in turn can lead to very few encounters in a significantly larger map, based on my initial gameplay tests. Few encounters were considered less engaging and less motivating by test players. Therefore, the size presented in this average example also represents a size considered 'engaging' and 'fun' by participants.

4.1.2 Genetic Algorithm Implementation

The Genetic Algorithm used in this study employs value encoding instead of canonical binary encoding, to ensure that properties of map elements such as street, building, flag pole, spawn point and container on street are kept intact through crossover and mutation. The chromosome for each candidate of a population is implemented as an array list with individual genes as list elements. The genes are the street elements that form the path including all relevant information as shown in Figure 11. The chromosomes are initialised with random properties for each gene and a random path length between 3 and 10 elements.

4.1.2.1 Encoding

Given that the path length varies from candidate to candidate, the implementation of the value string as a list enables varying path length by adding further elements into the list. Each element represents a street section, starting with the red spawn point (see Figure 11).

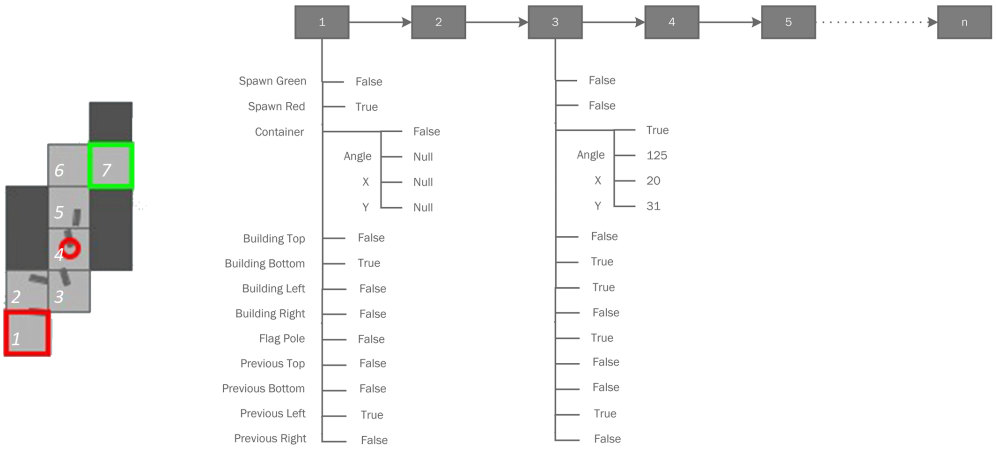


Figure 11 - Chromosome contains encoded values for each street element

The gene holds information about the direction of the neighbouring street elements, neighbouring buildings (if applicable), whether they contain a shipping container as obstacles and how the shipping container is positioned and rotated for each street element in the list. Further, information whether it is one of the two spawn points or the flag pole is stored.

4.1.2.2 Recombination

The crossover used in this study is a single point crossover chosen by a probability test against each path element of the first parent, which in turn is also randomly picked from the two parents submitted for recombination by the user. Owing to the variable path length, another random point is probabilistically determined for the second parent. Figure 12 illustrates how recombination is performed. Finally, based on the mutation rate, each of the chosen elements is altered, for example adjacent buildings added or removed, path direction changed (and subsequent elements moved into their new cells) or containers removed, added or simply rotated and repositioned.

In a final step, the path is re-centred on the terrain to avoid the path creeping out of the play areas through further recombination, and the flag pole re-centred to the middle of the entire path.

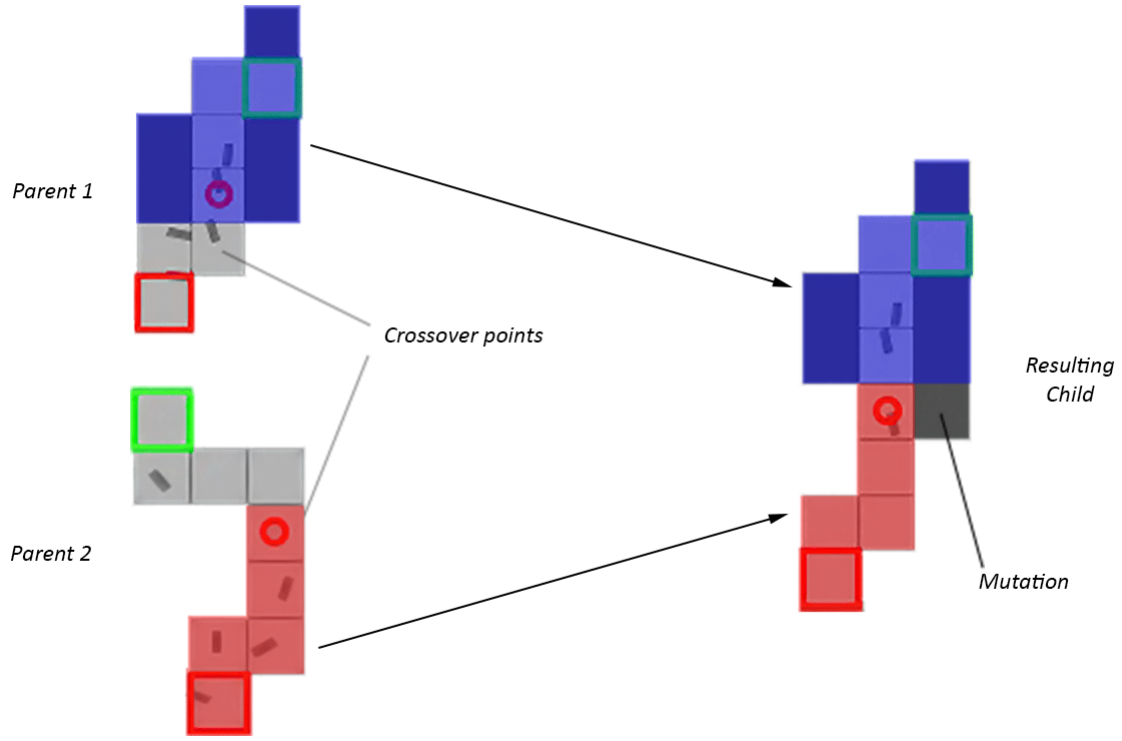


Figure 12 - Recombination including crossover and mutation

4.1.3 Level design prototype Interface

The level design in this research is concerned with procedural game level generation, which supports a human designer to create first-person shooter levels. The aim is to allow human designers to create a larger number of levels than would be possible by using a fully manual

level editor. The motivation behind this goal is to potentially reduce the cost of game asset (levels/maps) creation, reduce the time needed to create such assets, and to increase the variety of first-person shooter maps that a human designer is able to create within a certain timeframe. Cost of game asset creation has been identified as a source that can reduce the profitability of a computer game, which in turn cuts into the profit a game company is able to make (Koster, 2018). Time-to-release of a game or additional game maps are also factors that impact on a computer game company's ability to remain competitive (a shorter design cycle can lead to more frequent content releases). Reducing and simplifying game content creation can also help designers to create more iterations, which in turn may improve playability and ultimately, the success of a first-person shooter game.

This study uses a single human user (the designer) and multiple computational agents to control the underlying genetic algorithm of the design system. The human designer acts as the selection operator by selecting two parents within a population of 16 candidates. These candidates are presented as simplified top-down versions of a fully playable game map. An example of the user interface is shown in Figure 13.

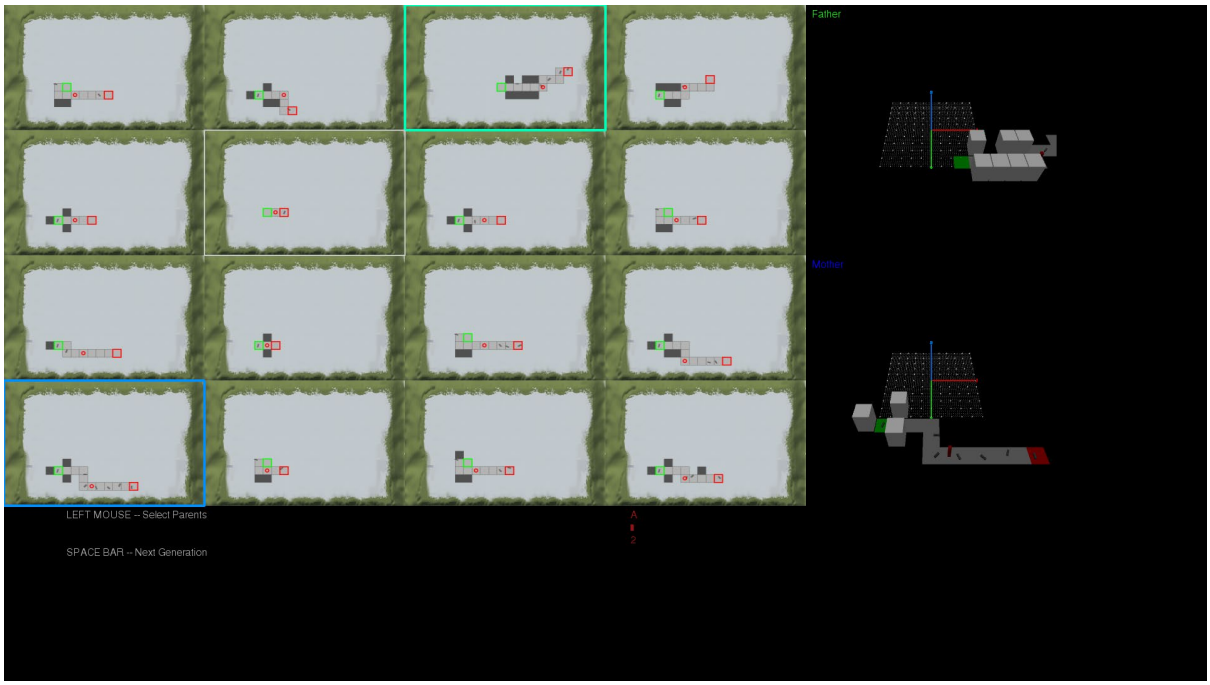


Figure 13 - GUI example for the prototype tool.

The 16 rectangular blocks in top-down view represent the 16 candidates. The highlighted green and blue candidates represent the parent solutions currently selected by the user. Both of these are also shown as a simplified pseudo-three-dimensional view on the right-hand side,

to provide a better view for judging the position of containers, buildings, and other elements. There are also simple debug outputs visible at the bottom, but these have little relevance to the user, and are only implemented for observation purposes by the researcher.

For the purpose of easy referencing of individual candidates, I will use the convention shown in Figure 14 in my discussion.

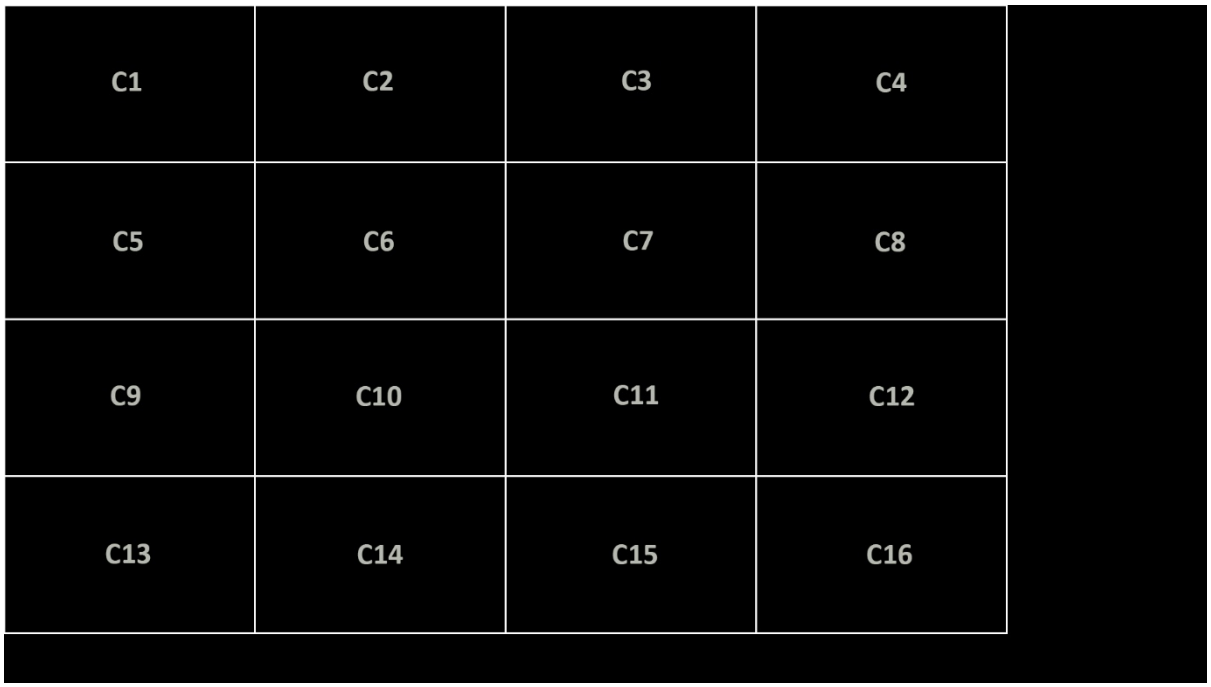


Figure 14 - Naming convention for individual candidate tiles.

As previously indicated, the two top left solutions (C1 and C2) are the parents from the previous generation (elitist strategy implementation), and the two top right solutions (C3 and C4) are the highest ranked candidates selected by the computational agents. A detailed description of how this selection and ranking is done will be provided in the following sub-sections on the different agents, and when discussing the multi-agent system in section 4.1.8.

After selecting these two parents (the selection can be changed at any time until breeding is initiated), the user simply has to press <SPACE> in order to start the breeding process. These are the only inputs required by the designer, which are also recorded as part of the telemetry stream. The user has the option of rotating the two pseudo-three-dimensional views in order to gain a better view of any blind spots.

4.1.4 Player experience goals

As the driving metric behind the player-centric design process, player experience goals (Fullerton, 2008) were central to the development of this study, the prototypes used, and the qualitative data collection effected by designers and play-testers. Prior to presenting a detailed discussion of the prototypes and the experimental design, an overview of player experience goals used in this research is needed.

It is important to understand that these goals are only partially influenced by my own work on the prototype. Ultimately, the designer has control over how these goals are defined, and consequently, how players will experience the resulting levels. This happens through the layout of the level, and all design decisions will have an impact on whether a particular goal gains more importance, or is less impactful, on the final result. Regardless, to enable the designer to set player experience goals, the prototype needs to have the option to do so. It is worth noting that these goals are not pre-set by the programmer of the tool, but rather, by the user of the multi-agent system. The programmer defines the set of possible mechanics that will help achieve player experience goals, but the multi-agent system defines which of these goals end up in the actual game level. This circumstance is important in my view, as it highlights that there is no pre-conceived nature embedded within the tools, allowing the designer to pursue their own goals; as such, the computational system augments the designer, rather than of replacing one of the most significant functions of their work. Implementation should offer as many potential goals as possible, enabling the designer and the agent system to select the specific goals relevant to them.

I will demonstrate my own thinking behind potential player experience goals, which can be construed as a result of the designer process. However, the designer is not bound by strictly following a scripted procedure and is still able to alter the goals quite significantly. As the results section will show, some designers made use of this freedom and subsequently derived unexpected results.

Free movement: players are free to roam the entire map without pursuing any match-attributing action. However, it is assumed that players will aim to work their way towards the flag and defend it by killing opponents.

Teamplay: while the players can pursue solo-play and explore the map on their own, and even ignore the necessity of killing opponents in order to ensure a team win, it is assumed

that players will seek to contribute to their team's efforts by providing cover fire, and actively looking to capture or defend the flag.

Cover: it is assumed that players will try to seek cover from enemy shots. Therefore, game elements in the form of obstacles will define an important part of the player's experience.

Map size: it is further assumed that players will (based on the above player experience goal) seek to work together to either capture or defend the flag. Map size plays into this goal, as the player needs to be able to see other team members and will try to cover any open ground with defending or cover fire. Therefore, a sufficient map size that is not too large or too small feeds into the players experience goals.

Balance: players are assumed to experience a sense of balance between a variety of game mechanics, which in turn will inform the overall play experience. For example, players will presumably seek to have similar opportunities to defend or attack a flag, no matter which side they are on. A bias for one team, through issues such as an imbalance in cover and line of sight, proximity to the flag, team size, or the skills of players, will have an impact on this player experience goal. Most of these factors are controlled by the designer, and only a few are subject to different mechanisms; for example, player skills will have to be balanced through other means such as a point-based match system, which is currently not part of the prototype design tool.

In conclusion, it is apparent that covering all possible design goals and implementing them as an 'offer' to the designer and the multi-agent system will be difficult. Nonetheless, I believe that the important mechanics have been considered, and that many different player experience goals can be achieved with the current prototype. The results chapter discusses whether design experts agreed on this matter, and how they may wish to see this aspect improved.

4.1.5 Design Expert Agent

The design expert agent (DEA) is the core of the computational agent system. It models a (human) game level designer, as suggested by Zhu et al. (2017), and applies high-level, abstract concepts to FPS maps in order to evaluate their quality. This agent represents a main contribution of this thesis and implements a novel approach to procedural game content

evaluation. Concurrently, the agent also serves as a low-level utility agent providing core functions to the system as a whole. It runs independently in the system, utilises digital differential analysis (Museth, 2014) to find intersections with solid obstacles from its current position (which is freely adjustable within the boundaries of the entire map), tests metrics pertaining to the relationship between both spawns and the flag pole, and assesses the entire breeding pool in preparation for candidate selection. Furthermore, it actively selects the two candidate solutions with the highest ranking from the breeding pool and feeds them into the 16 candidates of the next population.

To provide detailed insight into how this agent works, I will first describe the heuristics that have been developed from designers' accounts of their game level design process. Then, in the following subsection, I will provide a few examples of the analysis that the agent performs, enabling heuristics to be applied for a ranking of game levels, so that two candidates can be suggested to the user for each population of candidates.

4.1.5.1 Heuristics

Zhu et al. (2017) used a large database of online game reviews to develop playability heuristics. Taking a quantitative approach, they conducted a full lexical analysis of nouns and adjectives (Zhu et al., 2017). When I initially set out to obtain data from game designers, I realised that direct access to participants as a means for collecting data, and for gaining an understanding of their workflow and processes, could be impractical. I therefore considered the approach taken by Zhu et al. (2017) a viable option for collecting my data. However, there are not nearly as many online sources and print media that explain the actual methods game level designers use. Additionally, Zhu et al. effectively repurposed a source that had a different goal (an online game review that assesses games from a consumer perspective) and used it to interpret the playability of games. They highlight that this approach is naturally an interpretation, and prone to some inaccuracies. I felt that using designers' accounts of their processes directly would potentially remove some of these inaccuracies; however, my approach still depended on the quality of the designers' own assessment and accuracy of their description of their methods. I am also aware that a qualitative approach requires very careful consideration of biases that I may hold myself, in addition to the possible inaccurate reporting of sources. To mitigate these implications, I decided to conduct a full qualitative analysis of all sources using NVivo, where I could identify topical parallels between different sources

quite easily in order to filter out a number of stray remarks that may or may not have any actual impact on the game level design process. This also reduced the number of heuristics to a small number, which in turn provided me with an extremely focused approach for my agent design. Overall, my methods for developing designer heuristics is closer to the work of Pinelle et al. (2008), who took a qualitative approach to find usability heuristics for video games. These authors highlight the lack of formal evaluation methods and suggest a simple three-step workflow. First, the problem identified by the source needs to be captured; then, these problems need to be categorised and grouped. Finally, based on the resulting (few) categories, actual heuristics can be developed.

Contrasting the approach taken by Nacke et al. (2009), who employed a combination of biometrics (EMG) and psychophysiological measures, plus a number of simple in-game metrics, the present study uses heuristics based on expert publications (books, online blogs and journal articles authored by game designers), a selection of in-game metrics, and eye tracking.

The heuristics that were developed for the DEA can be summarised as follows:

H1. Balance of cover and line-of-sight

FPS level design experts identified a balance between sufficient cover and some areas that provide direct line of sight between competing teams as one of the crucial key elements that drives gameplay. There must be enough cover so that opponents are able to escape shots taken from a long distance. Most players do not have the skill to take long shots, and as such, the opportunity to move towards the goal of the level (in this case, the flag pole capture area) is increased. Otherwise teams may fall into a lockdown with neither moving forward, known as ‘camping’ (Wright, Boria & Breidenbach, 2002), which leads to the next heuristic.

H2. Remove camping hotspots

Hotspots (also called ‘choke-points’ by some experts) can be enjoyable areas where a balance between cover and line of sight is present. Camping hotspots however, are sections of a game level where players exploit the hotspot to gain kills, rather than play to win the match (Wright et al., 2002). Removing these camping hotspots has been identified as a key element for immersive gameplay. Expert designers also

pointed to better playability on several occasions as a result of these camping areas being removed.

H3. Distance to first cover from spawn points

The third heuristic is again based on areas of the map that provide cover to players. When the match starts, both teams presumably rush towards the capture point, but given sufficient map knowledge, the teams also know where first contact with the other team is likely to occur. Therefore, in competitive matches, teams generally only rush toward those first cover points, and take a more strategic play approach to avoid getting killed too quickly. The distance to the first cover point (or in some cases, the time it takes a player to run to the first cover point, depending on the run speed implemented by the developer) is another key heuristic for good, playable levels.

H4. Distance between spawn points

Linked to the above, the overall distance between both spawn areas defines the time players need to make first contact. However, given that there is generally more than one possible pathway that can be taken towards the capture point, for example, navigating left or right around a larger obstacle such as a building (or shipping container), the point for first clashes, as highlighted in (3) varies, depending on the path taken by the individual player. Experts identify the overall distance between spawns as another important metric.

H5. Distance from spawn points to capture point (flag pole)

Finally, the distance between each spawn point and the capture point is important, assuming that they are not exactly equal in distance, because then the flag pole/ capture point would essentially be the first point of contact between teams anyway. Measuring the distance between spawn and capture point for each team across different possible pathways allows for making a statement about balance between spawns. If one team gains a significant advantage because their spawn is next to the capture point, the level may not gain significant popularity, as it may be considered playable, but not very fair. This is particularly true if the spawn of the attacking team is significantly closer to the flag than the defending team, in which case the attackers will likely not face too much resistance before reaching the capture point. Expert designers consider a near equal distance across multiple pathways as ideal, and this was implemented as another heuristic in the designer expert agent.

The abovementioned heuristics were implemented in the current designer expert agent (DEA). Several possible pathways were considered in the final implementation using A* pathfinding, and multiple potential choke points were measured using digital differential analysis. A few additional heuristics that were extracted from expert accounts have, however, not yet been implemented, as they do not fit the minimum-viable product approach of this study. For the purpose of completeness, and also to provide an indication of where future level design prototypes can be taken, I wish to provide a brief discussion of these heuristics in the following paragraphs.

H6. Shape of main corridors

FPS levels that have more than one main route between spawn points and the capture point often follow either an H-shaped or a double T-shaped layout. This is considered successful, and encourages strategic gameplay in some of the most acclaimed FPS maps, such as 'Dust2' and 'Sienna' in *CS:GO*, but requires careful consideration of choke points and sufficient cover (see above) to avoid both teams getting stuck, or simply camping for points, rather than a win.

H7. Elevation

Game levels with not only a flat terrain, such as my simplified prototypes, but elevations and perhaps multi-storey buildings, must take the same heuristics into account, but apply them in a three-dimensional fashion. This is highlighted as an explicit goal by some experts, for example, 'Dust2' has a bridge and recessed corridor (the H-shape) that require additional measure to avoid players camping on high ground. To illustrate the problem, if a player has a wall as cover, and is above another player, only their head and gun will be visible. In contrast, the player who has no high ground is more exposed due to the angle of the attacker. The head and upper body, or even the full player character may be visible, which makes attacking and hitting the player much easier. Therefore, if one team has better access to high ground due to the map layout and the time it takes to reach certain strategic points, the other team must be given additional obstacles to ensure map balance.

H8. Orientation through landmarks

Higher complexity, such as three-dimensional gameplay provided by building interiors, elevation, changing terrain and similar factors, also necessitates additional

orientation for players, for example, through landmarks. This is not only an optional feature that makes the level easier to understand, but also provides a foundation for effective communication between players. This is very important in competitive FPS team-play.

4.1.5.2 Digital differential analysis implementation

The designer expert agent needs the ability to independently assess each candidate based on the heuristics that have been developed. It relies heavily on digital differential analysis, which is an efficient and fast way to traverse grid cells between two points, to find any intersections with obstacles. It essentially provides a simple vision for the designer expert agent, enabling it to detect open areas that provide direct line of sight between obstacles, and of course, a detection mechanism for the obstacles from player positions, which is similar to having eyes, or radar-like functionality. In a game engine, this is generally provided in the form of ray casting, which involves sending out a ray from a defined position, which subsequently returns intersections with colliders in the game environment. It is often computed on the GPU and is a highly efficient approach for giving NPCs a simple form of vision. Given that the prototype is not running inside a game engine, and owing to the lack of any ray casting provided by Java and the libraries I employed, I decided to implement a fast and simple version of ray casting myself that feeds into the digital differential analysis. My implementation allows for a two-dimensional analysis, with the capacity to be expanded into three dimensions if needed at a later stage, for example, for levels that include the interior of buildings, or game environments that are located in mountainous areas, or any other type of elevation.

The agent is able to move freely within the simplified environment of the candidate, and casts rays from different positions to observe where a player will see obstacles from. It records the first collision, and given that players cannot see through obstacles, this is sufficient, rather than recording every intersection with all obstacles along that particular ray. The same process is effected from the perspective of the other team. The agent starts inside the spawn areas and gradually works towards the flag pole using path planning. This is of course a simplified approach, as it ignores positions that are, for example, off the main pathways towards the target (the flag pole). In a game, some players may choose to veer off the main pathway for strategic positioning. I am planning to expand the agent's capabilities in future,

including adding different player traits such as defensive play, aggressive offensive play, strategic play, and others. This will impact on the agent's behaviour and add a broader scope for exploration of potential positions off the main pathway. As previously noted, some simplifications are the result of reviews and discussions with experts. I also feel that adding every possible option to an early prototype may convolute the output and make it more difficult to arrive at rigorous conclusions. Different player traits can be difficult to interpret from data, and are subject to significant noise, particularly when players opt to 'free-play'. Free-play commences when players decide to create their own goals, rather than follow the gameplay intended by the game designer. In this case, players virtually create their own mini-game within the game (Wright et al., 2002). Accordingly, players do not follow the immediate goals of the game such as capture the flag. Therefore, this prototype assumes that players will indeed follow the main goals, and disregards players engaging in contradictory behaviour.

Without jumping to conclusions, I believe the DDA has been proven a suitable approach for enabling an agent to assess game levels through simple ray casting and collision detection. It is an effective algorithm that offers the speed needed in a Java prototype, given that we analyse the entire mating pool, with large numbers of candidates, which in turn hold a number of objects and require analysis for many different points within the map, e.g. spawn points, flag pole, and every container and building that serves as an obstacle. If a design tool is implemented as a plugin of a game engine that already offers ray casting and collision detection, DDA can be replaced by tools native to the engine, as they will likely be highly optimised and potentially much faster. For an academic prototype, however, the DDA is a viable and reasonably simple solution.

4.1.5.3 Fitness assessment

The previous two subsections of the DEA discussed heuristics and digital differential analysis implementations. This section illustrates how the resulting fitness was numerically assessed in order to rank the candidates in the mating pool. A quick word about the difference between this approach and a canonical fitness function may be appropriate here. The canonical fitness function is a mathematical function that assesses a current population and is the core of the selection operator. The fitness calculation of the designer expert agent is used to select a candidate for the population after recombination and mutation operators have been applied to the mating pool. Its fitness does not guarantee that it will be part of the next mating pool

by default, and as such, it does not act as a selection operator. The high fitness here simply means that it will be presented to the human designer. The human designer will ultimately decide whether this highly fit candidate will make it back into the next mating pool. In this way, the human designer stays in full control of the selection process, and the designer expert agent merely makes a recommendation.

The fitness calculation performed by this agent takes the following measurements, initially performed by digital differential analysis:

M1. From green spawn point to flag, number of obstacles.

M2. From red spawn point to flag, number of obstacles.

M3. From green spawn point to red spawn point, the number of obstacles (cover) from the current ray casting point.

M4. Distance from green spawn point to first obstacle (cover).

M5. Distance from red spawn point to first obstacle.

M6. Distance from flag to first obstacle, casting towards red spawn point.

M7. Distance from flag to first obstacle, casting towards green spawn point.

Plus, the distance between:

M8. Both spawn points

M9. Red spawn to flag

M10. Green spawn to flag

The calculation of the number of obstacles between flag and both spawn points is based on the notion that a large number of obstacles will provide sufficient cover, whereas a low number of obstacles create open spaces. The first heuristic (H1) states that a balance of cover and open space is important for effecting engaging gameplay. The agent divides the number of obstacles between flag and spawn by the distance between flag and spawn. Therefore, if the level is larger than average, a larger number of obstacles is required to gain a high score for balance between cover and line-of-sight. If, on the other hand, the level is very small, a small number of obstacles is sufficient for providing balance that leads to a high fitness score.

H1 is implemented as:

$$H1 = \frac{\left(\frac{M1}{M10}\right) + \left(\frac{M2}{M9}\right)}{2}$$

Calculation of distances between certain points of interest such as the flag pole and nearby obstacles, as well as number of obstacles between spawns and the overall distance, is based on the notion that this distance represents open space, without cover to remove camping hotspots (H2), and to protect the spawn for each team. A larger distance means a lower score, and a smaller distance provides a higher score.

H2 is there calculated as follows:

$$H2 = \frac{\left(\frac{1}{M6}\right) + \left(\frac{1}{M7}\right) + \left(\frac{M3}{M8}\right)}{3}$$

The normalised measurements of distance between spawns and first cover (M4 and M5), distance between spawns (M8), and finally, distance from spawns to flag (M9 and M10), all directly address heuristics H3 to H5.

Finally, the agent calculates an average between these measurements, which results in a score between 0.0 and 1.0 as each of the heuristics calculations has already been normalised. This average represents the unweighted fitness score and is calculated as:

$$unweighted\ fitness\ score = \frac{H1 + H2 + H3 + H4 + H5}{5}$$

I decided against applying weights to the individual scores, simply because there is no indication in my data sources that any of the aforementioned heuristics take precedence over another. I believe, however, that there may be merit in conducting a study among level design experts to obtain a weighting of heuristics, if the experts agree on a ranking of importance. This may bias the fitness score of the agent towards certain factors, for example, cover and open space around the flag pole may be more significant for engaging gameplay than cover and open space around the spawn points. For now, I do not believe that this bias should be introduced based on a hunch, rather than reliable data. An alternative to weighting may be multi-objective optimisation, using each of the heuristics to remove solutions that are not close to the pareto-optimal boundary. Both of these ideas can be explored in future work.

4.1.6 Diversity Agent

This agent has some similarities to the DEA, in that it acts on the DEA's knowledge, but applies it in a different way. It seeks to identify levels that are not represented in the population shown to the human agent, but nonetheless have a high degree of fitness, while being very

different to the solutions proposed by the DEA. For example, if the DEA selects levels that are predominantly horizontal, the DA will seek out levels that are vertical, while maintaining a high level of fitness. However, fitness is the parameter it compromises on first – the most important goal for the DA is to identify levels that are different, levels that provide some degree of diversity to the population. The main idea here is to avoid the entire multi-agent system from getting stuck in a local maximum, that is, the state where the user can only select from very similar levels and therefore, cannot get out of the local maximum themselves.

To avoid creating an agent that simply selects the weakest candidates from the breeding pool, the diversity agent seeks to find a candidate where only one property is significantly different from the highest ranked solution according to the designer expert agent. For example, path length may be very different, or one team may have much less cover than in the strongest candidate. All other parameters are kept at a high ranking. The diversity agent selects the one trait that it seeks to vary in its selected candidate, based on a random decision from generation to generation. It is not the parameter that is ranked the highest by the designer expert agent that drives this decision by default. The reasoning for this is to avoid including only the most obvious counterpart of the highest ranked candidates, and instead create actual diversity.

This agent is the reason why both agent-suggestions in C3 and C4 are always somewhat different, but highly-ranked candidates. Without the diversity agent, C3 and C4 will often be identical, or virtually identical. With this agent, both suggestions look similar but have one distinguishing feature. The reasoning for this is to avoid two (virtually) identical candidates among 16, as this will result in a significant proportion of each population not offering any variety. To a degree, the diversity agent can be considered a high-level mutation operator that is based on heuristic diversity. While it does not actually change the DNA of any candidate, it ensures that the new population includes high-performing, but diverse candidates.

One reason why it is difficult to claim that the designer expert agent always selects the highest fitness, with the diversity agent taking second place, is the fact that both candidates may have the same numerical score. While this may seem counterintuitive, it can be easily explained. Both candidates may be different in that, for example, the path is horizontal in one case and vertical in the other, and the numerical score may be identical, as this does not matter to the

DEA. Thus, within the mating pool, there may be a number of candidates that score the highest rank in that particular pool. This will be correct if one only considers the particular snapshot of that particular generation. There is no difference if both levels with a horizontal and vertical level arrangement are played by players, assuming that all other metrics such as cover, line of sight, place of spawn, and so forth are identical. Where it does make a significant difference, however, is when the human user combines one candidate with a different parent. Here, the arrangement of streets suddenly makes a significant difference, as it may lead to very different children when two street paths are combined. Therefore, the diversity agent makes a difference to the overall process, while appearing seemingly insignificant when only judged by a sometimes identical and often very similar numerical score compared to the Design Expert Agent.

4.1.7 User Preference Agent

The user preference agent was an aspect developed during my Master of Philosophy and was implemented and tested prior to embarking on this degree. I decided not to use it for my evaluation and will provide a reason for this. However, I believe that it needs to be included here, as it is a logical addition to the multi-agent system in future.

The user preference agent uses a J48 algorithm, which is an open-source implementation of the C4.5 decision tree algorithm. It aims to detect user preference by considering user input using the mouse and keyboard, which is used as a training set for the classifier over a number of generations. Then, in order to counteract user fatigue, a typical issue with interactive genetic algorithms, the classifier deduces user input, based on the training it received. A number of generations are used to compare the prediction of the model and the actual input for further training of the classifier. Eventually, the computational approach is used over the actual user input at a high ratio, for example, 10 computational generations and then one human selection. This allows running the interactive genetic algorithm for significantly more generations, which in turn helps the system to converge towards a solution. At the same time, the user is not affected by fatigue as much as when using a traditional interactive evolutionary algorithm, because the user preference agent runs a significant number of generations once it has been trained by initial user selections. My study found an increase in performance of roughly 15%, representing significant time-saving for the human user. Even a slight reduction of production time in a game level design department, will allow for either more levels to be

made for a particular game, or simply effect the better utilisation of resources with the help of computational tools.

There are two reasons why I decided to leave the user preference agent switched off in the present study. First and foremost, the agent had been developed as part of a previous project and lacks novelty in its current form. Therefore, I believe that it will not add significantly to this thesis in terms of novelty and scholarly contribution. Second, the original implementation uses a J48 classifier, as noted above, which is prone to noise when used in small datasets. Finding a classifier that is better suited to this particular use, where few samples can be used for training, and testing measures to counteract shortcomings (Finlay, Connor & Pears, 2011) without removing J48 altogether are steps that will be taken in future updates of the multi-agent system. These measures may include feature selection and pruning the decision tree.

4.1.8 Multi-Agent System

The level design in this study is loosely based on Kosorukoff's (2001) human-based genetic algorithm framework, which allows a human user to act as an integral part of the genetic algorithm. The intent is to augment the abilities of level designers and allow designers to explore different options (diversity), as well as additional options, through a higher number of iterations. If the user is simply employed to control a genetic algorithm by selecting the candidates for breading, effectively becoming the selection operator of the genetic algorithm, and if there were no additional human or computational agents, the resulting system will simply be an interactive genetic algorithm (Takagi, 1998). However, the approach proposed by Kosorukoff (2001), and adopted by the current study, develops this concept further. The human user (or users) can take on different roles within the system, and effectively act as different operators of the genetic algorithm. Additionally, one or multiple computational agents will take on additional roles and control one or many operators.

The aforementioned agents are responsible for level analysis, creating diversity among the potential candidate solutions in each population of the genetic algorithm, and applying design knowledge extracted from human expert statements about their process. These three agents work in conjunction with a human level designer to create a playable FPS game. This section describes the structure that ties these individual components together. It is founded on human-based genetic algorithms, a framework that allows human and computational agents to become either selection, crossover, or mutation operators (Kosorukoff, 2001). My multi-

agent system uses a variant of this framework, where the human agent is strictly responsible for selection, and where multiple computational agents share the tasks of recombination and mutation (Figure 16).

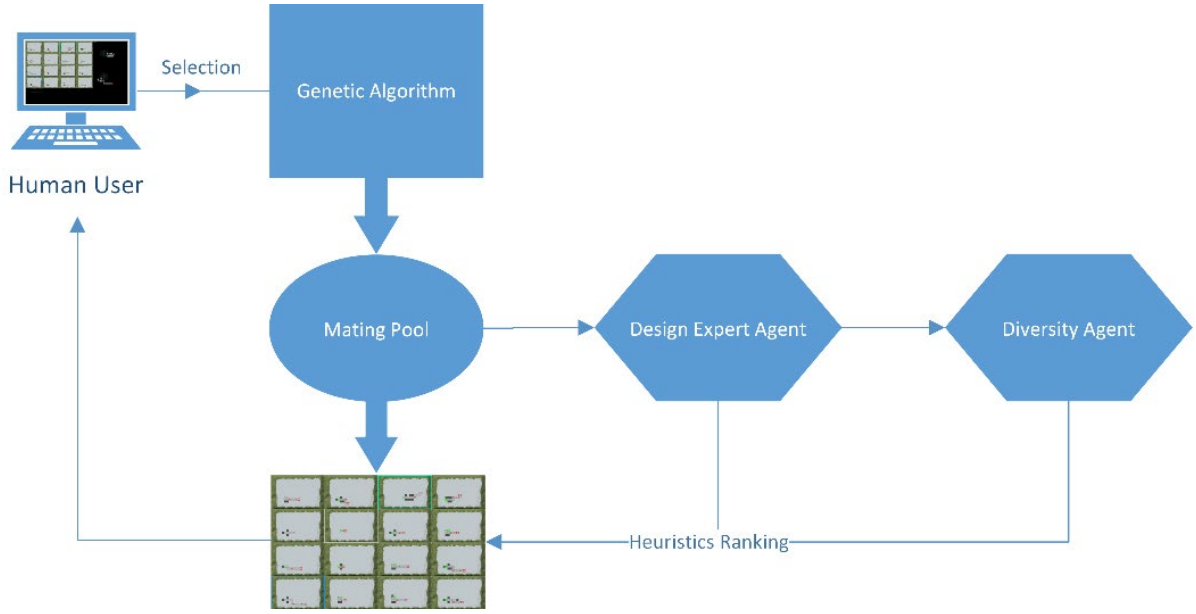


Figure 15 - Schematic overview of the multi-agent system.

Therefore, instead of providing a description of the multi-agent system on one hand, and the genetic algorithm on the other, I am presenting the entire system in this section. This is also owed to the heavily interlinked nature of the computational agents, the human designer, and the operators of the genetic algorithm.

The system has no mathematical fitness function that selects candidates for the next mating pool (as in the case of a canonical genetic algorithm), but uses the output of several agents to evaluate fitness of the candidate pool and individual candidates (see Figure 17). It actively selects two candidates based on metrics learned from the documented processes of expert designers.

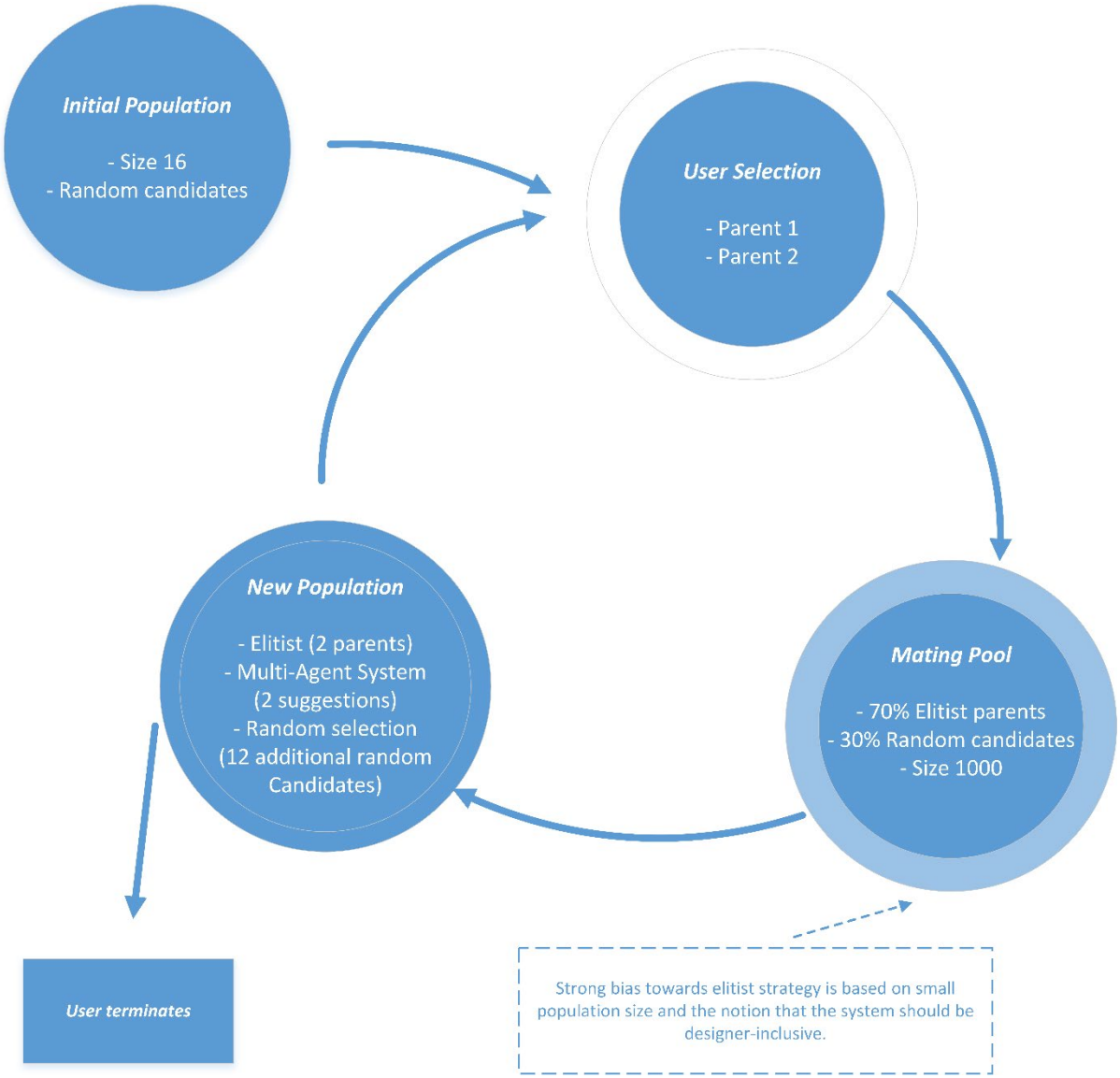


Figure 16 - Overview of interactive evolutionary process employed in this study.

It is ultimately the designer who makes the selection, regardless of whether the agents deem the user selection the fittest option. While this will potentially lead to the selection of candidates that are not the fittest by numerical measures, it provides the designer with some flexibility. Rather than being locked in by metrics, the designer is able to change course for the final result by intentionally selecting any solution, regardless of its fitness. My particular test case is domain-specific and has not yet been generalised to domains other than FPS game level designs. However, I believe that a similar system can be adapted beyond FPS levels, assuming that sufficient data for extracting heuristics for the designer expert agent is available.

The genetic algorithm of this system employs an elitist strategy, which means that both parents, selected by the human agent (the game level designer), are put straight back into the next population without any adaptation. The main reasoning for employing an elitist strategy is that any subsequent population should not be less fit than the previous one. Given that the designer may not have chosen the numerically fittest candidates, but perhaps any of the other candidate solutions for breeding the new population, this may not hold true. However, it ensures that the human designer views their selection from the previous population in the current selection, in case the genetic algorithm did not recombine these two candidates into any acceptable new solutions. This measure was taken after initial tests showed that participants were sometimes confused by not seeing any solutions that were (subjectively) at least 'as good' as the previous selection they made. Presenting the user with exactly the same two solutions that were selected in the previous run, alongside 14 new candidates based on recombination, and including mutation, removed this perceived issue with the system.

In summary, any bred population of 16 candidates consisted of the two parents selected by the designer, two candidates selected from the breeding pool by computational agents, which ranked the entire breeding pool by fitness, and 12 candidates randomly selected from the breeding pool.

4.1.9 Eye tracking and heatmap generation

During the evaluation trials for my prototype design tool, each participant worked through a series of game level generations. For each run, I collected the raw position data streamed from the eye tracking SDK. This stream contained continuous x and y positions in screen-space at a rate of 30 Hz. The eye-tracker provides full pre-processing, so that the stream can be saved directly to the computer hard disk. This pre-processing includes the triangulation necessary to convert gaze data from both eyes into a target position on screen – essentially, the point that the participant looked at. Unfortunately, I had no access to a commercial research-grade eye tracking system, which provides extremely high accuracy, and had to fall back on using a relatively simple device made for playing games, a TOBII 4C.

I conducted initial tests to establish the eye tracking accuracy and found that I could reproduce results within a small ellipse of 30 pixels, with high accuracy and high precision. 'Accuracy' here is defined as the ability to locate an area on the screen using eye tracking.

'Precision' is defined as the ability to reproduce these results. I tested this with a simple colour bubble on screen and a number of target points for positioning this ellipse at. Given that I was interested in a fairly large area (the tiles that include the game level candidates are 320 x 200 pixels each), the accuracy was sufficient by a large margin.

For the evaluation series with my participants, I streamed the data into a CSV (comma separated values) file. The collection of eye tracking was synchronised to the main module of the genetic algorithm via OSC, a simple network protocol. The OSC module was expanded to include a simple check as to whether a data pair had actually been received, and that it did include a number. If no number was present, a NAN (not a number) was transmitted. I also implemented a simple visual cue (a small, dark red rectangle) that was barely noticeable and easily ignored by the participant. This simply served my observations in a manner that ensured tracking generated valid data, without creating any distraction for the user.

The heatmaps were generated by an automated process in a Java-based software module. The CSV files were read as x and y position pairs. For each position pair, a 50 pixel-wide circle was created surrounding the centre of the gaze data. This was written into a framebuffer. Once all pairs were processed, the density of all circles at each position of the framebuffer was established. Then, the minimum (zero) density to the maximum density read from the framebuffer was mapped into a green to red colour ramp to represent a change in density. Density represents the intensity at which the user stared at particular areas of the screen. While this approach may seem somewhat complicated at first glance, it is based on the need to process very different numbers of position pairs, as the length of each trial was defined by how long the user took to perform their chosen number of runs, not by a fixed timeframe or time limit. This means that however many seconds a participant looked around the screen, I would receive 30 position pairs for each second. Using a framebuffer, I was able to generate consistent and therefore comparable heatmaps. Otherwise, the length of each trial run would have influenced the density of the heatmap, and therefore, potentially distorted visual results. Additionally, very short runs, where the designer finalised the decision after only a few generations, would barely have been visible at all. At the same time, the runtime of each run was still visible by looking at how abundant or sparse bubbles were. Figure 18 shows an example of a final heatmap with a long run and high density in some areas. Red represents an area of high intensity, and green an area of low intensity. Simply stated, the red 'heated'

areas of the image highlight an area the user looked at for a long time, compared to darker and green areas. Figure 19 is an example of a heatmap of a very short run, where the designer decided to call the result finished after only a few candidate generations. Through the aforementioned normalisation, areas of higher intensity are still readable, but the sparse distribution of colour blobs clearly indicate significantly fewer measurements, which in turn means a much shorter overall runtime.

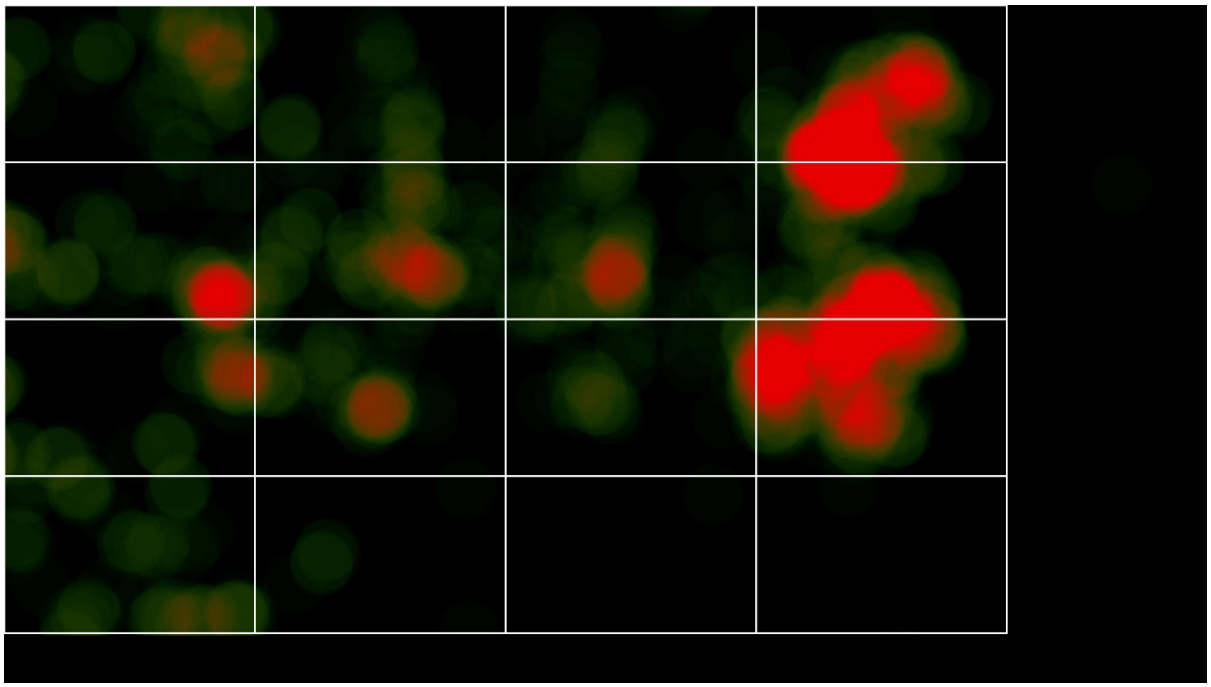


Figure 17 - Example of a heatmap with long runtime.

Heatmaps in other projects often show gaze data as rather large blobs, which frequently combine into one big coloured bubble. I kept gaze circles fairly small to preserve as much detail in my heatmaps as possible, given that I was not interested in a rough tendency of what the user looked at, but interested in being able to distinguish between the 16 candidate tiles on screen. As a result, individual circles showing gaze position are better observable and appear as colour-bubbles. The white grid indicates the area of the 16 candidate tiles.

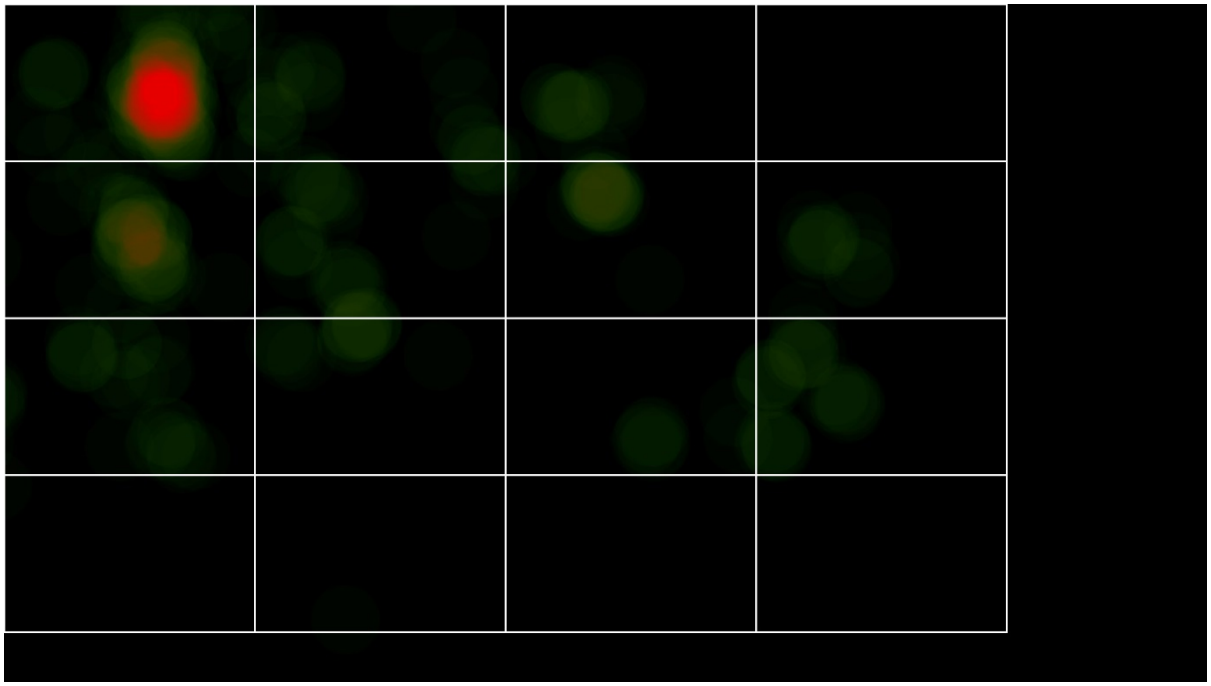


Figure 18 - Example heatmap with a short runtime.

This section introduced the research prototype and more importantly, underlying heuristics in detail. The following chapter offers a discussion of the results of the design prototype evaluation.

5 Results and Observations

This chapter discusses the prototype evaluation with professional game designers. While the methodology chapter addressed the reasons for collecting the data that has been selected to be the focus of this thesis, similar questions need to be asked regarding the analysis and discussion of the collected data. What is the perspective and lens that I am applying to my analysis? What do I hope to learn from the data? These questions can be linked back to why and how I chose to collect data in the first place, which has only been partially answered within a broad overview in the previous two chapters.

My primary objective with this thesis was to develop an understanding of whether the multi-agent system in the form of a genetic algorithm, using two computational agents plus a human agent, is able to perform complex design tasks in such a way that it augments the human designer's abilities, and capture the designer's intent, without interfering with their ideas, and not limiting them to a predetermined choice, for example, through a classical fitness function. A fitness function will be limiting in that it only seeks to optimise a small number of parameters. I wanted the design to be able to pursue complex tasks, and to have a strong influence on the direction the system takes, while concurrently saving time and effort. Furthermore, I wanted the system to provide active assistance when design choices had to be made, and to provide suggestions to the designer that were not necessarily simple optimisations, but which also provided variance and diversity, while also considering best practice level design principles. Accordingly, the agents provide a fitness ranking and they are also not directly acting in the selection process, so that they are not considered a predefined fitness function. To verify whether the system delivered in this regard, I wanted to tap into the participant's game design experience. Accordingly, this is what I attempted to extract in the post-participation semi-structured interviews. The focus of this chapter is therefore designers' response to working with the multi-agent system: did they believe it added to their experience, or did it perhaps get in their way; did it help them to explore more ideas in the same or a shorter period of time; was it not of much use to them, and did they prefer to manually design the level layouts instead.

The issue at hand is that most designers will need a lengthy explanation of what the system does and how it works in order to provide direct responses to these questions. The designer will need to understand how it works to provide feedback, which will subsequently help to

improve the system's tools. Accordingly, my aim was set on gauging designers' experience, rather than engaging in direct (software) design feedback. This allowed me to engage with any game designer, as opposed to only those who also understand the underlying mechanics of the system, which would be required for direct feedback.

I also wanted to use the design tests to create a baseline for my multi-agent system as well, simply because access to experienced game designers is limited. Consequently, I had to run a number of tests with and without the MAS active, while the game designers performed their level layout design task. To avoid a potentially loaded question of whether they liked the system when it was active, and disliked it when it was not, and to reduce bias arising from participant perspectives for or against AI-driven systems altogether, I decided to use the aforementioned interview questions, without telling the designers when the system was active, and when they were in fact simply controlling the genetic algorithm, without any agent support, essentially using a simple interactive genetic algorithm, rather than a multi-agent system driving the genetic algorithm.

Despite my original intent to simply test the hybrid MAS against a baseline, without the game designer's knowledge, the observations and interview responses revealed some interesting insights into how game designers think of their own design abilities, particularly when they have worked in a professional capacity for a long time. I will discuss this in the following sections of this chapter.

5.1 Evaluation of Game Level Design Prototype

This section presents the results of the level generation tests with professional game designers, and highlights observations recorded during the tests and follow-up interviews.

Data was collected in a number of ways, specifically, using a pre-test demographic questionnaire and the think-out-loud method, in addition to other observational data gathered during the tests, software telemetry data (keyboard and mouse inputs, ranking of candidate solutions) streamed at the time of the tests, eye tracking data derived from the Tobii 4C eye tracking device, and finally, responses to semi-structured interviews, after the tests had been completed. Accordingly, a number of quantitative and qualitative sources need to be examined and discussed; therefore, the data were segmented into two main groups, as shown in Figure 20.

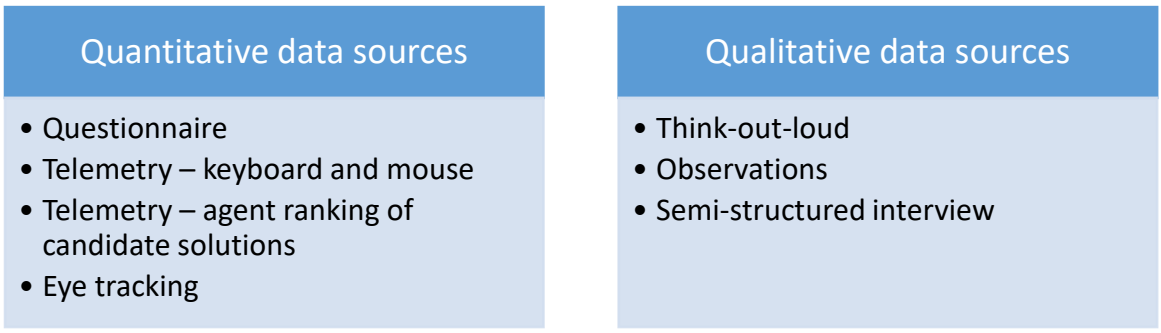


Figure 19 – Data source categorisation.

Bryman (2016) points out that categorisation of research methods and their relevant data sources is subjective, and in some situations not entirely predetermined. He highlights that even if questionnaires are used, the results may still be considered qualitative data (Bryman, 2016). In the current study, how the data was collected and how many participants took part in the study must be considered. It could be argued that the small number of participants will not hold up as a statistically significant sample for drawing wide-ranging conclusions about the success of this research for use within the game design community. However, given that this is a foundational study that serves as a first step on a longer trajectory, and since this study was conducted to determine the next steps towards the augmentation of human design processes, and perhaps the automated play-testing of game levels, it nonetheless makes a significant contribution, despite its small sample size. In order to mitigate the potential conflict of categories, it also has to be acknowledged that generalisation may not be possible based on the results of this study. Instead, the data may only serve to draw conclusions related to the direction future development of multi-agent systems will need to take. However, to be able to examine the data, and for the simple clarity of this thesis, some of the data obtained from game design tests was treated as quantitative, and some as qualitative, as shown in Figure 20, which reflects the way it has been collected, rather than treating all data from these tests as qualitative, based on sample size. The data will also be presented and discussed according to these categorisations, instead of following, for example, a simple chronological theme in the order of collection. The reasoning behind this is that clustering the data by type seems more intuitive than doing so in order of collection, and may make it easier to follow the logic of the approach presented here.

First, the qualitative data will be discussed, then the quantitative data will be analysed. The reason for this order is based on information highlighted in the research design chapter of

this thesis, that is, the need for the triangulation of qualitative data through quantitative means.

This discussion will also consider the semi-structured interviews first, as they seem to provide good insight into what the participants experienced from their perspectives, while using the multi-agent system. Following on, additional, less-structured elements captured through observation and the think-out-loud method will be used to add to the findings of the semi-structured interviews. Given that these elements are heavily dependent on whether the participant was willing and able to verbalise their experience while going through it – a trait that seems to significantly depend on having a talkative personality – somewhat more quiet participants did not reveal much via their own initiative while using the prototype. However, all participants responded well to the semi-structured interview questions, and therefore, they are considered a more complete reflection of participants' experience.

5.1.1 Participating Game Designers

The decision to use semi-structured interviews was discussed and justified in the research design section of this thesis. I believe the flexibility that this type of data collection offers has been hugely beneficial to this project, despite the difficulties that a less-structured approach poses when it comes to data analysis.

In my justification for my participant selection, I point out that access to experienced game designers is difficult, not only because it is a position in game studios that is generally sparse (as creating a computer game requires only a handful of designers and a much larger number of artists and developers). Additionally, most studios bind their employees to non-disclosure agreements (NDAs), and these are generally strictly enforced, a situation that is familiar to me from my film industry work. Accordingly, speaking about one's work is typically restricted, and meaningful research is either conducted internally, and not to be published, or publication is very difficult, both of which hold implications for this study. In summary, it is difficult to find a broad selection of game designers to choose from. I have been successful in finding a number of different participants who cover a diverse range in terms of gender, cultural background, age, gameplay preference, and game design experience. The result is an arguably small number of participants (11) for this study, and variance in expertise, due to the variance among participants in experience as game designers. However, it has been noted that in usability studies, a good rule of thumb for number of participants is 16 ±4 (Alroobaea

& Mayhew, 2014); thus, 11 can be considered acceptable. I am aware that the literature is a bit divided over what a good sample size should be. Some argue that only 5 users are able to highlight 80% of usability problems in prototypes, but only 20 or more participants find at least 95% of all issues (Faulkner, 2003). Others find that 10 ±2 participants are sufficient (Hwang & Salvendy, 2010). Schmettow (2012) however argues that there is no 'magic number' and that a great discrepancy between general and expert users can be demonstrated. Using a less rigid interview style that enables diverting from scripted interview questions helped to offset the discrepancies between the different participants with varying levels of expertise I was able to find for my study. I would also argue that usability testing is strictly aiming to identify faults in a system or interface, whereas this study is explorative and seeks to understand what people think and how they behave in collaboration with a Multi-Agent System.

My participants agreed to submit a pre-participation questionnaire that considers a small number of demographic and gameplay, as well as game design related preferences. The intent of doing this was to gain an understanding of who would be working with the prototype, and how some of their think-out-loud and interview responses may be explained.

The least experienced designer had roughly two years of game design experience, whereas the most experienced game designer had been active in the industry for more than 21 years, and had worked on *Counter Strike Condition Zero*, among other released AAA titles, which made their experience highly relevant to this study.

The pre-participation questionnaire captured three age groups reflecting young adults, middle-aged designers, and older participants, who were mostly born before the dawn of home computer games (ages 18-29, 30-49, 50+). Except for one participant indicating that he belonged to the last group (aged 50+), the remaining participants were evenly distributed across the first two groups, with four in the first group (ages 18-29) and six in the second group (ages 30-49).

Considering job requirements indicated by advertisements in the game design industry, it can be said that this study was conducted with game designers mostly at senior level (5+ years of experience) and a few at junior level (1-3 years).

The gender balance was roughly two thirds in favour of male participants, which possibly reflects a gender bias in the game industry. If I had had access to a larger group of potential participants, I would have preferred to have a balanced gender distribution, as I believe that games are played by both genders, and while female game developers are still under-represented in the industry, I believe this will eventually change, similar to the environments in a number of other industries.

5.1.2 Observations and think-out-loud results

After introducing the game designers to the prototype tools, and after highlighting features and limitations of the prototype system, I left participants to conduct their level design test by having them select the candidates they wanted to be the parent candidates for the next generation of the genetic algorithm. I specifically asked them to think-out-loud, if they were comfortable doing so. During the evaluation, I simply noted any observations and comments that were made by the participant. Some provided a high level of feedback, while others were reasonably quiet.

5.1.2.1 Stated design goals

Some of my participants stated an explicit design goal, sometimes involving several player experience goals:

1. One participant stated that she "really wants a small level where the teams are immediately facing each other" in order to provoke a very fast game, where players do not survive very long. This was different to most design goals, but reflects the style of some small scale, fast paced FPS levels from *Battlefield* and *Call of Duty*.
2. Another game level designer aimed to create a "street that is long and [winding]" so that "there are many corners for cover". This implies the assumption that most corners will have either buildings or street elements that offer shipping containers as elements for cover, which will be a second design goal for the same level.
3. A few designers suggested at the start of their design process that "players should have as much cover as possible, so more buildings are needed". While this is similar to the goal stated in (2), it does not imply that the path defined by the street needed to be very long or winding.
4. Some goals were more abstract, such as: "I just want to make something silly, not your typical map". I had to clarify these goals and the designer stated that the level should

have large open fields without much cover, and a long and straight street to make the map very linear. While this may go against best practice for FPS level design, the participant was not only able to achieve this design goal, but also felt that the final result would be “an interesting addition to a map package that contains more traditional map designs. It would be very quick games though”.

5. Another similar goal, achieved using a different approach, was announced as “aiming to create a ‘mean’ level, where both spawns are close to each other and both are far away from the flag”. This kind of design will “probably introduce a very strategic approach. You [the player] can’t just rush in otherwise you die quickly”. Here it can be seen that some design goals were based on assumptions that can only be verified by play-testing.
6. Some participants were quite specific about typical elements that create mechanics similar to many popular games, and that are in line with what expert designers suggest for FPS levels. For example, one participant stated that they “only want a building if it provides cover at a choke point. Otherwise no buildings, just containers”.

5.1.2.2 Comments related to computational agents

A number of comments were made that relate to the support provided by agents, and in other cases, comments on the absence of the computational agents. As I was aware of agents being switched to an active or inactive status while the tests were conducted, I categorised these comments into two groups, one reflecting the agents helping the designer, and the other related to the agents being inactive. The participants were made aware that the agents would be active and inactive at times, but they did not know when this was the case. Thus, participants were not deceived, but had no knowledge about the state of any particular run.

When the computational agents were active, comments such as “Oh, it understands me” or “It feels like every new generation is much better than the previous” were made. In particular, when participants looked at and also pointed to the two agent candidates in C3 and C4, comments such as, “There are a couple of good levels here”, “Oh, that is a good one” and “I like this one” were made. A more generic “The top row is generally the most interesting” was consistent with both elitist parents and two computational agent suggestions in C1 to C4. These comments were found to be consistent with the eye tracking data, which is presented later.

Statements in stark contrast to those mentioned above were made in runs were the computational agents were inactive. This was, again, reasonably consistent with the eye tracking data as per the following remarks: “It is a bit random, there was a better solution in the previous generation I think”; “It just does not improve enough from selection to selection”; “The tool does the same thing over and over. There is not enough change”; “These candidates are all very random”; “I am not getting any closer to what I want”. These participant statements reflect the common theme dominant in non-agent runs. Some mild frustration was observed in particular when it came to particular design goals, such as “Trying [to] give the red team some cover, but that is not happening”. In other words, the designer was unable to pursue a specific design goal in this particular case. From previous experience, I would not necessarily ascribe this to the impact of the multi-agent system (or in these cases, the lack of a multi-agent system); rather, it may simply have been normal frustration and fatigue, which is to be expected in a pure interactive genetic algorithm, which can take a large number of runs to eventually converge on a set goal (Kruse, 2014). Through indirect interaction with the artefact that is being designed (in this case, FPS game levels), some designers seem to experience a sense of loss of control, simply because the genetic algorithm often takes multiple runs to modify a simple change, which could have been manually altered very quickly. However, this is only true for some specific changes. The genetic algorithm is in principle very fast when it comes to variations and idea generation. It is simply the perception of the user that makes it appear to take longer, due to a lack of direct interaction with the artefact.

It can be said that the response to the multi-agent system, based on observations and think-out-loud recordings, was overall very positive, and that the inactive system showed a stark contrast to the active agents, which confirms these observations. However, there was also a single case in which the active multi-agent system did not perform according to the very straightforward design goal of “some cover and more bends in the pathway” at all. This was stated as an issue by the participant, and I believe that in this isolated case, the algorithm converged into a local minimum without any means to escape. Even after a significant number of additional generations, very minimal changes (likely based on mutation) occurred. Following this event, I experimented with different mutation rates and could not replicate the problem. Aside from an increase in mutation, I believed the diversity agent may have been

useful, but realised that it would need to be re-implemented with some form of memory that tracks a number of previous generations, for the purpose of removing repetitive patterns and enhancing diversity; for example, by adding or removing streets, buildings, and other cover at the same time, rather than randomly focussing on a single parameter (which is the case for the current implementation). I decided against effecting any change to the agent before finishing the trials, as I wanted to keep my results consistent. This change is therefore subject to future research.

Overall, I observed that participants seemed to favour the top right corner of the screen (were the agent suggestions were located) significantly when the agents were active. This observation appears to be supported by some think-out-loud comments made by the designers, specifically and unequivocally when the agents were active. At the same time, participants seemed to randomly select candidates when the agents were not active. I made a note of this observation several times in the observation protocols. This is, however, my personal interpretation of the selections made by the candidates as I observed them, so may be related to my own confirmation bias. Confirmation bias is the notion that one seeks to confirm one's own hypotheses or expectations, even though their veracity may be questionable. This can lead to the selective collection and use of data (Nickerson, 1998). In order to test the observation of whether agents had a significant impact on user choices, I had to unpack the telemetry data, which showed, among other things, the selections that were made in each generation of the genetic algorithm, and whether the agents were active in each run or not. I also asked my participants whether the top right two candidates were of specific interest to them, as I was interested in how the activity of the computational agents had been perceived by designers. Before I present the quantitative data concerning the telemetry system and eye tracking, I will discuss the results of the semi-structured interviews and make an attempt at interpreting the data.

5.1.3 Interview responses

This section presents the qualitative data of the evaluation of the game level design prototype, in particular, the multi-agent system. The catalogue of indicative interview questions can be categorised into three groups: usability of the prototype tool, questions related to the multi-agent system, and a more generically, an open request to comment on anything that the participant felt was important. The former two are likely obvious choices,

while the latter intended to capture any overlooked issues and generic comments, positive or negative. The choices here show that my intent was to gain truly qualitative results that were concerned with how the user perceived their experience, and how they felt about it. When I finalised my research design, I was aware that this data would be highly subjective and prone to misinterpretation; regardless, I nonetheless felt that my quantitative components would help me to conduct a rigorous analysis. The opportunity to possibly obtain insight into the perception of participants while conducting a design task was not only intriguing, but also presented the potential to understand the design process better, and to observe whether a participatory study would align with the extracted metrics from accounts of design processes. In addition, using multiple quantitative methods to triangulate the results could potentially provide me with outcomes that had the same rigour as pure metric data. The following three subsections reflect the aforementioned three categories of interview questions.

5.1.3.1 Usability and User Experience

Prior to asking participants about their perspectives of the multi-agent system, I wanted to observe how they perceived the overall experience, and their thoughts on usability. Any major issues related to general usability could have had a significant impact on the results of the multi-agent system, assuming that if there were problems with the basic use of the tool, it may have distorted the results to the point where they were useless, for example, if a random bug crashed the software intermittently, the genetic algorithm would have been interrupted. Less impactful problems with usability could also have presented issues for further testing. I expected, however, and hoped for significant constructive criticism, which would eventually lead to an interface design iteration, improved handling of the software, and perhaps some additional functionality suggested by testers. I also wanted to capture commentary on problems, as I assumed issues would arise that I had been unable to anticipate while implementing the prototype. Users were aware that this was a software prototype and not a final product; thus, the comments were numerous and helpful, and included a few interesting suggestions.

It is important to note that the overall user experience includes the multi-agent system; therefore, to some extent, the following responses are a reflection of user experience related to immediate interactions such as user interface, and mouse and keyboard, but also an indirect critique of the underlying algorithms, including the genetic algorithm and the multi-

agent system. This is also one of the reasons why I wanted to ask specific questions about the agents, rather than only about general user experience.

The overall experience was described as 'intuitive, 'encouraging with a lot of potential, offers possibilities', 'creative' and 'enjoyable'. I also heard attributes such as 'fun', 'surprises', and 'creates [a] diverse range of levels'. The possibility of setting a design goal and actually achieving it, or get close to doing so, was generally described as 'easy' or 'possible'; however, one participant stated taking a 'playful approach' and therefore did not set any initial design goals, but rather, employed an 'intuitive' and 'iterative' approach. Even participants who were extremely reserved in their assessment of the multi-agent system and stated that they did not believe the tool would be very useful to them, still found the overall experience positive and the tool 'fun to use'.

I also wanted to observe whether there were any serious obstacles to conducting the experiments using the multi-agent system and it was unanimously stated that the tool was easy to use.

Furthermore, participants stated that they could easily create different variations of the same map and liked the possibility of discovering new ideas very quickly.

One participant contributed some criticism by stating that the visual guides were not sufficient for judging the level, and that the tool reacted very aggressively towards one or the other direction; however, these are factors either indicate the request for additional features such as visual guides or were simply factors that arose from the parameterisation of the genetic algorithm. The request for visual guides can be considered a usability issue; however, this participant responded to the experience question that the tool was intuitive and enjoyable, indicating a small contradiction in this regard. Based on these findings, I decided that the tool was overall usable, in particular among expert users who are aware that a software prototype may have minor flaws, and that the main purpose for their participation was the evaluation of an agent-based design system.

5.1.3.2 Multi-Agent System

The responses to the multi-agent system were quite mixed. It was interesting to observe that designers who claimed 10 or less years of experience[4] showed a positive tendency in their

[4] Demographics groups 1-5 years and 6-10 years of experience combined.

responses when asked about the agent system, whereas designers with more than 10 years of experience made significantly fewer encouraging statements. These outputs need to be considered very carefully, given that only a very small number of designers participated in this study. Further investigation is needed to create a more robust picture. Long-standing experts thought that they did 'not need any algorithm to support' them when conducting the trial. They felt that they could find 'good choices' on their own, and that the previous selections in C1 and C2, as well as the suggestions made by the agents in C3 and C4, were not needed, and that they 'selected different levels than those at the top anyway'. Designers with less than 10 years of experience were more enthusiastic, and thought that 'the agents made good suggestions', but these designers also realised that this was not true for all runs and could clearly identify those in which the agents were inactive. Participants responded that some runs took a long time, as 'no good levels were presented by the system for many generation[s]'. Designers seemed to recognise that the system was not always performing as they had hoped, and from observation protocols, it is clear that these runs were without agent support. For example, one participant stated that the multi-agent system offers a 'playful approach' and that it is 'easy to find some interesting suggestions', even though 'some did not turn out very good', which was ascribed to a 'lack of options by the algorithm'. The comment (made in a case where the agents were inactive) makes sense in light of five runs having been conducted by this individual, of which two did not have the agent system active. The observation protocol confirms that the two sessions without agents were perceived as slow and difficult. This participant also made a very interesting comment about 'best design practice', which an algorithm should implement. The comment included a suggestion to look at existing maps 'like Dust2 and analyse classic elements and suggest them to the designer'. Given that both 'Dust' and 'Dust2' are maps created by one of the designers that I used to generate my heuristics, this felt like a confirmation for this study having taken the right direction; at the same time, however, it also served as an indication that the heuristics may need to be developed further, and reassessed in light of this statement. I understand that a heuristic for a designer expert agent is likely never directly visible to the user of a multi-agent system; nonetheless, perhaps if a three-dimensional map layout with elevations or building interiors across multiple floors was used, a stronger emphasis on best practices demonstrated in highly successful maps must be considered.

Finally, five participants indicated that the system 'offers interesting alternatives' and 'rather unexpected' alternatives[5] in the top left corner, where the diversity agent seemed to have offered a solution that was different to the designer expert agent, and both candidates were considered compelling. I will discuss the limitations that apply to this finding in section 6.3.3, but I consider these responses quite encouraging, as they highlight that the multi-agent system appears to add value to the decision-making process of designers.

5.1.3.3 Feature suggestions

The advantage of semi-structured interviews is that it provides the option to veer off-script if a particular issue arises during the interview. Ideas for feature changes or additions emerged on a number of occasions in my first few interviews, and I included a query about new functionality in every subsequent discussion. This section reviews common ideas that arose from interview responses.

1. Visual aids

   Participants suggested that it would be helpful to see the grid that is being used to place elements such as streets, buildings, or spawn points. A visual guide may provide a quick way to evaluate, for example, path length, distance of spawn points to the flag pole, and other similar metrics that participants use to assess the quality of a candidate, in order to make a decision for or against it. This may indeed be a helpful feature, particularly if users feel that it saves time and removes some of the strain put on the attention span of each user. Fighting fatigue in interactive genetic algorithms is a dominant issue, and if a visual guide in the form of a grid helps to mitigate even a small amount of said fatigue, it will be beneficial to the overall process. It may increase the number of runs that the user is able to undertake without becoming tired and may also increase the speed at which a particular design problem is solved, which increases the productivity of the design process.

   A second aid that has been requested multiple times is a debug overlay that shows what the agents do, and how they assess the levels. While the experiments were designed to keep the activity of agents from the user in order to be able to create a baseline, by switching the agents off for some runs (which would be revealed if a

[5] In addition to similar comments related to variants of the two agent suggestions.

debug overlay showed the decisions and activity of the agents), the argument that a designer needs to understand the agent system they are working with in order to maximise its benefits to the design process (Seidel et al., 2018) supports this feature suggestion, and will be worth testing in future.

2. Workflow improvements

   Workflow improvements capture a range of general suggestions that are related to the workflow implemented in the prototype, rather than suggestions specific to game level design. The latter are captured in the following bullet points.

   a. A feature repeatedly asked for was an *undo* function. While this would be possible, and perhaps desirable in a commercial design tool, and although I considered this during my planning phase for the design prototype, I consciously decided against it. An *undo* function in the context of a genetic algorithm does not seem like the correct solution at first glance, because the genetic algorithm is based on the concept that solutions can be 'pushed' into different directions by making selections accordingly. However, I do understand why designers, who are used to virtually limitless *undo* steps in their familiar tools, would request this as a convenience tool. Moreover, I am aware of the time-saving nature such a function would offer, given that pushing the algorithm into a different direction may cost a few runs, which translates to time spent, and ultimately means increased user fatigue as well. Some designers considered the two previous parents (elitist approach) in the top left corner of the candidate tiles a compromise, and I believe this to be a good way to view this. On the one hand, it keeps the genetic algorithm pure in the sense that evolution cannot wind back on the user's request, but at the same time, access to at least the previous selection is still possible.

      The ability to exclude certain candidates from future breeding is another workflow improvement that has been requested a number of times.

   b. There are two sides to this argument. On the one hand, additional functions in the design prototype, which runs a process similar to interactive genetic algorithms, and presumably carries the same issues such as user fatigue, are likely to amplify problems. If the user performs additional tasks for each run,

attention span may be exhausted even quicker, which may result in fewer runs. On the other hand, having a mechanism that enables users to exclude certain candidates, or even more specifically, certain features of candidates, may lead to an even faster convergence to a final solution. Accelerating the genetic algorithm may therefore be an approach for counteracting fatigue. To establish whether this is the case, future work is required. I suggest the possible implementation of an exclusion agent in 6.4.2.

3. Level design

   Specific suggestions that are directly linked to computer game level design emerged in a few interviews. Two main themes pertaining to feature suggestions were apparent; first, constraints prior to initiating the genetic algorithm, and second, constraints that users add while conducting the selections for the algorithm.

   a. A number of designers asked for additional manual control of the actual level design process. Two participants suggested the introduction of design constraints, which advanced users could define prior to using the actual design tool. These constraints can include the length of the path, how much branching occurs in a path, how much curvature is in the path, whether a rather square and parallel New York-style branching system, or a more organic European city-style branch is required, where the spawn points are, and whether the number of obstacles is very high from the start, as opposed to the algorithm starting without any/very few buildings and containers, among others. To some degree, these features resemble a number of extracted design features from expert accounts regarding their design process. Thus, first, given that a DEA is part of the multi-agent system, and that its features have to be defined regardless, manual control is in principle not difficult to add. An agent that manages user constraints can be added to the multi-agent system, to help shape the breeding pool even faster towards the ideas of the user. However, I see a degree of danger in doing so. Given that participants thought of the system as a tool that encourages exploration, enhances curiosity, and offers a playful approach to level generation, a set of constraints set by the individual user (rather than the collective experience of design experts, who 'contribute' their workflow and decision-making processes to the system) may also give

rise to a very narrow and preconceived design workflow. It implies that experts (or 'advanced' users) already know what the end result might look like, a notion that contradicts everything that makes design an innovative field. Instead of knowing what the outcome will be, design should be an iterative process that develops as a result of discovery, happy accidents, and deep exploration, alongside the application of expertise and skill. I believe a constraint agent would actually be counterproductive to a healthy design process.

b. Two participants suggested implementing a locking mechanism for spawn points, that is, the ability for users to ensure at any point in any run that both spawn points remain in their current grid location and evolve only the other features such as path length, path curvature, and obstacles through the genetic algorithm. I can see how a designer may be wanting to design a map where the position and proximity to the flag pole and the opposite spawn location are predefined by the user, and only the curvature and length of the street is influenced by the Genetic Algorithm. This seems like an interesting addition, and follows Kosorukoff (2001), who suggests including the human agent into operators, as opposed to making available only selection in an interactive genetic algorithm. One of these potential operators might be mutation, while the aforementioned locking mechanism essentially freezes part of the candidates' DNA; this is similar to mutation, in that it directly interferes with the chromosome, not only by actively changing it, but by actively protecting it against change.

c. Another suggestion, not dissimilar to (b), is the manual selection or deselection of buildings and containers. This represents another interesting potential change to the system in future, that also requires further investigation. It would be interesting to establish whether this will lead to unintended consequences such as additional amplification of user fatigue, or if it can potentially accelerate the overall process, and thereby serve as a measure for reducing fatigue. Another aspect of adding functions like building placement or spawn locking mechanisms is the impact this will have on the final level. It would be worth testing a range of levels created by an agent system that

augments the user, and has very few functions that are directly controlled by the user, against a system where the user has deep control of the underlying genetic algorithm and its operators, in a bid to establish whether the added control leads to any playability improvements, or if these features are possibly based on the fear among designers to submit control to an autonomous process.

5.1.4 Telemetry data

Telemetry data consists of keyboard and mouse inputs, ranking of individual candidates for debugging purposes, ranking of the entire breeding pool, and ranking of each population.

I wanted to capture the keyboard and mouse inputs to keep track of user selections, where I needed to verify how many times the user selected a random candidate, and when the designer selected either an agent-selected candidate, or simply their own previous selection again (the elitist candidates). I needed to capture each generation, which would enable me to generate a distribution over time and observe whether there was any tendency to use the suggestion made by the agents. In runs where the agents were inactive, I expected a more random distribution across all 16 candidates. Runs with inactive agents would also allow me to observe whether a user would simply select the top right corner out of habit.

5.1.4.1 Breeding pool fitness

This study employs a derivative of an interactive genetic algorithm with the human designer acting as the selection operator. The fitness ranking is computed to allow agents to filter the breeding pool and is not used as a fitness function as such. Capturing the ranking data of the breeding pool allowed me to plot the ranks for all generations to see if there was an increase (essentially, an increase in fitness of the breeding pool) over time. Given that this increase is a direct result of the decisions made by the designer expert agent, it would provide me with an indication of whether the agent had an impact on the overall fitness increase. In order to assess the fitness (or rank), I plotted the maximum pool fitness of a number of individual runs. I combined runs with agents and runs without agents into two different graphs, as shown in Figure 21 and Figure 22. Although it seems like the clusters of lines indicate an overall higher performance when the agents were active, the variance between low performing runs and high performing runs was quite significant and may have distorted the outcome. Therefore, I plotted a mean for all data points of all runs with agents, and the same for all runs without

agents being active, into a single graph, as shown in Figure 23. The light green line represents the mean across all runs with active agents, while the red line shows the mean of all runs without agents. The agents appear to have a significant impact on how fast the fitness increases. With agents, a much higher average pool fitness is evidenced than without agents.

It is important to note that I had to normalise the length of the runs, as the decision to terminate a run was driven by the designer, which led to some short runs with only 12 generations, and some much longer runs with up to 76 generations. I normalised the data to 100 data points, using linear interpolation between data points.

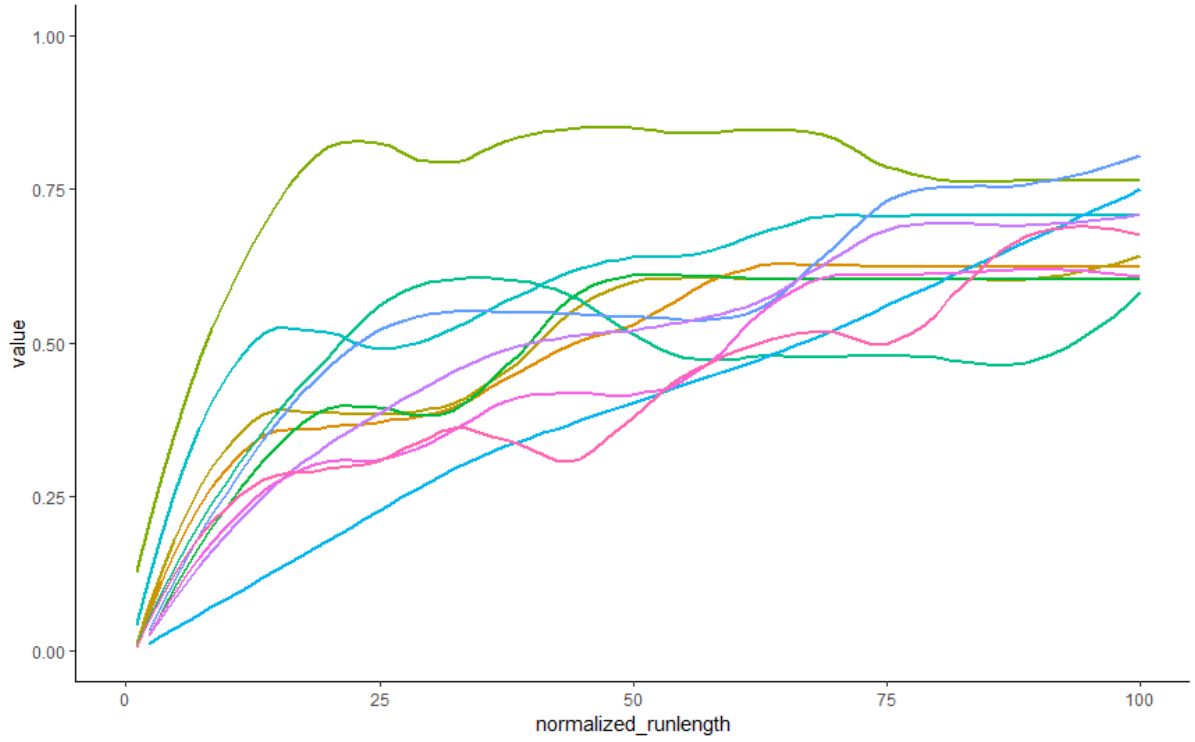


Figure 20 - Maximum fitness value, agents active. Each colour represents a different run for easier visual reading.

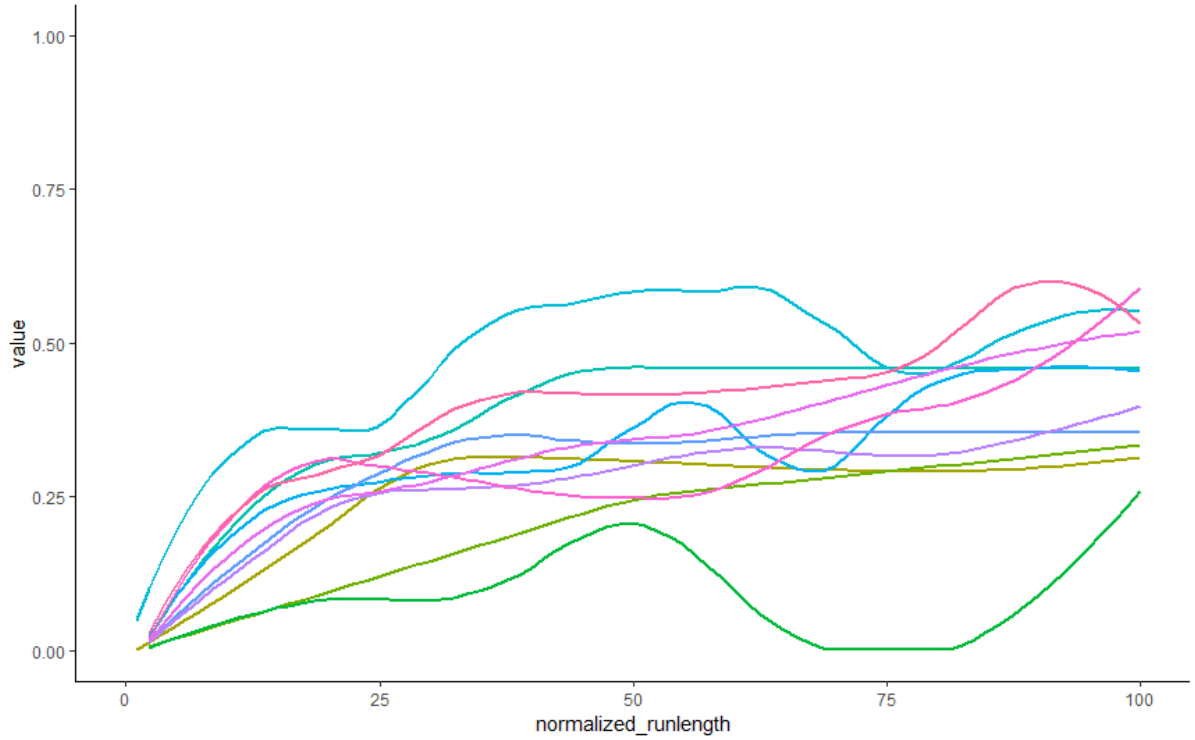


Figure 21 - Maximum fitness value, agents inactive. Each colour represents a different run for easier visual reading.

While this increased the error in some cases, it did not pose a problem to the overall comparison between runs with agents and without agents, assuming that the distribution between shorter runs and longer runs would be similar in both cases. While the length of runs without agents tended to be longer (by roughly 20%) than the length of runs with active agents, the difference was surprisingly small. Prior to starting the experiments, I expected the runs without agents to be significantly longer than the ones with agents, based on my previous experience with interactive genetic algorithms. However, considering that a large number of runs without agents never reached the pool fitness of runs with active agents, and given that the termination was initiated not by an objective goal such as fitness of the final candidate, or a very specific design goal that designers had to reach, participants seemed to simply terminate the session after having conducted a large number of runs, and felt that they were 'not getting anywhere' in the case of the inactive agents.

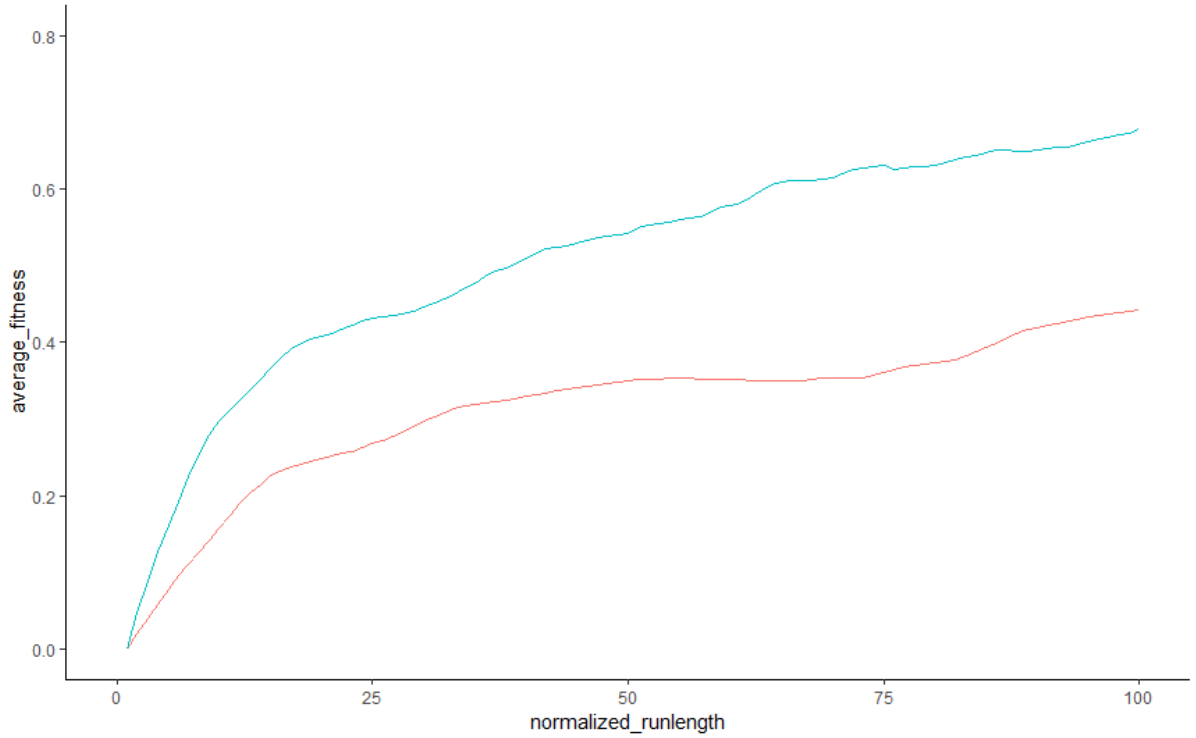


Figure 22 - Maximum pool fitness of all runs combined. The blue line signifies 'with agents' and red line 'without agents'.

As such, the provided graphs need to be considered carefully. They are not an absolute reflection of the performance of the agent system, given that the designers were able to terminate after as many runs as they felt were appropriate for reaching a playable level. This implies that some designers may have terminated despite levels not having reached very high quality. An absolute measure does not exist in this context. One suggestion for future studies aimed at establishing whether two levels with different ranks can still both be considered immersive, playable game levels, is to select top performers from both active agent and inactive agent interactive runs and find players to test them. Another potential solution may be a study that sets very narrow and specific design goals. Both of these ideas are subject to interpretation (by either the players or the designers and researchers), but can potentially help to understand the underlying phenomenon somewhat better. Despite these differences, however, it seems that the agents contribute to a much faster increase in fitness, and also to a higher overall ranking within the breeding pool.

There are a number of other interesting points to be raised here. First, there is a logarithmic-looking increase in the fitness of breeding pools. The longer the runs, the smaller the increase of fitness of pools. This was likely due to the small population size in comparison to the breeding pool. The breeding pool comprised 70% candidates that were bred based on the

parents, and 30% bred through random DNA. For roughly 20 generations, the decisions made by the designer seemed to have a significant impact on fitness, and the increase happened rapidly. Thereafter, the increase slowed down and only marginal improvements in rank/ fitness were observed. This can be interpreted as a potential failure of the computational agents. If the user is the main contributor to the increase, given that their decision contributes 70% of the mating pool, the agents may not be as useful as I had hoped. It is important to keep in mind, however, how these user decisions were made. The user observed either their own selection, and a number of randomly selected and mutated candidates in the case of inactive agents, or the user was able to choose from their own selections, two agent-suggested candidates, and a number of randomly selected and mutated candidates (in the case of activated agents). The much quicker increase at the start of the runs with activated agents may indicate that the agents did, in fact, play a role. This can, however, only be verified by looking at what the user decided to select, and perhaps to some extent, by what caught the attention of the user more often. The results for these two important additions are presented in sections 5.2.4.2 and 5.2.5.

Another observation is the maximum fitness rank reached by any of the runs. Neither the agent-driven nor the purely user-driven runs reached the full score of 1.0, which may infer a number of things. Perhaps the runs were not sufficiently long. If a genetic algorithm is not able to run through enough iterations to reach the stopping criteria, optimisation will terminate prematurely. If this was the case, the user simply did not make enough selections to reach the theoretical maximum fitness (playability) defined by the designer expert agent. Another possible explanation is that the ranking score is simply not reachable, and that the criteria that the agent is based on are flawed. I believe the latter is not the case. Looking at the combined plots of the mean of all runs, it can be seen that the fitness ranking did not achieve saturation, and the maximum continued increasing in an almost linear fashion following the initial rapid increase over 20 generations. I believe that a fitness value close to the theoretical maximum of 1.0 can be reached, given enough iterations. Previous studies and my own observations confirm that user fatigue arose after some time. Additionally, the subjective nature of what a ‘good, playable level’ is may also have played a role here. If users felt that there was no significant improvement to be gained after many selections, they may have terminated the run. Looking at the score that the average run reached, a fitness of 0.65

may have been considered sufficient. In conclusion, I believe that user fatigue contributed to premature termination, but that this can be mitigated by adding a user preference agent (Kruse & Connor, 2015). Furthermore, it does not mean that the resulting levels are not playable. The theoretical maximum of 1.0 is based on expert accounts, that includes professionals who created some of the most popular and highly acclaimed FPS levels ever. Here, the score can simply serve as what it is – a theoretical maximum. My aspiration, however, is to design an agent system that gets close to producing the best possible levels. I believe that supporting the designer through a preference agent can improve these results.

5.1.4.2 User selections

As indicated in the methodology section of this thesis, I initially decided to present the selections made by the human designer as a pseudo-heatmap visualisation, rather than pure numerical data, as numbers may not particularly intuitive to some, in which case they would offer little value. The table in Appendix A shows an example of such tabular representation. The first column represents the generation (zero indexed), while columns *Parent1* and *Parent2* are the index of the selected candidates. It is easy to see why a different tool for their analysis was needed. Figure 24 is an example indicating user input where it occurred. Given that the user's mouse-clicks generally occurred somewhere inside the candidate tiles (which confuses the picture due to their random click positions), the clicks have been re-centred towards the middle of each tile for easier reading. The light green and light blue colours indicate each parent, while the small text field on the lower left is for debugging purposes only. This simplified representation shows an indication of the user selections at first glance and makes it straightforward and very intuitive to observe where most clicks occurred.

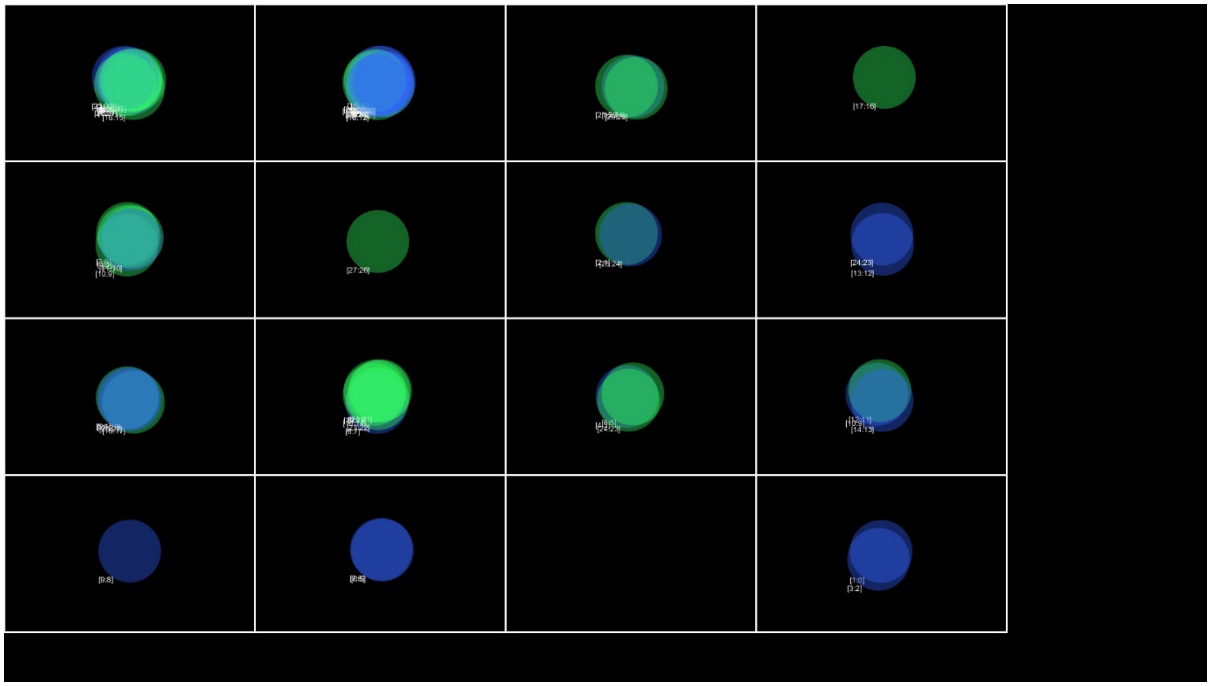


Figure 23 - Pseudo heatmap representing user input. Dark blue-green denotes fewer clicks and light blue-green more clicks.

Employing this method to gain a better understanding of user selections did not prove to be of great value. I also found, when comparing these particular visual maps to numerical data, that it was difficult to interpret and draw robust conclusions from the former.

Thus, this study used simple statistical approaches to extract meaning from the telemetry data. I ran each telemetry file from every run in RStudio and investigated the relationship between frequency of clicks and the candidates, as shown in Figure 25. I was particularly interested in noting whether there was a difference between runs where the agents were inactive and runs where the agents were active; their two highest ranked recommendations are shown in C3 and C4. I also wanted to observe whether users made use of the elitist parents in C1 and C2, which gave them an opportunity to incorporate the previous selection into the current run.

Figure 25 depicts a plot of all selections by all users across all runs, where the agent system was deactivated. The intent behind these runs was to create a baseline for the multi-agent system. Green represents the two elitist parents, whereas the blue bars represent the number of clicks for the random candidates.

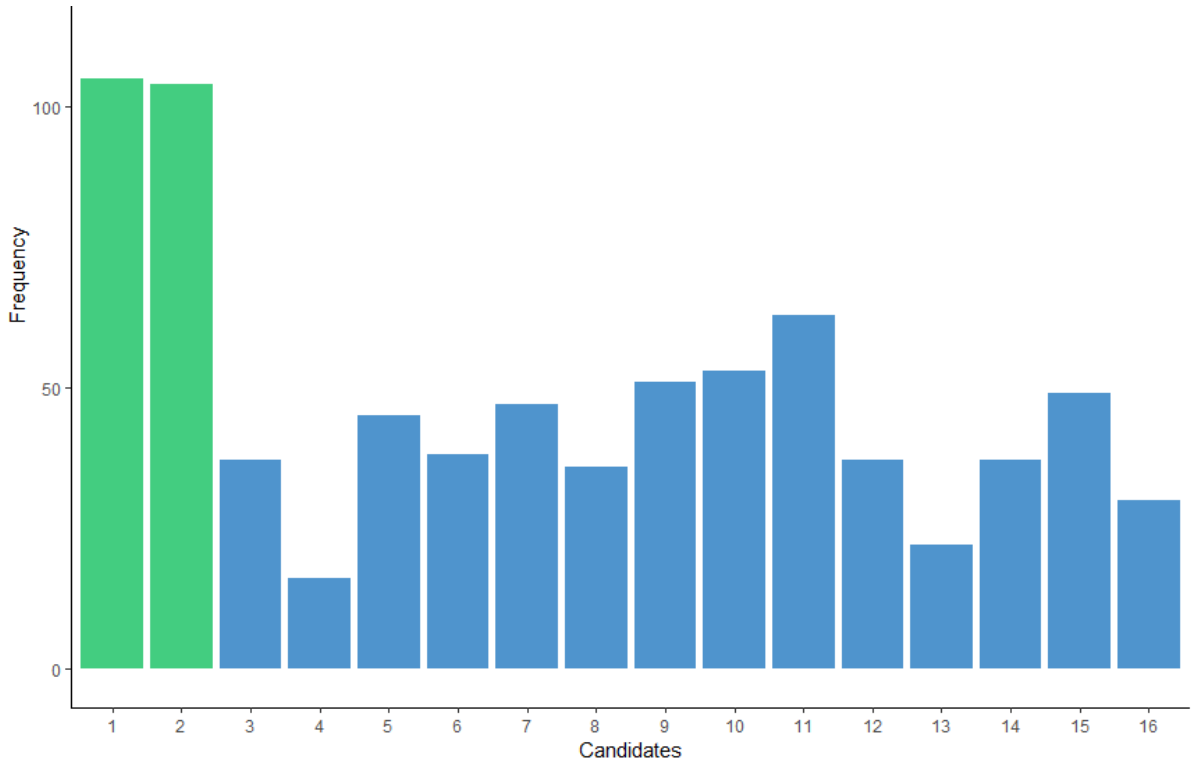


Figure 24 - Plot across all runs with agents inactive.

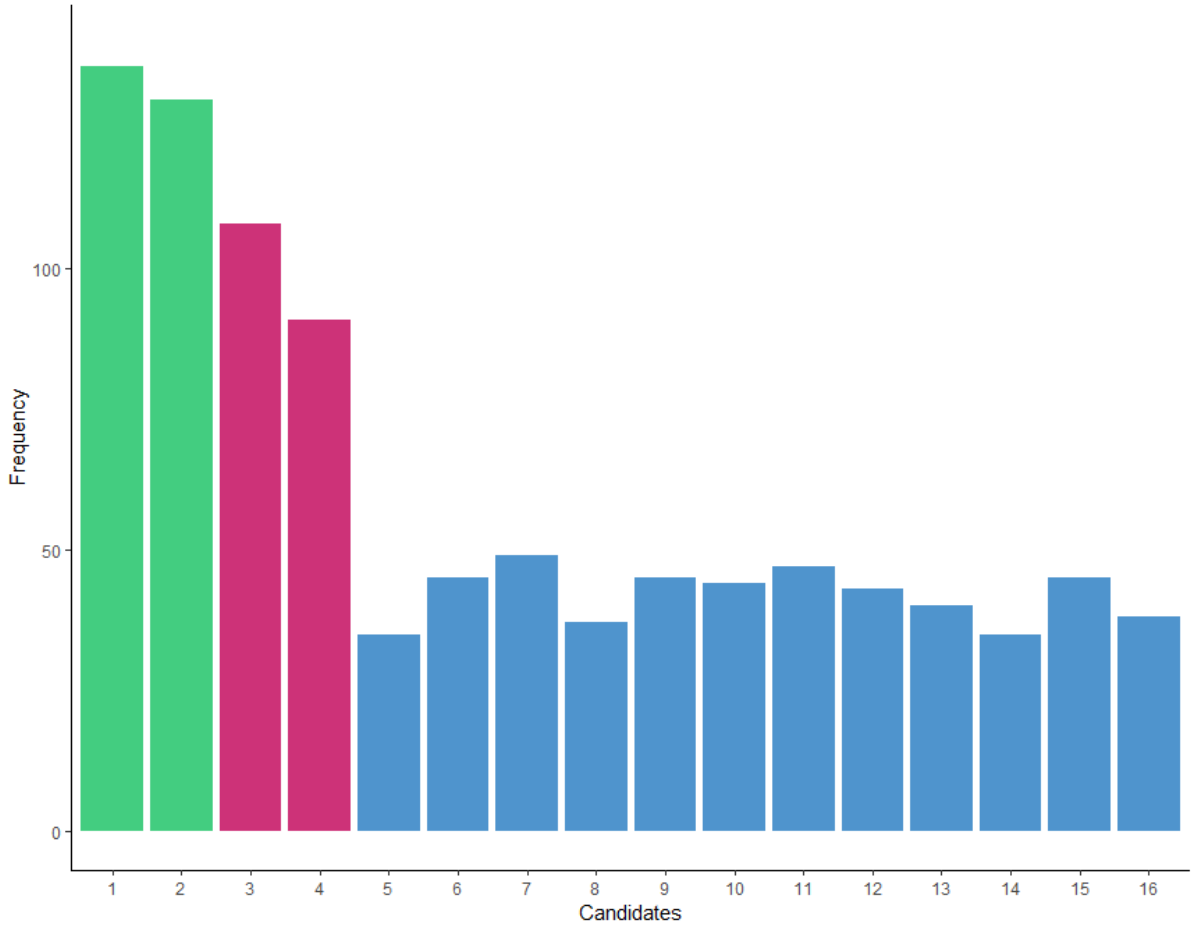


Figure 25 - User selections plot across all runs with agents active.

Participants showed a significant preference of the two elitist parents, which at first glance may seem surprising. My observations and think-out-loud data, however, confirm that users tended to select one of the previous parents, and combine it with a random candidate for breeding. Participants stated that they used this mix of previous selection and new choice to 'explore additional options', without losing specific features they wished to preserve. Some viewed it as an opportunity to 'create potentially more interesting variations' of an already 'good' level. Therefore, a certain preference is to be expected. Additionally, the intuitive conclusion that there was a very high preference for the elitist levels needs to be viewed in relative terms. Overall, either of the two elitist levels were selected 209 times, but random levels were selected 561 times. Given that the elitist levels presumably represent a level that the user considered the best possible selection in the previous run (and therefore selected it), it should be expected that this choice would not be fully abandoned in favour of random levels from the breeding pool. It can, however, be said that users seem to use the system to evolve previous selections, rather than breed, based on random choices. Assuming that participants were following a structured approach towards certain experience goals, it seems logical that roughly every third selection involved previous choices.

Figure 26 shows a plot of all selections by all users across all runs with active agents. Green highlights the two elitist parents, red the two agent suggestions, and blue all randomly selected candidates.

There is a relatively even distribution of clicks across all random candidates in blue. However, candidates in C1 to C4 show a significantly higher click frequency. The two agent selections average twice the number of clicks, while the two elitist parents are slightly higher, similar to runs without agents active. Acknowledging the significant preference for the agent-based suggestions feels like stating the obvious, but rather than jumping to possible conclusions, I would like to note a few important points. The outcomes here are slightly distorted, as these two bar-graphs show all clicks across all runs for all users. However, as the decision to terminate a run had been up to the user, the overall count for runs with agents (968 clicks), as opposed to runs without agents (770 clicks), differed by almost 200 clicks. This may have led to a slightly distorted interpretation. The jump in clicks for C3 and C4 from no agents to active agents was quite significant (from 53 clicks to 201 clicks). C3 and C4 received four times as many clicks when the agents placed their suggestions here, compared to without any agent

support. Participants seemed to have responded to the suggestions in a very positive manner, placing their importance almost on par with the elitist suggestions, at least according to their selection behaviour.

5.1.5 Eye tracking results

I have previously noted that I wanted to treat this study as an opportunity to challenge myself, to use a variety of different research methods in order to make mistakes, learn from these mistakes, and expand my own horizons. Given that I already had a quantitative component in the form of telemetry data from the software prototype in my research design, and given that this keyboard and mouse data would allow me to triangulate the qualitative results of my study, the addition of eye tracking as yet another quantitative data source could have been viewed as redundant. However, 'hindsight is bliss', as the proverb states, and I have to admit that I did not foresee the value that eye tracking subsequently added to my findings. Additionally, while unnecessary data may only produce more noise, I believe that this additional metric would be useful, because it represents a very different quality in the form of observing the designer's decision-making process, compared to keyboard and mouse data. While keyboard and mouse clicks merely indicate the actual decisions made by the designer, the eye tracking data provides continuous insight into what the designer physically looked at while forming the basis for these decisions. Instead of a simple snapshot (a mouse or keyboard click), it adds a temporal component. Though it is still bound to interpretation, I felt that it would help me to capture the time during which the decisions had been formed, rather than the result of this decision-making process per se. This proved to be more valuable than I anticipated, and I believe it adds to the overall rigour of my research, as I will show in this section of my thesis.

The heatmaps are listed in Appendices 8.2 and 8.3, based on whether the agents were active or not. For the sake of clarity, I will only provide two examples here, one from each category. I believe these are representative of the overall results.

Figure 27 shows an example of eye tracking data with an active multi-agent system. The top left cells, C1 and C2, show light heat, which indicates very moderate interest on the part of the user, as does C7. C3 and C4, on the other hand, show very intense heat, an indication that the user had an elevated interest in these candidates. While not all heatmaps show strong indications like this, most of them reveal a common pattern: either C3 or C4 (or both) indicate

strong focus on the part of the user, with moderate interest in C1 and C2, and a low average interest in C5 to C16.

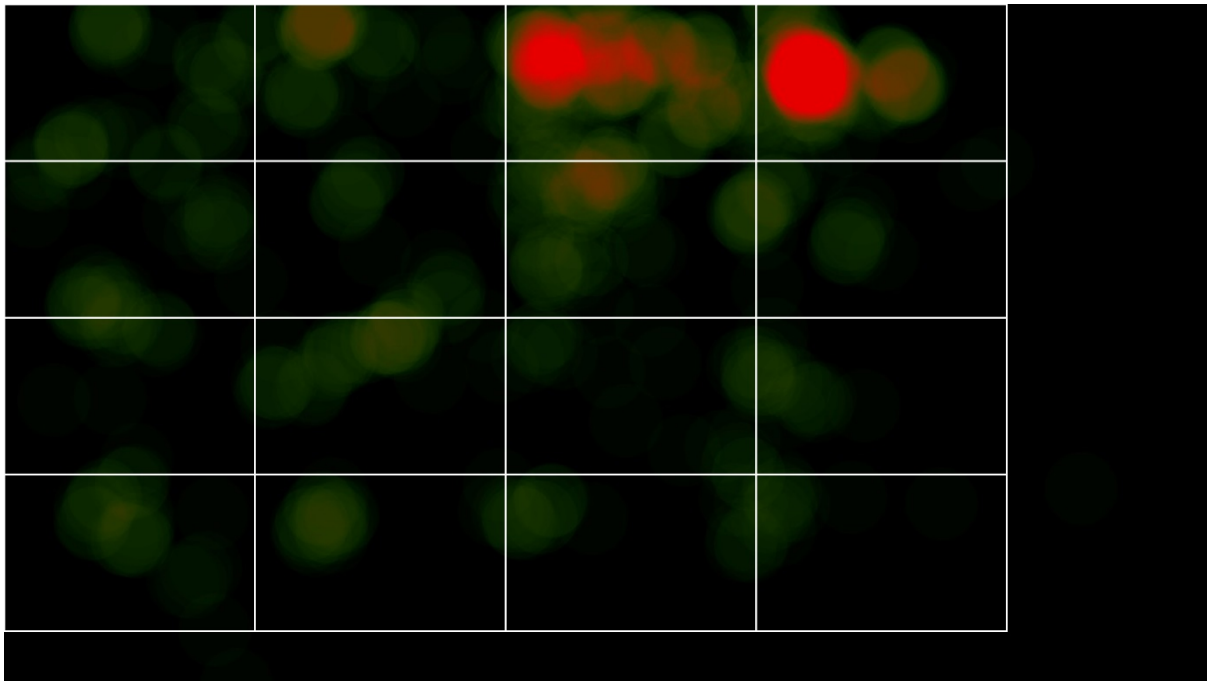


Figure 26 - Heatmap of eye tracking data (active agents).

Figure 28 shows a plot of a run without agents. It can be seen that user interest was fairly evenly spread across all candidates. The heatmap does not indicate a particular importance for any candidate.

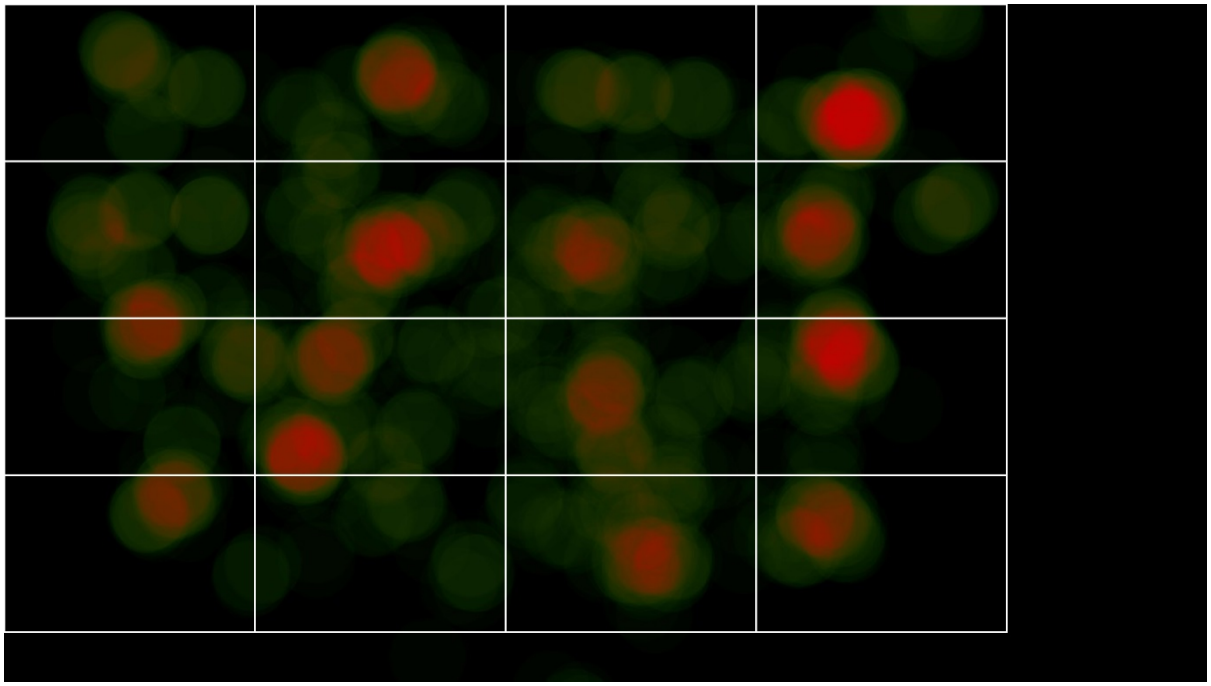


Figure 27 - Eye tracking heatmap without agents.

Contrasting any claims made by some participants that they were not biased towards the agent suggestions, which they knew were present in C3 and C4, there is a strong indication that whenever the agents were active, the users seemed to look at the top right corner a great deal (C3 or C4). This is in line with a number of notes made in the observation protocols. While this does not hold true for all cases, there appears to have been a bias for this screen position. My first possible explanation when I initially looked at these images was that users possibly had a bias towards focusing on the top right corner. All participants hailed from a Western background and identifies as European, New Zealand European, American, or Pakeha. As such, my initial expectation was, in fact, a bias towards the top left corner, given that this is where Western reading starts, and I believed that this trained behaviour may have an impact on my findings. I assumed that, based on Western reading, there may have been a preference for looking at the top part of the screen first, and perhaps for a longer time, which would have explained an accumulation of gaze blobs in the top row. This was, however, not the case, and I will show why not presently. My second assumption, based on the images above, was that for a reason unknown to me and unrelated to the position of the outputs of the designer expert agent, users may have had a tendency to favour the top right. Both these premises turned out to be false. The following depictions of runs without any agent suggestions clearly contradict both these assumptions. In these runs, no bias towards the top right corner is visible at all; however, this would have been the case had there been a cause unrelated to the test, such as our natural tendency to start at the top left, according to reading habits in certain cultures. I believe that the bias towards the top right can be cautiously assumed to be linked to the presence of the agent-based solutions in C3 and C4, and the tendency towards C1 and C2 may be the result of these being the elitist parent candidates. Designers seemed to show a preference for these candidates. I am aware that ‘preference’ does not mean ‘confirmation’ at all, and that the heatmaps, as well as the possible reasons for what they indicate, are subject to my own interpretations; however, the observed preference could be an indication that the agents had an impact on designers’ decision-making processes and drew designers’ attention towards them. Keeping in mind that this is a foundational study that can only serve as a precursor to a broader and deeper investigation into the quality of multi-agent systems, I believe that these indications are quite positive signals. Moreover, it can be said that these quantitative measures signal a good option for

verifying interview responses and observations, and I would employ these or similar methods again to triangulate qualitative results in future research.

A potential anomaly in the data was visible in runs 12 to 14 without agents (Appendix C). A participant seemed to show a preference for the left side of the grid, and barely looked at any of the candidates in C4, C8, C12, and C16. However, given the insignificant sample size, I believe this claim may not hold up to additional testing. To substantiate this further, the same participant conducted runs 11 and 12 (with agents). The same pattern could not be confirmed there; instead, a number of hotspots are visible on the right side of the grid. I believe that the bias towards the left side when tested without agents happened by pure chance, but cannot dismiss nor verify this, as doing so requires additional research. Given that the eye tracking only complements the selections (clicks) and interview responses, I believe this irregularity had no significant impact on the results of this study.

Another potential anomaly was hidden in the strong bias towards either C3 or C4, but almost never towards both at the same time. This seemed odd at first, but a possible explanation is that both agent-based candidates in C3 and C4 are often very similar in path length, starting and end point positions for teams, flag pole position, and even (for the most part) the position of obstacles for cover and breaking line of sight. These similarities are owed to the selections that C3 and C4 represent, namely, the two top performers in the ranked breeding pool. Particularly in an advanced breeding pool, after a number of generations, the potential candidates will converge towards similar features, given that a large number of imperfections would have been bred out of the DNA. The agents working on the selection process alongside the user will accelerate the increase in fitness of each generation, and this will give rise to solutions that were either similarly good or bad in terms of fulfilling the design expert criteria. Selecting the top two performers of the breeding pool will yield two very similar selections. However, this does not explain why the users chose to favour either C3 or C4, and why they did not alternate between the two. This is certainly a question that needs to be asked in future. However, given that the analysis of the eye tracking data did not happen in situ, I did not have the opportunity to include this question in the semi-structured interviews. My best guess is that users may start out favouring certain candidate tiles, because subconsciously, they found solutions in these tiles that they considered having been successful in previous runs, and therefore tended to again start looking for similar successes at the same position.

This is of course only an assumption; it would be interesting to investigate the reason for this bias in a separate study.

Most of the measurement points were within the grid-layout of the candidate tiles. However, in some cases, there are points outside the grid, which can easily be explained. These were created by the user looking away from the candidate tiles. In some cases, this was simply caused by looking around, for example, at the keyboard or away from the screen. This is very normal in many activities, and I did not consider it important to filter these strays, simply because users were generally very focused on the task and as such, looking at the relevant content. In some cases, there are significant amounts of blobs visible on the right hand side of the screen. This is also easy to explain, as this was the location where the two three-dimensional depictions of the two currently selected parents were located. Some users made more use of this feature than others, resulting in a slightly higher density outside the 16 tiles.

A surprising element was evident in the comparison of the semi-structured interview responses given by the designers and eye tracking data, in relation to the question of whether participants felt that the computational agents added any value, and whether they could see a difference between the runs. At this point, participants were made aware of the fact that the agents were not active in all runs, but only in roughly half of them. The responses were mixed. Some participants felt that the agents may have added interesting variants in some cases. Others pointed (correctly) to a run that was perceived as problematic, as it did not lead to any desired result, or at least, not very quickly. These were indeed the runs in which the agents were switched off. However, all of these more or less constructive statements were contradicted by the two most senior designers that took part in this study. Both strongly rejected the idea that the agents were of any value and confirmed they would only follow their own experience. They were also not able to tell the difference between any of the runs. A surprising aspect is noted here: both of these designers produced the following eye tracking and selection (mouse and keyboard) results. It is clearly visible in the figures (see Appendix) that they made heavy use of the agent-based suggestions and looked at them much more frequently than any of the other candidates, aside from their own previous selections (elitist candidates). There are two possible explanations for this. First, perhaps they were very much in line with the agents, and for this reason, the heatmap and selections show a strong preference of these specific solutions. This will serve as an indication that the expert agent is

performing reasonably well, subject to testing with a larger veteran game designer group. Alternatively, the longer an expert spent designing levels, in particular when they were long-term domain experts, the more significantly they became pre-occupied by their own views, and strongly believed that their expertise and skills were more important to their success than the tools they used. While this may be true, it indicates a much stronger bias than that shown by intermediate or junior game designers, who seemed to be more open to new technologies, and positively embraced the opportunities that this system offered them.

6 Discussion and Recommendations

The main outcomes of this study can be found in the evaluation of the game level design prototype, a multi-agent system that employs computational agents and a human designer to run an evolutionary algorithm. The outputs of this prototype are simple, yet still offer playable game levels that provide sufficient immersion for promoting gameplay. The results of said evaluation show that the agents employed in this study had a significant impact on designers' decision-making processes. When active, the agents selected the highest-ranked candidates from the breeding pool and offered them amongst a number of elitist and random level candidates to the user. The core of this system is an agent modelled on expert designers' accounts of their design processes. This agent seems to capture core qualities of 'good' levels, i.e. levels with a high playability score, based on a selection of domain-specific features that are applied during the design period, and these core qualities appear to be what game level designers are attracted to.

The following sections conclude the final chapter of this thesis by considering the original contributions made herein, the findings and limitations of the presented work, as well as some suggestions for future research.

6.1 Contributions

This thesis makes a number of contributions to the field of procedural content generation and evaluation. The most significant is a novel designer expert agent that models the design process of expert game level designers for FPS games. Based on several accounts of the decision-making process behind FPS level designs, a number of metrics have been extracted and implemented as a computational agent. This agent has been evaluated as part of a human-machine multi-agent system running an evolutionary design task. The results appear quite promising and provides a solid foundation for future research in this domain. The study implements and uses an evaluator based on how designers think, contrary to the commonly applied player-centric approach. This fills a gap in the literature identified by recent publications.

An additional contribution of this foundational study is that it employs a low-cost eye tracking device as a tool for triangulating qualitative results in design research. The workflow and code will be published in open source format to help other researchers interested in utilising this

powerful tool, but who do not have the know-how to implement the software themselves. The study also uses telemetry for triangulation. The need for applying rigour via the use of complementary interviews and observations made using quantitative tools was confirmed on several occasions throughout the study, as qualitative results would have led to a potentially distorted interpretation. The eye tracking in particular is very easy to add, and cost-effective, requiring less than $200 for the device[6], and is transparent to the user, as it does not appear as invasive as other biometric triangulation tools. The cognitive biases of expert designers appear to have had an impact on their decisions, and quantitative methods can mitigate these impacts. Together, these aspects render this solution accessible to a broad population of researchers and has the potential to add rigour to qualitative studies.

6.2 Findings

In this section, I will highlight how the research question and its sub-questions have been addressed, starting with the sub-questions.

6.2.1 Sub-question 1

> (Q1) Are cognitive agents as game content evaluators considered a useful addition to game level design by experienced game designers?

This study extracted a range of qualitative data from expert accounts and modelled two agents, based on traits that emerged as a scheme from the accounts. This DEA made level design suggestions to participating game levels designers, who showed a clear preference for these suggestions when compared to a baseline, where the DEA was inactive, and random levels were selected from the breeding pool for generation presented to participants. While the number of participants in this study does not represent a statistically viable sample (with more research required), the initial results obtained by this study indicate a promising outcome.

6.2.2 Sub-question 2

> (Q2) Can a cognitive model be devised as an expert system, and can it employ knowledge extracted from secondary sources such as the personal accounts of expert game level designers?

[6] Disclaimer: I am not hired or paid by Tobii to promote their products.

This research shows that a simple approach such as using a rule-based expert system, rather than more sophisticated, deep neural network-based machine learners, can create an effective cognitive model of expert designers. 'Simple' instead of 'sophisticated' has been the mantra of this study, and I believe this has enabled me to discover some interesting results reasonably fast, while at the same time providing interesting expansions for my next research project. These expansions may include changes such as using a more sophisticated expert system, or perhaps using a more sophisticated asset pool, in order for resulting game levels to achieve a higher visual impact.

The DEA appears to have suggested levels that my participants responded to. Assuming that the users of the design prototype tool were aiming to create levels with high playability, a DEA appears to offer benefits, despite users' verbal responses in interviews that may appear contradictory to this notion. These responses may simply represent an initial reaction to a computational tool that seeks to help, rather than replace. The responses may also have been driven by a bias, and the user relying too much on perceived experience, as opposed to than factual expertise. Given that these suggestions are, however, based on assumptions and a small number of participants, these initial findings should be considered with care, and an emphasis placed on the fact that more research is needed.

While the quantitative data indicates a strong preference among participants for using the suggestions made by the agents, long standing expert designers in particular appeared to reject the idea that cognitive agents are helpful, even when shown evidence to support this. Expert game level designers seemed to believe that they needed to rely on their expertise more than 'metrics and algorithms'. However, this study cannot fully confirm this preconception. Using statistical data to triangulate these responses indicates a discrepancy between the perception of experts and the actual decisions made during design time. To confirm or reject this hypothesis, a study with a larger number of participants may be useful. Another possibility is to investigate whether this is simply a bias that can be found within creative industries, perhaps beyond the scope of game design, and specifically, within game level design. Perhaps this indicates that the often-cited Dunning-Kruger effect (Kruger & Dunning, 1999) does not necessarily hold true in certain domains? Dunning-Kruger is often associated with an unskilled individual unable to adequately estimate their own skill, which more often than not results in an overestimation of individual skills (Kruger & Dunning, 1999).

David Dunning (2011) clarifies, however, that this is not only true for unskilled individuals. According to Dunning, highly-skilled individuals also appear to underestimate their own skills (Kruger & Dunning, 1999). I involved highly-skilled individuals in my research and expected to see them underestimate their skills. However, I discovered a very contrasting result, where these experts seemingly overestimated their own abilities, emphasising their own experience and dismissing strong indications that they do rely on helpers. This may indicate two things: either these proclaimed experts were, in fact, not highly skilled, or they were unable to interpret their own abilities correctly. The former can be dismissed based on evidenced experience, and a track record of having successfully designed levels particular to this specific domain. The latter, however, may hold up following additional testing. A third possible explanation may simply be that these experts relied on their skills and experience, and as a result, selected levels that were very good (in terms of playability) in their opinion. However, this would only confirm that the agent-selected levels are indeed in line with what expert designers would choose to create, and thus confirm that the agents in this study were quite successful in selecting good playable levels.

6.2.3 Expert suggestions

All of my participants made a number of suggestions and gave very specific responses to a range of questions involving aspects of the game level design prototype. These suggestions and ideas need further investigation in some instances, given that they may be based on a fear to relinquish control of parts of the design process to a computational entity. In particular, the strong rejection pertaining to agent support suggests that fear to submit control could be the issue at hand. The question that needs to be asked here is whether suggestions made by experts can lead to improved workflows, more efficient design processes, and enable users to create more content for a wider audience, or if it has a more self-serving function, essentially, to reduce any computational support as much as possible, and to rely primarily on personal experience. If reliance on personal experience is the dominating factor here, this may indicate a bias, as discussed in the previous section. But it could also simply mean that experts do not want an increased computational support for their design tasks, and rather adhere to their existing tools and workflows. This question cannot be resolved without further investigation.

6.2.4 Expert responses in interviews also require triangulation

It would be an easy assumption to surmise that non-experts within a specific domain may provide responses that do not reflect reality as a result of their inexperience. This seems likely to be true. On the other hand, it can be assumed that experts are much more conscious of their actions and are able to assess their own views in a much more nuanced manner. My results, however, appear to contradict this assumption. Experts rejected the idea that the computational agents made any significant difference to their decision- making processes during design-time using my game level prototyping tool. However, the heavy inclination to not only divert their attention towards the agent-selected candidates, which was confirmed using eye tracking data, as well as the over-proportional selection of the agent-suggested candidates, appears to contradict the self-assessment of experts. Furthermore, while I cannot be sure that this finding holds true for all experts, due to the small number of participants included in this study, to me, it highlights the necessity for using a range of different data acquisition methods in order to be able to triangulate qualitative data.

I did not expect such strong indication of the value of triangulation, and never hypothesised it, but it is nonetheless an intriguing and happy accident. I believe it will be interesting to dedicate more time to this, perhaps within a domain where a much larger number of participants can be employed in a future study, to observe whether this is worth investigating further. It would also be interesting to see if it held true not only with a larger statistical sample, but also across different domains.

6.2.5 Educating users about autonomous systems

Understanding, at least at a basic level, how an autonomous system works has been established a crucial aspect of a design process in which human and computational agents undertake design tasks, particularly in those involving evolutionary systems. A recently published position paper introduces the concept of a triple-loop for design supported by autonomous tools (Seidel et al., 2018). The authors emphasise that simply considering the learning conducted by the machine, which results in a computational model that helps choosing the right actions in order to achieve a specific result, is simply not enough. They argue that the human designer also formulates a mental model that needs to be trained in order to achieve effective results when working with the support of autonomous tools. This human model includes an understanding of some of the inner workings of the autonomous

design tools, so that both systems can evolve their capabilities, and subsequently become a more effective system as a whole (Seidel et al., 2018). I believe that Seidel et al.'s findings support some of the approaches taken in this study. Educating the user, at least at a fundamental level, so that they understand how the autonomous tools may respond under certain circumstances, was a choice I made as part of my pre-participation briefs with participants. I agree with the article published by Seidel et al. (2018), and do not think that my brief explanation induced bias to my evaluations, because my participants were not aware of whether the computational agents were active or not.

6.3 Limitations

I have deliberately pointed to some of the limitations of this study throughout this thesis, as I believe that a progressive approach to highlighting the boundaries of this study, while also looking for potential future solutions, allowed me to take a critical view of the work presented here. The following sections unpack these limitations and point to possible solutions that can be explored in future work.

6.3.1 Sample size

The main limitation, in my opinion, was the relatively small sample size, and thus, the low number of participating game level designers. Access to game level designers is limited, and particularly considering that I needed them to engage with the eye tracking system. It can be argued that such a limitation could have been avoided by either including a larger geographical area, or by simply removing the eye tracking method from this study.

While I could have expanded my search for participants, for example, to Australia, which has a healthy game development industry and presumably, a number of game level designers, this was cost-prohibitive. If I am able to secure additional funding, I would certainly prefer to expand the recruitment range and increase the sample size.

The second option of removing the eye tracking method from this study is likely not ideal, given that this study established the necessity to triangulate interview responses with quantitative data, and since eye tracking in particular provided interesting insights into the behaviour of participants. Thus, if at all possible, I will attempt to include eye tracking again in future studies, and rather find sufficient funding for expanding the geographical area to conduct a larger study.

6.3.2 Rule-based Expert System

Due to the lengthy nature of a PhD and rapid development in the area of procedural content generation, neural network-based agents and the recent rise of deep neural networks, the two agents in this prototype may appear very traditional, and not particularly progressive. A rule-based expert system to capture human decision-making may be viewed as conservative when compared with deep learning for decision-making support systems. This can be viewed as a limitation of this study. I would argue, however, that the underlying principle of merging human and computational capacity in the form of different roles within a multi agent system, which in turn drives part of an interactive genetic algorithm, is an advancement of existing knowledge, and therefore, a contribution of this thesis. The contribution here lies in the addition of several computational agents to the system, following Kosorukoff's (2001) original proposal. The next step, in my view, is an additional agent based on a deep neural network, most likely a generative adversarial network, which is capable of generating new variations, based on a learned set of features (Giacomello et al., 2018). I believe that extending the existing system is reasonably straightforward and would allow for rapid prototyping of new agents, either replacing existing agents or simply adding new capabilities to the overall system. Testing the success or failure of these additions will be fast, assuming another set of evaluations with experienced game design experts can be conducted. But there are also a few important points to consider. An expert system is directly connected to the data derived from designers. A generative adversarial network is trained based on outcomes (the actual levels) of the design process. While there are numerous diagrams or images of levels available, which can be used to train the network, the number of levels that are considered very good and that match the popularity, and presumably the playability of the levels discussed by the experts that served as data for the cognitive model created in this study, are scant. Training a deep neural network with very little training data will potentially be very difficult, and a simpler approach, such as creating a rule-based system, is relatively easy to achieve. It is not an automated and domain-independent method, but it is also not bound to the use of large datasets, which is an advantage in applications that do not offer rich and readily available large sets of data.

6.3.3 Tracking individual agent performance

With the current setup of the design prototype, there is no provision to track the performance of individual agents. While the current results show some promise and indicate that the multi-agent system is a helpful addition to pure interactive evolutionary content generation, it is impossible to draw any conclusions linked to the contribution of the diversity agent and the DEA. This may appear to be a major flaw of this study. I would argue against this, however, because the diversity agent is heavily dependent on the DEA for making any decisions. Without the ranked top candidate selected by the DEA, there is no way for the diversity agent to find a top ranked candidate that has one different heuristic feature. The latter requires the former to finish the search first, and to publish the message that identifies the top ranks. Therefore, the diversity agent depends on another computational agent. As such, at least the statement that the designer expert agent seems to perform reasonably well and increases the speed of convergence towards a high fitness of the overall pool, is defendable. Furthermore, the claim that the designer expert agent helps to augment the human user by offering a fast track towards solutions that the user is in favour of, and which therefore improves the overall system performance, is also likely accurate, albeit additional research is needed to clearly support this.

6.3.4 Post-design play-testing

Furthermore, and this leads to a second limitation of this work, it will be beneficial to conduct broader play-testing with a large number of users, in order to establish the success of the resulting levels. Design experts may offer opinions and assumptions about the resulting levels that can be extrapolated from their responses; however, play-testing with the end user is viewed as the ultimate test and was also suggested by one of the participants in this study. The data generated via a large-scale playtest can also be fed back into the original generative system in the form of an agent, given that a large dataset of user inputs and actions can be used as training data for a deep neural network. This can ultimately lead to a system that improves over time, by capturing the designer's intent, as well as player actions.

6.4 Future Work

In the previous section, I discussed a number of limitations of this study. While I consider the sample size of the current research the most significant restriction, I will likely make a number

of adjustments to the overall methodology in future work, and not only increase the number of participants.

The above recommendations for future work are captured in this section.

6.4.1 Sample size

Two possible ways to increase the sample size, which has been identified as a possible issue of this study, are highlighted in the limitations section. Given the value that eye tracking shows as a means for triangulating interview responses, I will most certainly prefer to increase the sample by expanding the recruitment to other locations, which in turn requires an increase in research budget for future studies.

6.4.2 Exclusion agent

Two participants suggested enabling the user to move beyond simple parent/parent selections, and to also allow for excluding specific candidates from future breeding. This is an idea that I had worked on before even starting the evaluation, but in favour of a pure approach to the genetic algorithm, I decided not to use this agent. The focus of my thesis is on one of its main contributions, the designer expert agent, which acts as an evaluator in a procedural content generation system. The integration of an exclusion mechanic through the recording of user input, and feeding it into an additional agent is too close to Kosorukoff's (2001) human-based genetic algorithm, as it makes use of the suggested multi-role of human agents. However, I wanted to focus on the designer model rather than a sophisticated (and large) multi-agent system. Therefore, I decided not to use an exclusion agent in my prototype evaluation. Part of this decision was also my aim to preserve the simple use of the tool. Adding additional steps would have potentially increased complexity for the user, which may have added to user fatigue.

An exclusion agent kept track of candidates that the user selected as unsuitable, and that the user did not wish to use in any future generation. This was easily implemented by taking the designer agent's output, which reflects a highly desirable set of goals, and benchmarking the selected unwanted candidate against these goals. The largest resulting discrepancy would be a feature that was to be avoided in future populations, by allowing the exclusion agent to reject randomly selected candidates for the next population based on their undesirable features. The usefulness and performance of an exclusion agent is subject to future research.

6.4.3 Additional features

A number of feature suggestions were made by participants that can potentially improve the usability of the design tool and enable the designer to take an active role in some of the tasks currently handled by agents or operators of the genetic algorithm.

While some feature suggestions may only provide minor workflow improvements, a couple of points appear important, and should be considered in future work. These include visual aids that would likely enable the user to conduct a more efficient assessment of the candidates. Adding a grid is not a significant change, but given the known issue of user fatigue in interactive computational systems, it seems like a good suggestion to increase overall system performance and usability. Additionally, an overlay that shows how the agents assessed the levels may be useful, as it will help designers to understand what the autonomous system does, and there is evidence in other publications that this may be important in hybrid systems (Seidel et al., 2018). However, there is also the argument that increasing the cognitive load by increasing visual elements may have a negative impact on attention (see, for example Kiefer, Giannopoulos, Raubal & Duchowski, 2017), and lead to increased user fatigue. Based on previous experience with interactive evolutionary systems (Kruse & Connor, 2015), it may be worth investigating a system that offers additional visual information to the user, while at the same time includes a user preference agent that mitigates the risk of fatigue.

6.5 Conclusion

This thesis discusses a novel approach to FPS game level evaluation, based on a multi-agent system that includes the designer, rather that attempting to replace them. A novel DEA performs tasks based on traits that have been extracted from expert game level designer accounts, and that are subsequently transformed into a decision-support system. The system was evaluated and indicates promising results. Participants also provided some surprising interview responses that contradicted their own selections and differed to what they paid significant attention to. While the results need to be verified in future research using a larger sample size, the project is deemed a small success within a much larger trajectory.

7 References

Adams, E. (2013). *Fundamentals of Game Design* (3 edition). Berkeley, CA: New Riders.

Alroobaea, R., & Mayhew, P. J. (2014). How many participants are really enough for usability studies? *2014 Science and Information Conference*, 48–56. https://doi.org/10.1109/SAI.2014.6918171

Andersen, R. A., Bracewell, R. M., Barash, S., Gnadt, J. W., & Fogassi, L. (1990). Eye position effects on visual, memory, and saccade-related activity in areas LIP and 7a of macaque. *Journal of Neuroscience*, *10*(4), 1176–1196.

Anderson, C., Buchsbaum, D., Potter, J., & Bonabeau, E. (2008). Making Interactive Evolutionary Graphic Design Practical. In A. P. T. Yu, P. L. Davis, D. C. Baydar, & P. R. Roy (Eds.), *Evolutionary Computation in Practice* (pp. 125–141). Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-75771-9_6

Bakkes, S. C. J., Spronck, P. H. M., & van Lankveld, G. (2012). Player behavioural modelling for video games. *Entertainment Computing*, *3*(3), 71–79. https://doi.org/10.1016/j.entcom.2011.12.001

Bentley, P. (1999). *Evolutionary Design by Computers*. Morgan Kaufmann.

Beyer, H.-G., & Schwefel, H.-P. (2002). *Evolution Strategies - A Comprehensive Introduction*. *1*(1), 3–52. https://doi.org/10.1023/A:1015059928466

Boden, M. A. (1977). *Artificial intelligence and natural man* (Vol. 5). JSTOR.

Boden, M. A. (2006). *Mind as machine : a history of cognitive science*. Oxford : Clarendon Press ; New York : Oxford University Press, 2006. (City Campus Main Collection 153.09 BOD).

Bohil, C. J., & Biocca, F. A. (2007). *Cognitive Modeling of Video Game Players*. Retrieved from Mindlab website: http://mindlab.org/images/d/DOC1089.pdf

Bohnacker, H. (2012). *Generative design: visualize, program, and create with processing*. New York: Princeton Architectural Press.

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, *4*(1), 25–30. https://doi.org/10.1147/sj.41.0025

Brown, D. (2012). *The suspension of disbelief in videogames* (Brunel University School of Arts PhD Theses). Retrieved from http://v-scheiner.brunel.ac.uk/handle/2438/7457

Brustoloni, J. C. (1991). *Autonomous agents: characterization and requirements*. School of Computer Science, Carnegie Mellon University.

Bryman, A. (2012). *Social Research Methods*. OUP Oxford.

Bryman, A. (2016). *Social research methods*. Oxford : Oxford University Press, [2016]. (South Campus Main Collection 300.72 BRY).

Buchanan, B., Randall, D., & Feigenbaum, E. (2006). Expert Systems: A Perspective from Computer Science. *Cambridge Handbook of Expertise and Expert Performance*.

Candy, L. (2006, November). Practice Based Research Guide. Retrieved May 26, 2012, from http://www.scribd.com/doc/72480138/Practice-Based-Research-Guide

Charles, D., McNeill, M., McAlister, M., Black, M., Moore, A., Stringer, K., … Kerr, A. (2005). Player-Centred Game Design: Player Modelling and Adaptive Digital Games. *Proceedings of DiGRA 2005 Conference*. Presented at the Digital Games Research Association Conference, Vancouver, Canada. Retrieved from https://www.cp.eng.chula.ac.th/~vishnu/gameResearch/AI_november_2005/digra05.pdf

Charters, E. (2003). The use of think-aloud methods in qualitative research an introduction to think-aloud methods. *Brock Education Journal*, *12*(2).

Connor, A., Greig, T., & Kruse, J. (2018). Evolutionary generation of game levels. *EAI Endorsed Transactions on Creative Technologies*, *5*(15). Retrieved from http://eudl.eu/doi/10.4108/eai.10-4-2018.155857

Connor, A. M. (1996). *The synthesis of hybrid mechanisms using genetic algorithms (BL)* (Ph.D., Liverpool John Moores University (United Kingdom)). Retrieved from http://search.proquest.com/docview/301467506?pq-origsite=summon

Cook, M., & Colton, S. (2011). Multi-Faceted Evolution Of Simple Arcade Games. *CIG*, 289–296.

Cook, M., Colton, S., & Gow, J. (2017). The ANGELINA Videogame Design System—Part II. *IEEE Transactions on Computational Intelligence and AI in Games*, *9*(3), 254–266.

Creswell, J. W. (2014). *Research design : qualitative, quantitative, and mixed methods approaches*. Thousand Oaks : SAGE Publications, [2014]. (North Campus Main Collection 300.72 CRE).

Dawkins, R. (1986). *The blind watchmaker: why the evidence of evolution reveals a universe without design*. New York: Norton.

Dawson, M. R. W. (2003). *Minds and Machines: Connectionism and Psychological Modeling* (1 edition). Malden, MA: Wiley-Blackwell.

Dean, R. T., & Smith, H. (2009). *Practice-led Research, Research-led Practice in the Creative Arts* (1st ed.). Edinburgh: Edinburgh University Press.

Desurvire, H., Caplan, M., & Toth, J. A. (2004). Using Heuristics to Evaluate the Playability of Games. *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, 1509–1512. https://doi.org/10.1145/985921.986102

Duchowski, A. T. (2007). *Eye tracking methodology : theory and practice*. London : Springer, c2007.

Duda, R. O., & Shortliffe, E. H. (1983). Expert systems research. *Science*, *220*(4594), 261–268.

Dunning, D. (2011). Chapter five - The Dunning–Kruger Effect: On Being Ignorant of One's Own Ignorance. In J. M. Olson & M. P. Zanna (Eds.), *Advances in Experimental Social Psychology* (Vol. 44, pp. 247–296). https://doi.org/10.1016/B978-0-12-385522-0.00005-6

Edwards, R. (2006, May). The Economics of Game Publishing - IGN. Retrieved February 23, 2019, from IGN website: https://www.ign.com/articles/2006/05/06/the-economics-of-game-publishing

Einhäuser, W., Schumann, F., Bardins, S., Bartl, K., Böning, G., Schneider, E., & König, P. (2007). Human eye-head co-ordination in natural exploration. *Network: Computation in Neural Systems*, *18*(3), 267–297.

Elshoff, J. L., & Hulina, P. T. (1970). The Binary Floating Point Digital Differential Analyzer. *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, 369–376. https://doi.org/10.1145/1478462.1478516

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis : verbal reports as data*. Cambridge, Mass. : MIT Press, [1993]. (City Campus Main Collection 006.3 ERI).

Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, *35*(3), 379–383.

Feigenbaum, E. A. (1980). *Expert systems in the 1980s*. Retrieved from https://stacks.stanford.edu/file/druid:vf069sz9374/vf069sz9374.pdf

Feigenbaum, E. A., & Feldman, J. (1963). *Computers and Thought*. New York, NY, USA: McGraw-Hill, Inc.

Ferro, L. S., Walz, S. P., & Greuter, S. (2013). Towards Personalised, Gamified Systems: An Investigation into Game Design, Personality and Player Typologies. *Proceedings of The 9th Australasian Conference on Interactive Entertainment: Matters of Life and Death*, 7:1–7:6. https://doi.org/10.1145/2513002.2513024

Ferrucci, D., Levas, A., Bagchi, S., Gondek, D., & Mueller, E. T. (2013). Watson: Beyond Jeopardy! *Artificial Intelligence*, *199–200*, 93–105. https://doi.org/10.1016/j.artint.2012.06.009

Finlay, J., Connor, A. M., & Pears, R. (2011). Mining Software Metrics from Jazz. *2011 9th International Conference on Software Engineering Research, Management and Applications (SERA)*, 39–45. https://doi.org/10.1109/SERA.2011.40

Fisher, G. (2014). *Blender 3D Basics: Second Edition* (2 edition). Birmingham, UK: Packt Publishing - ebooks Account.

Fogel, D. B. (2006). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Wiley.

Franklin, S. (1997). *Artificial Minds*. MIT Press.

Franklin, S., & Graesser, A. (1996). *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*. 21–35. Springer-Verlag.

Fraser, A., & Burnell, D. (1970). Computer models in genetics. *Computer Models in Genetics.* Retrieved from https://www.cabdirect.org/cabdirect/abstract/19711604313

Freyermuth, G. S. (2015). *Games, Game Design, Game Studies: An Introduction*. transcript Verlag.

Fullerton, T. (2008). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games* (2 edition). Amsterdam ; Boston: Morgan Kaufmann.

Fum, D., Missier, F. D., & Stocco, A. (2007). The Cognitive Modeling of Human Behavior: Why a Model is (Sometimes) Better Than 10,000 Words. *Cogn. Syst. Res.*, *8*(3), 135–142. https://doi.org/10.1016/j.cogsys.2007.07.001

Galanter, P. (2012). Computational Aesthetic Evaluation: Past and Future. In J. McCormack & M. d'Inverno (Eds.), *Computers and Creativity* (pp. 255–293). Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-31727-9_10

Galuzin, A. (2016). *Preproduction Blueprint: How to Plan Game Environments and Level Designs* (2 edition). North Charleston, South Carolina: CreateSpace Independent Publishing Platform.

Gaudiosi, J. (2016, April 6). These Are The Most Popular ESports Games On Twitch. Retrieved November 6, 2017, from Fortune website: http://fortune.com/2016/04/06/most-popular-esports-games-on-twitch/

Giacomello, E., Lanzi, P. L., & Loiacono, D. (2018). *DOOM Level Generation using Generative Adversarial Networks*. Retrieved from https://arxiv.org/abs/1804.09154

Given, L. (2008). *The SAGE Encyclopedia of Qualitative Research Methods*. https://doi.org/10.4135/9781412963909

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning* (1 edition). Reading, Mass: Addison-Wesley Professional.

Goldberg, J. H., & Kotval, X. P. (1999). Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics*, *24*(6), 631–645. https://doi.org/10.1016/S0169-8141(98)00068-7

Greenberg, I., Xu, D., & Kumar, D. (2013). *Processing: Creative Coding and Generative Art in Processing 2* (2 edition). Berkeley, Calif.; London: friendsofED.

Greuter, S., Parker, J., Stewart, N., & Leach, G. (2003). Real-time procedural generation ofpseudo infinite'cities. *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, 87–ff. ACM.

Gu, Z., Xi Tang, M., & Frazer, J. H. (2006). Capturing aesthetic intention during interactive evolution. *Computer-Aided Design*, *38*(3), 224–237. https://doi.org/10.1016/j.cad.2005.10.008

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.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor: University of Michigan Press.

Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, *15*(9), 1277–1288. https://doi.org/10.1177/1049732305276687

Hwang, W., & Salvendy, G. (2010). Number of people required for usability evaluation: the 10 +/- 2 rule. *Communications of the ACM*, *53*(5), 130–133.

Jackson, P. (1998). *Introduction to Expert Systems* (3rd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Järvinen, A. (2007). Introducing Applied Ludology: Hands-on Methods for Game Studies. *DiGRA Conference*. Presented at the Tokyo, Japan. Tokyo, Japan.

Järvinen, A. (2009). *Games Without Frontiers: Methods for Game Studies and Design*. Tampere, Finland: VDM, Verlag Dr. Müller.

Jenkins, H. (2004). Game Design as Narrative Architecture. *Computer*, *44*, 53.

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

Juul, J. (2011). *Half-Real: Video Games between Real Rules and Fictional Worlds*. MIT Press.

Keeton, W. T. (1996). *Biological Science* (6 edition). New York: W W Norton & Co Inc.

Kelly, G., & McCabe, H. (2007). *Citygen: An interactive system for procedural city generation*. Presented at the GDTW, Liverpool, UK. Retrieved from http://procedural.googlecode.com/svn-history/r121/trunk/articles_cities/citygen_gdtw07.pdf

Kiefer, P., Giannopoulos, I., Raubal, M., & Duchowski, A. (2017). Eye tracking for spatial research: Cognition, computation, challenges. *Spatial Cognition & Computation*, *17*(1–2), 1–19. https://doi.org/10.1080/13875868.2016.1254634

Kosorukoff, A. (2001). Human based genetic algorithm. *2001 IEEE International Conference on Systems, Man, and Cybernetics*, *5*, 3464–3469 vol.5. https://doi.org/10.1109/ICSMC.2001.972056

Koster, R. (2018, January 17). The cost of games. Retrieved February 23, 2019, from Gamasutra website: https://www.gamasutra.com/blogs/RaphKoster/20180117/313211/The_cost_of_games.php

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (1 edition). Cambridge, Mass: A Bradford Book.

Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., & Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence* (1st ed. 2003. Corr. 2nd printing edition). Norwell, Mass: Springer.

Kruger, J., & Dunning, D. (1999). Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of Personality and Social Psychology*, *77*(6), 1121–1134. https://doi.org/10.1037/0022-3514.77.6.1121

Kruse, J. (2014). *Interactive evolutionary computation in design applications for virtual worlds* (Thesis, Auckland University of Technology). Retrieved from http://aut.researchgateway.ac.nz/handle/10292/8593

Kruse, J., & Connor, A. M. (2015). Multi-agent evolutionary systems for the generation of complex virtual worlds. *EAI Endorsed Transactions on Creative Technologies*.

Lally, A., Prager, J. M., McCord, M. C., Boguraev, B. K., Patwardhan, S., Fan, J., … Chu-Carroll, J. (2012). Question analysis: How Watson reads a clue. *IBM Journal of Research and Development*, *56*(3.4), 2:1-2:14. https://doi.org/10.1147/JRD.2012.2184637

Lavrakas, P. (2008). Conversational Interviewing. In *Encyclopedia of Survey Research Methods*. https://doi.org/10.4135/9781412963947.n107

Lawson, B. (1997). *How designers think : the design process demystified*. Oxford ; Boston : Architectural Press, [1997]. (City Campus Main Collection 745.4 LAW).

Liapis, A., Yannakakis, G. N., & Togelius, J. (2013a). Sentient Sketchbook: Computer-aided game level authoring. *FDG*, 213–220.

Liapis, A., Yannakakis, G. N., & Togelius, J. (2013b). Towards a Generic Method of Evaluating Game Levels. *AIIDE*.

Liapis, A., Yannakakis, G. N., & Togelius, J. (2014). Computational game creativity. *Proceedings of the Fifth International Conference on Computational Creativity*, 285–292. Retrieved from http://www.itu.dk/people/anli/research/computational_game_creativity.pdf

Macklin, C., & Sharp, J. (2016). *Games, Design and Play: A detailed approach to iterative game design* (1 edition). Boston, MA ; San Francisco, CA: Addison-Wesley Professional.

Marks, S. (2006). *Evolving autonomous locomotion of virtual characters in a simulated physical environment via neural networks and evolutionary strategies*. University of Applied Sciences Gelsenkirchen, Gelsenkirchen.

Mitchell, B. L. (2012). *Game Design Essentials* (1 edition). Indianapolis, Indiana, USA: Sybex.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.

Moogk, D. R. (2012). Minimum Viable Product and the Importance of Experimentation in Technology Startups. *Technology Innovation Management Review*, (March 2012: Technology Entrepreneurship), 23–26.

Mühlenbein, H. (1992). *How Genetic Algorithms Really Work: Mutation and Hillclimbing.* Berlin: Springer.

Münch, J., Fagerholm, F., Johnson, P., Pirttilahti, J., Torkkel, J., & Jäarvinen, J. (2013). Creating Minimum Viable Products in Industry-Academia Collaborations. In B. Fitzgerald, K. Conboy, K. Power, R. Valerdi, L. Morgan, & K.-J. Stol (Eds.), *Lean Enterprise Software and Systems* (pp. 137–151). Berlin: Springer Berlin Heidelberg.

Murdock, K. (2017). *Autodesk Maya 2018 Basics Guide* (Pap/Psc edition). Mission, KS, USA: SDC Publications.

Museth, K. (2013). VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)*, *32*(3), 27.

Museth, K. (2014). Hierarchical digital differential analyzer for efficient ray-marching in openvdb. *ACM SIGGRAPH 2014 Talks*, 40. ACM.

Musil, J., Schweda, A., Winkler, D., & Biffl, S. (2010). Improving Video Game Development: Facilitating Heterogeneous Team Collaboration through Flexible Software Processes. In A. Riel, R. O'Connor, S. Tichkiewitch, & R. Messnarz (Eds.), *Systems, Software and Services Process Improvement* (pp. 83–94). Springer Berlin Heidelberg.

Nacke, L., Drachen, A., Kuikkaniemi, K., Niesenhaus, J., Korhonen, H. J., Hoogen, W. M. van den, … Kort, Y. A. W. de. (2009). *Playability and Player Experience Research*. Presented at the Proceedings of DiGRA 2009: Breaking New Ground: Innovation in Games, Play, Practice and Theory. Retrieved from http://www.diva-portal.org/smash/record.jsf?pid=diva2:835637

Nacke, L. E., & Lindley, C. A. (2010). Affective Ludology, Flow and Immersion in a First- Person Shooter: Measurement of Player Experience. *ArXiv:1004.0248 [Cs]*. Retrieved from http://arxiv.org/abs/1004.0248

Nagle, A., Wolf, P., & Riener, R. (2016). Towards a system of customized video game mechanics based on player personality: Relating the Big Five personality traits with difficulty adaptation in a first-person shooter game. *Entertainment Computing*, *13*(Supplement C), 10–24. https://doi.org/10.1016/j.entcom.2016.01.002

Negnevitsky, M. (2004). *Artificial Intelligence: A Guide to Intelligent Systems* (2nd ed.). Addison-Wesley.

Nickerson, R. S. (1998). Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, *2*(2), 175.

Okita, A. (2015). *Learning C♯ programming with Unity 3D*. Boca Raton : CRC Press, [2015].

Okun, J. A., & Zwerman, S. (Eds.). (2010). *The VES Handbook of Visual Effects: Industry Standard VFX Practices and Procedures* (1st ed.). Burlington, MA: Focal Press.

Olson, G. M., Duffy, S. A., & Mack, R. L. (2018). Thinking-Out-Loud as a Method for Studying 11 Real-Time Comprehension Processes. *New Methods in Reading Comprehension Research*, 253.

Ølsted, P. T., Ma, B., & Risi, S. (2015). Interactive evolution of levels for a competitive multiplayer FPS. *2015 IEEE Congress on Evolutionary Computation (CEC)*, 1527–1534. https://doi.org/10.1109/CEC.2015.7257069

Paavilainen, J., Korhonen, H., & Saarenpää, H. (2012). Comparing two playability heuristic sets with expert review method: A case study of mobile game evaluation. In *Media in the ubiquitous era: Ambient, social and gaming media* (pp. 29–52). IGI Global.

Parish, Y. I. H., & Müller, P. (2001). Procedural Modeling of Cities. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 301–308. https://doi.org/10.1145/383259.383292

Pearson, M. (2011). *Generative Art* (1 edition). Shelter Island, NY : London: Manning Publications.

Pentland, A., & Liu, A. (1999). Modeling and prediction of human behavior. *Neural Computation*, *11*(1), 229–242.

Pimpale, P., & Bhande, N. (2007, December). *Genetic Algorithms Made Easy*. Retrieved from http://www.slideshare.net/pbpimpale/genetic-algorithms-200688

Pinelle, D., Wong, N., & Stach, T. (2008). Heuristic evaluation for games: usability principles for video game design. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1453–1462. Retrieved from http://dl.acm.org/citation.cfm?id=1357282

Prager, J. M., Brown, E. W., & Chu-Carroll, J. (2012). Special Questions and techniques. *IBM Journal of Research and Development*, *56*(3.4), 11:1-11:13. https://doi.org/10.1147/JRD.2012.2187392

Prieto-Díaz, R. (1990). Domain analysis: An introduction. *ACM SIGSOFT Software Engineering Notes*, *15*(2), 47–54.

Raffe, W. L., Zambetta, F., Li, X., & Stanley, K. O. (2015). 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

Renner, G., & Ekárt, A. (2003). Genetic algorithms in computer aided design. *Computer-Aided Design*, *35*(8), 709–726. https://doi.org/10.1016/S0010-4485(03)00003-4

RStudio Team. (2015). *RStudio: Integrated Development for R*. Retrieved from http://www.rstudio.com/

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, *5*(1), 96–101. https://doi.org/10.1109/72.265964

Russell, S. J., & Norvig, P. (1994). *Artificial intelligence: a modern approach* (1st ed). Upper Saddle River, N.J: Prentice Hall.

Russell, S. J., & Norvig, P. (2003). *Artificial intelligence: a modern approach* (2nd ed). Upper Saddle River, N.J: Prentice Hall.

Sadilek, A. (2012). *Modeling human behavior at a large scale*. University of Rochester.

Saldaña, J. (2012). *The Coding Manual for Qualitative Researchers* (2 edition). Los Angeles: SAGE Publications Ltd.

Salen, K., & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. MIT Press.

Sánchez, J. L. G., Simarro, F. M., Zea, N. P., & Vela, F. L. G. (2009). Playability as Extension of Quality in Use in Video Games. *I-USED*.

Saygin, A. P., Cicekli, I., & Akman, V. (2000). Turing Test: 50 Years Later. *Minds and Machines*, *10*(4), 463–518. https://doi.org/10.1023/A:1011288000451

Schatz, E. (2017, June 27). Defining Environment Language for Video Games. Retrieved February 23, 2019, from 80lv website: https://80.lv/articles/defining-environment-language-for-video-games/

Schell, J. (2014). *The Art of Game Design: A Book of Lenses, Second Edition* (2 edition). Boca Raton, Florida, USA: A K Peters/CRC Press.

Schmettow, M. (2012). Sample size in usability studies. *Commun. ACM*, *55*(4), 64–70.

Seidel, S., Berente, N., Lindberg, A., Lyytinen, K., & Nickerson, J. V. (2018). Autonomous Tools and Design: A Triple-loop Approach to Human-machine Learning. *Commun. ACM*, *62*(1), 50–57. https://doi.org/10.1145/3210753

Shaker, N., Shaker, M., Abu-Abdallah, I., Al-Zengi, M., & Sarhan, M. H. (2013). A Quantitative Approach for Modelling and Personalizing Player Experience in First-Person Shooter Games. *UMAP 2013 Extended Proceedings*, *997*. https://doi.org/urn:nbn:de:0074-997-7

Shaker, N., Smith, G., & Yannakakis, G. N. (2016). Evaluating content generators. In N. Shaker, J. Togelius, & M. J. Nelson (Eds.), *Procedural Content Generation in Games* (pp. 215–224). https://doi.org/10.1007/978-3-319-42716-4_12

Shaker, N., Togelius, J., & Nelson, M. J. (2016a). Fractals, noise and agents with applications to landscapes. In N. Shaker, J. Togelius, & M. J. Nelson, *Procedural Content Generation in Games* (pp. 57–72). https://doi.org/10.1007/978-3-319-42716-4_4

Shaker, N., Togelius, J., & Nelson, M. J. (2016b). *Procedural Content Generation in Games* (1st ed. 2016 edition). Springer.

Shoham, Y. (1993). Agent-oriented Programming. *Artificial Intelligence*, *60*(1), 51–92. https://doi.org/10.1016/0004-3702(93)90034-9

Short, T. X., & Adams, T. (Eds.). (2017). *Procedural Generation in Game Design* (1 edition). Boca Raton, USA: Routledge.

Sims, K. (1992). Interactive evolution of dynamical systems. In *Proceedings of the First European Conference on Artificial Life*. *Toward a practice of autonomous systems* (pp. 171–178).

Smith, D. C., Cypher, A., & Spohrer, J. (1994). KidSim: Programming Agents Without a Programming Language. *Commun. ACM*, *37*(7), 54–67. https://doi.org/10.1145/176789.176795

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

Sprague, R. E. (1952). Technical Developments: Fundamental Concepts of the Digital Differential Analyzer Method of Computation. *Mathematical Tables and Other Aids to Computation*, *6*(37), 41–49. https://doi.org/10.2307/2002747

Summerville, A., Mariño, J. R., Snodgrass, S., Ontañón, S., & Lelis, L. H. (2017). Understanding mario: an evaluation of design metrics for platformers. *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 8. Hyannis, MA, USA: ACM.

Takagi, H. (1998). Interactive evolutionary computation: System optimization based on human subjective evaluation. *IEEE Int. Conf. on Intelligent Engineering Systems (INES'98)*, 17–19. Retrieved from http://pdf.aminer.org/000/305/069/discrete_fitness_values_for_improving_human_interface_in_an_interactive.pdf

Takagi, H. (2001). Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, *89*(9), 1275–1296. https://doi.org/10.1109/5.949485

Takagi, H., & Iba, H. (2005). Interactive Evolutionary Computation. *New Generation Computing*, *23*(2), 113–114.

Thompson, J., Berbank-Green, B., & Cusworth, N. (2007). *Game Design: Principles, Practice, and Techniques - The Ultimate Guide for the Aspiring Game Designer* (1 edition). Hoboken, N.J, USA: Wiley.

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. N. (2011). What is Procedural Content Generation?: Mario on the Borderline. *Proceedings of the 2Nd International Workshop on Procedural Content Generation in Games*, 3:1–3:6. https://doi.org/10.1145/2000919.2000922

Togelius, J., & Shaker, N. (2016). The search-based approach. In N. Shaker, J. Togelius, & M. J. Nelson, *Procedural Content Generation in Games* (pp. 17–30). https://doi.org/10.1007/978-3-319-42716-4_2

Togelius, J., Shaker, N., & Dormans, J. (2016). Grammars and L-systems with applications to vegetation and levels. In N. Shaker, J. Togelius, & M. J. Nelson, *Procedural Content Generation in Games* (pp. 73–98). https://doi.org/10.1007/978-3-319-42716-4_5

Togelius, J., Shaker, N., & Nelson, M. J. (2016). Introduction. In N. Shaker, J. Togelius, & M. J. Nelson (Eds.), *Procedural Content Generation in Games* (pp. 1–15). https://doi.org/10.1007/978-3-319-42716-4_1

Togelius, J., Shaker, N., & Yannakakis, G. N. (2013). Active player modelling. *ArXiv Preprint ArXiv:1312.2936*. Retrieved from http://arxiv.org/abs/1312.2936

Toy, M., Wichman, G., Arnold, K., & Lane, J. (1980). Rogue. *Computer Science Research Group, UC Berkeley*.

Turing, A. M. (1950). Computing Machinery and Intelligence. In R. Epstein, G. Roberts, & G. Beber (Eds.), *Parsing the Turing Test* (pp. 23–65). Retrieved from http://link.springer.com/chapter/10.1007/978-1-4020-6710-5_3

Vaishnavi, V. (2008). *Design science research methods and patterns: innovating information and communication technology*. Boca Raton: Auerbach Publications.

Vaismoradi, M., Turunen, H., & Bondas, T. (2013). Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. *Nursing & Health Sciences*, *15*(3), 398–405. https://doi.org/10.1111/nhs.12048

Verzani, J. (2011). *Getting started with RStudio*. O'Reilly Media, Inc.

Vygotskiĭ. (2012). *Thought and language*. Massachusetts, PA, USA: MIT Press.

Warman, P. (2017, February 14). Global Esports Market Report. Retrieved November 6, 2017, from Newzoo website: https://newzoo.com/insights/articles/esports-revenues-will-reach-696-million-in-2017/

Wolcott, H. F. (1990). *Writing Up Qualitative Research*. Newbury Park, Calif: SAGE Publications, Inc.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems* (2nd Edition edition). Chichester, U.K: John Wiley & Sons.

Wooldridge, M., & Jennings, N. R. (1995). Agent theories, architectures, and languages: A survey. In M. J. Wooldridge & N. R. Jennings (Eds.), *Intelligent Agents* (pp. 1–39). Retrieved from http://link.springer.com/chapter/10.1007/3-540-58855-8_1

Wright, T., Boria, E., & Breidenbach, P. (2002). Creative player actions in FPS online video games: Playing Counter-Strike. *Game Studies*, *2*(2), 103–123.

Yannakakis, G. N., Spronck, P., Loiacono, D., & André, E. (2013). Player Modeling. *Artificial and Computational Intelligence in Games*, *6*, 45–59.

Yannakakis, G. N., & Togelius, J. (2011). Experience-driven procedural content generation. *Affective Computing, IEEE Transactions On*, *2*(3), 147–161.

Zhong, W., Liu, J., Xue, M., & Jiao, L. (2004). A multiagent genetic algorithm for global numerical optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions On*, *34*(2), 1128–1141.

Zhu, M., Zhao, F., Fang, X., & Moser, C. (2017). Developing Playability Heuristics Based on Nouns and Adjectives from Online Game Reviews. *International Journal of Human–Computer Interaction*, *33*(3), 241–253. https://doi.org/10.1080/10447318.2016.1240283

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2009). Human Behavior Modeling with Maximum Entropy Inverse Optimal Control. *AAAI Spring Symposium: Human Behavior Modeling*, 92.

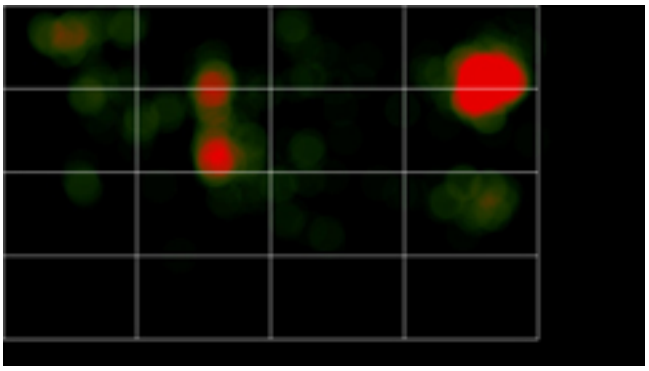
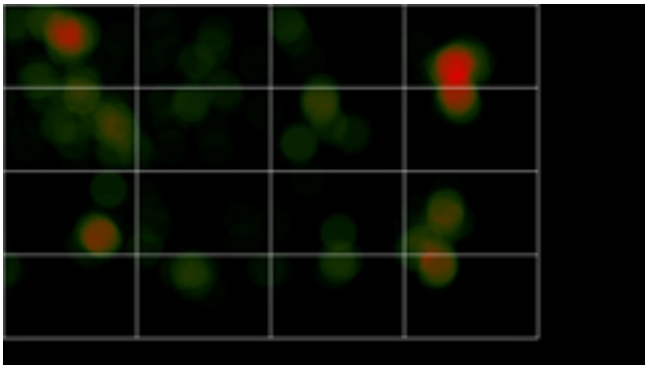
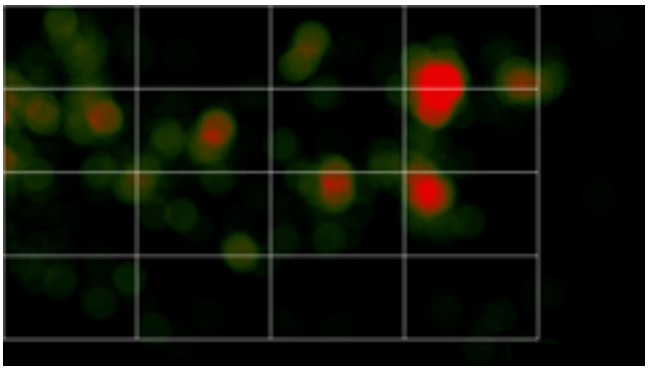
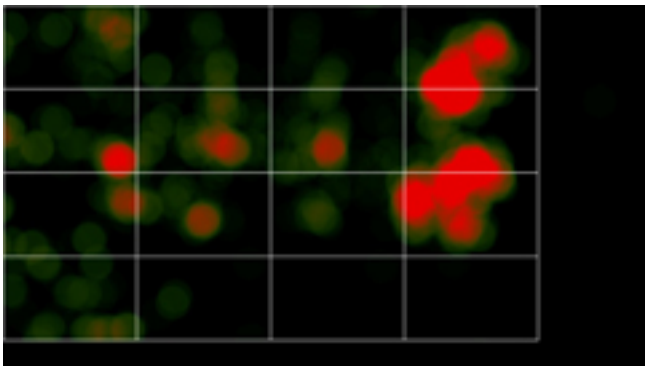
Appendix

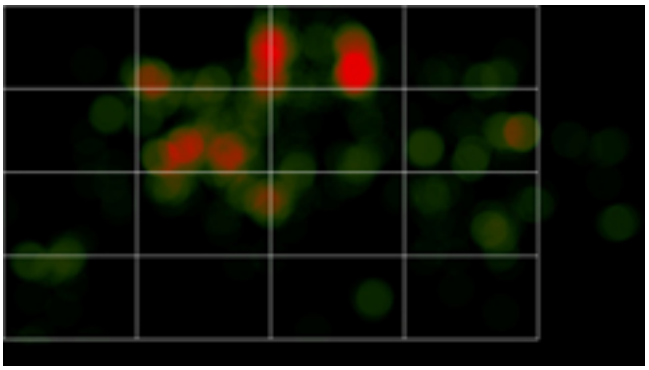
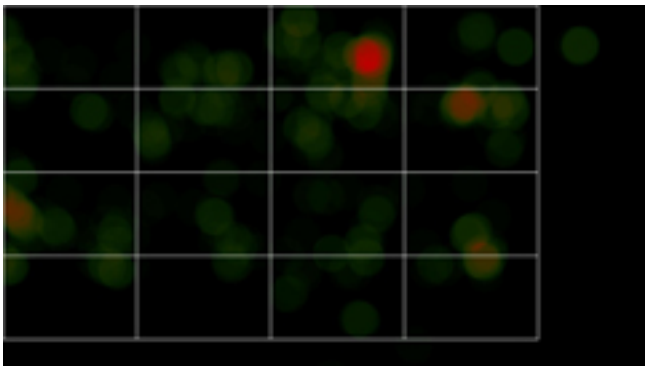
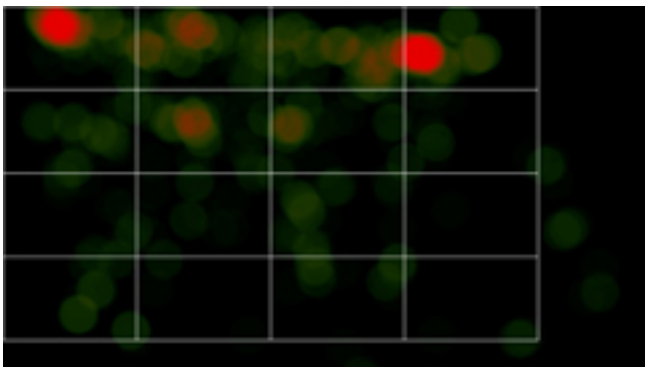
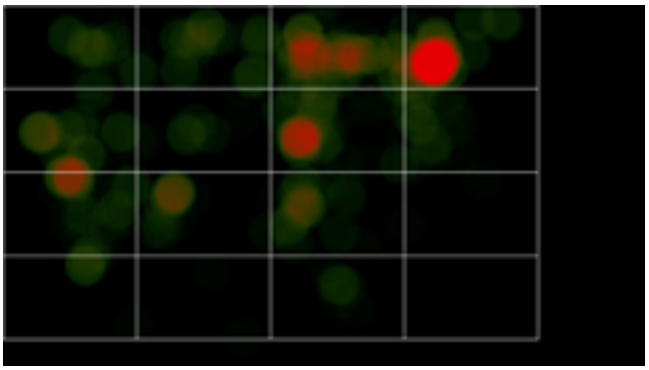
A. Selection data as table

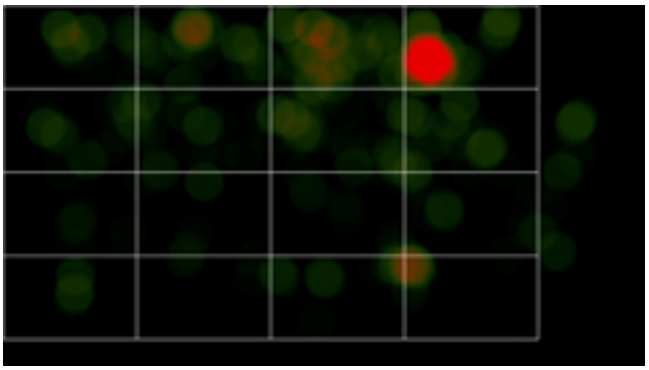
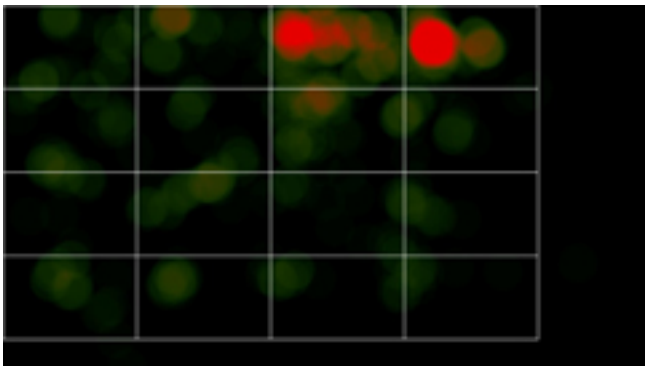
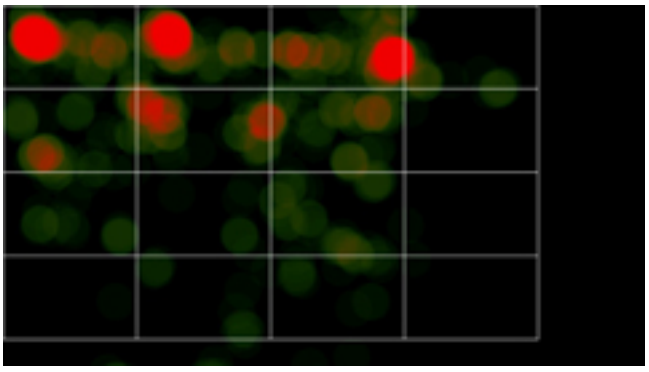
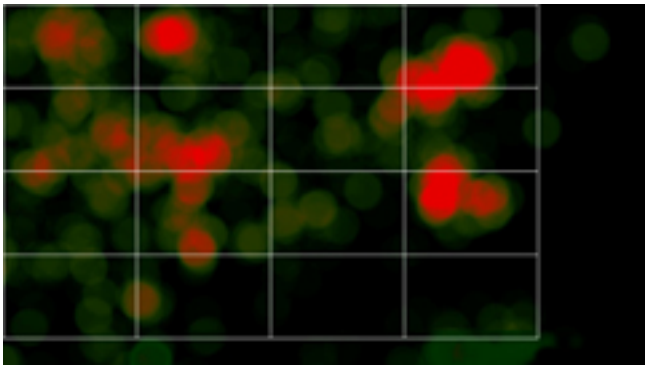
Table 1 - Example of numerical selection representation

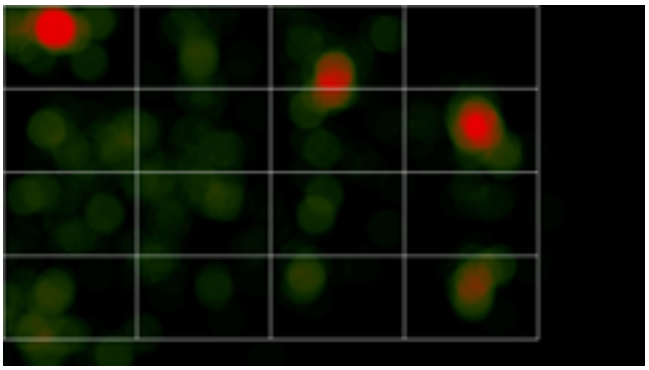
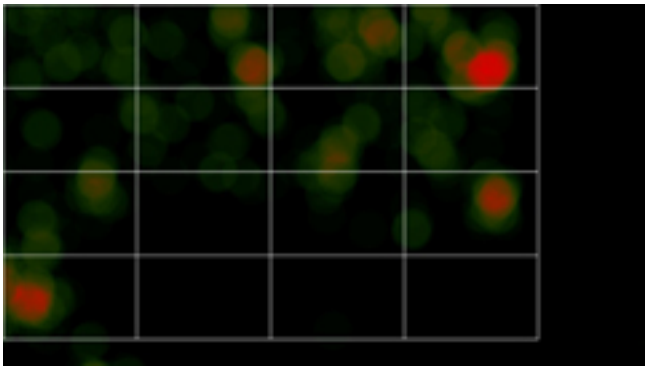
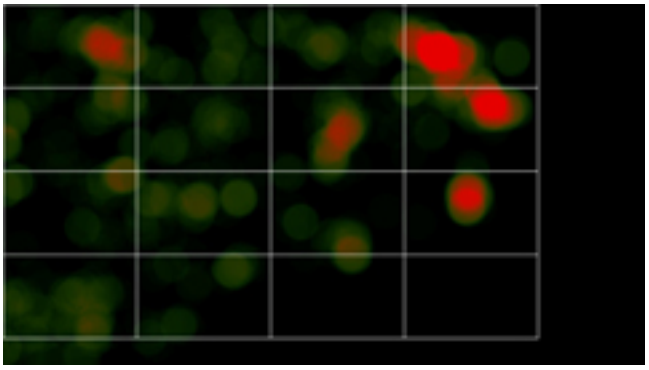
Generation	Parent1	Parent2
0	15	9
1	0	6
2	15	4
3	10	4
4	0	8
5	13	10
6	13	4
7	9	0
8	12	1
9	11	4
10	4	1
11	0	11
12	7	1
13	11	1
14	2	9
15	1	0
16	1	3
17	1	8
18	1	9
19	8	0
20	0	2
21	1	9
22	0	9
23	7	10
24	6	0
25	1	1
26	1	5
27	8	0
28	1	2

B. Eye tracking heatmaps (agents active)

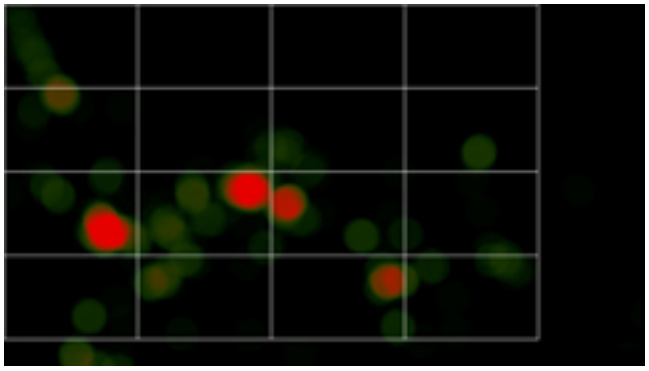
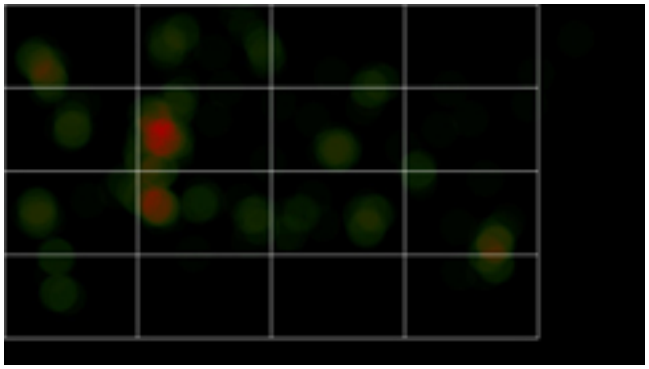
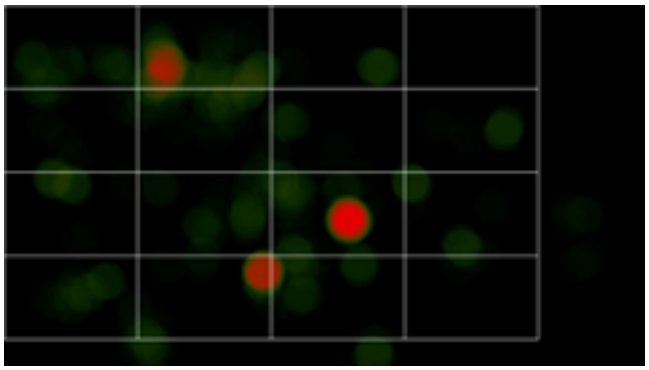
1 	Eye tracking result shows a strong tendency towards the second agent-based candidate (C4). There is also a moderate bias towards one of the elitist parents (C2) and a random candidate (C6).
2 	A strong indication in C4, a moderate indication that C1 (parent) was of interest and the expected browsing of the random candidates over time.
3 	A strong marker on C4, but also moderate to strong indications of some random candidates. The eyes of the participant spent most of their time on C4 though.
4 	Strong indications on C4 and C12 as well as C5. It would be a stretch to claim that this result shows a bias towards the agent-based candidates in C3 and C4.

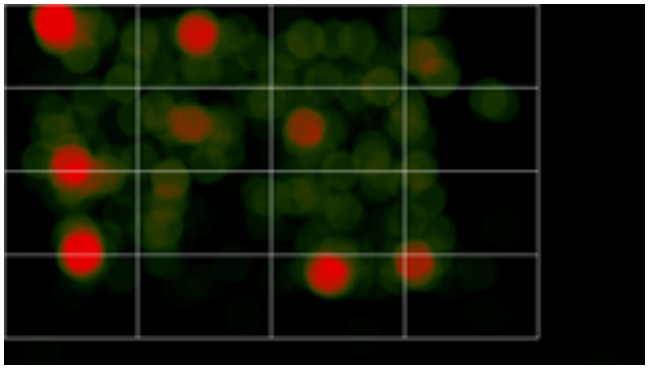
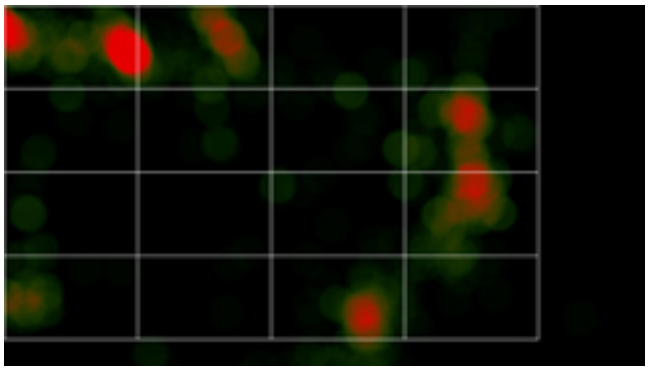
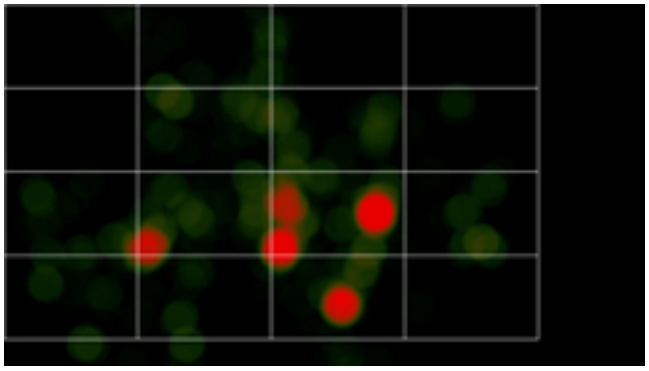
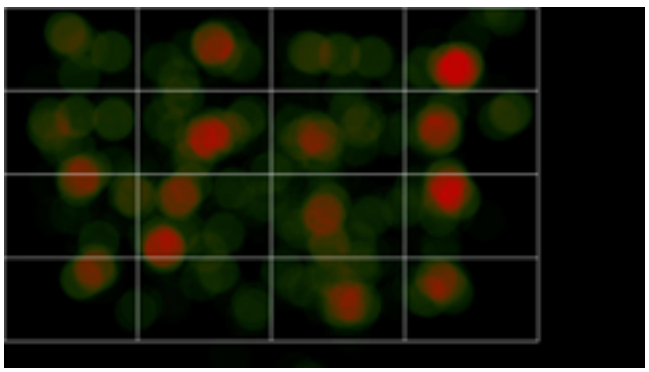
5 	A strong bias towards C3 and even C2. Otherwise a distribution across most of the random cells. C1 seems to be almost ignored.
6 	The strongest indication is in C3. Otherwise a reasonably even distribution, but very sparse. This indicates a short run.
7 	Strong indication in C1 and C4. Shows a preference for agent-based candidate and previous parents. Overall a bias towards the top row C1-C4). Also a few stray blobs next to C12 which indicates the use of the three-dimensional depiction of the parents.
8 	A very strong indication on C4, plus moderate bias towards C3, C7 and (probably) C9 – this light red blob sits on the border to C5.

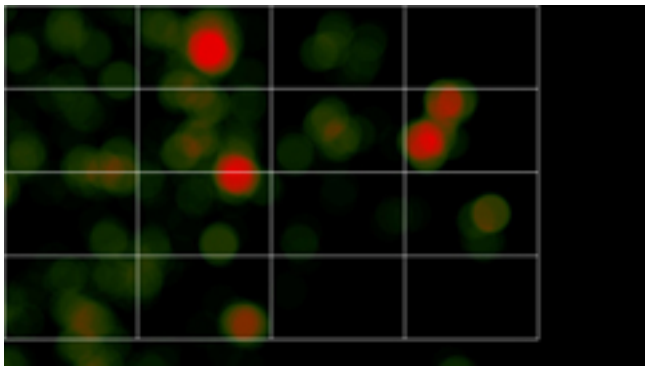
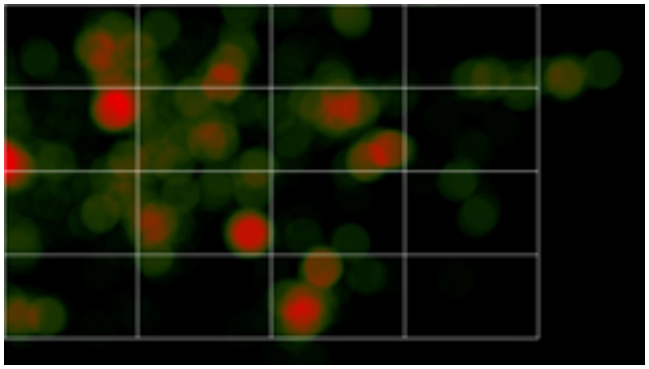
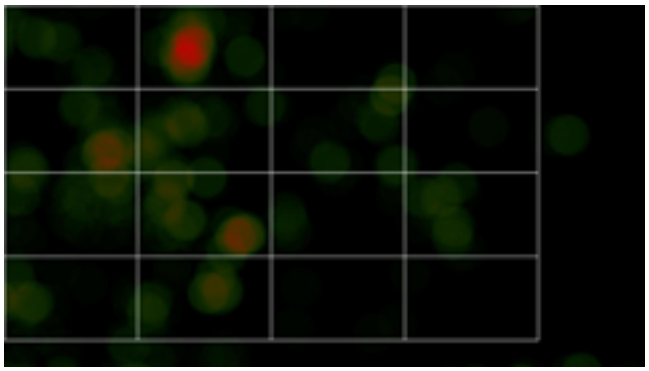
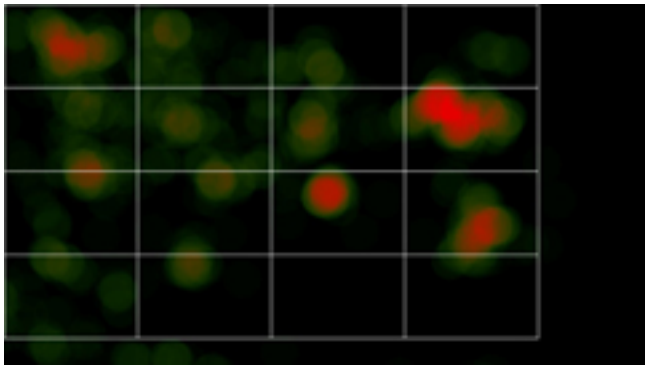
9 	A strong marker on C4 and moderate activity in C2 and C3 as well as C15/C16 border. There is an indication that this participant made use of the three-dimensional view as well (next to C8). Observation protocol notes that the participant called C4 very interesting in a number of runs.
10 	Two strong indications on C3 and C4, with a very even distribution across the other candidates. This participant showed a strong tendency to consider agent-based candidates. Observation protocol entries confirm this.
11 	There are strong indications in C1 – C3. While there is virtually no consideration of C4, this heatmap shows a strong bias towards the elitist parents and an agent-based suggestion.
12 	This heatmap shows several tiles with a strong signature. But this only means that this was a very long run. Given that a number of tiles have large red areas, for example C2, C4, C6, C12, it would be false to point to the agent-based solutions here. However, observation protocols show that the participant was looking for “something different” and want to “push the algorithm into a different direction”.

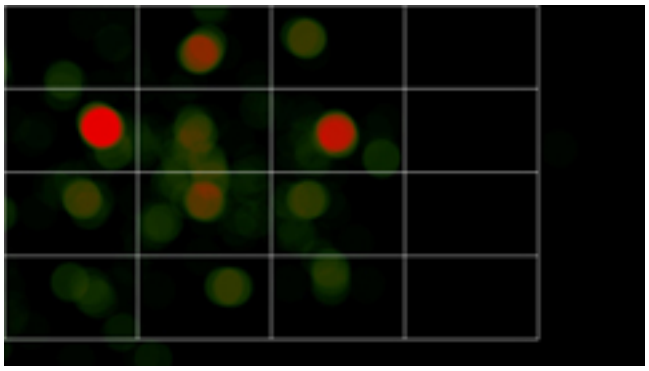
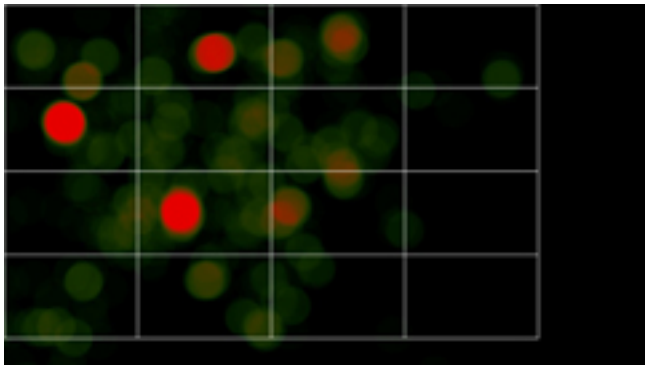
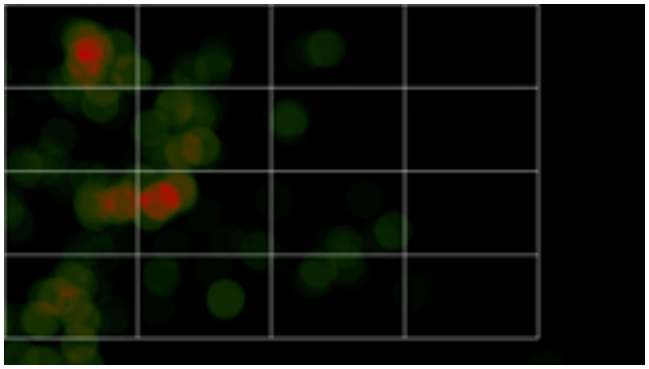
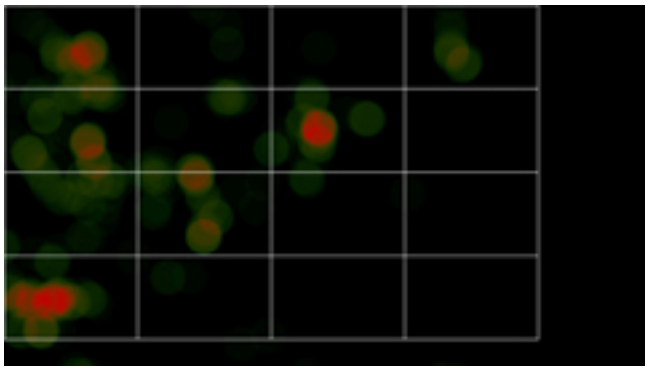
13 	A strong marker on C1 and moderate to strong indications on C3 and C8. The user looked at the previous choice (elitist parent in C1) a lot. There is no clear indication that the agents made a contribution to the decision-making process. This was not a very long run according to the distribution of the markers.
14 	The strongest marker is on C4. Moderate indications on C2, C12 and C13. The sparse distribution of measurements indicate a short run.
15 	The strongest indication is in C4. But also C8 and C12 show hotspots. There is only moderate heat in C1 and C7. This could indicate a preference of C4, but C8 and C12 are also quite strong, so that this result should be used for only a careful indication of user preference.

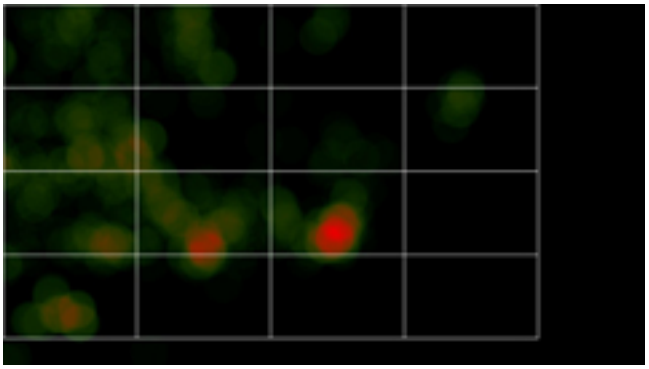
C. Eye tracking data (agents not active)

1 	A strong marker on C9 and C10, otherwise low to moderate distribution across all candidates. There is no emphasis on either parents in C1 and C2 nor C3 and C4 as before (the agents are inactive here).
2 	The distribution is fairly even across all candidates with moderate increase on C6 and C10. There is no indication of preference for C1 to C4.
3 	C11 is highlighted and C2 and C14 receive moderate attention. There is no significance on C3 or C4 nor C1.
4	C1, C2, C5 and C9 as well as C15 are emphasized. Otherwise a reasonably even distribution. While the elitist parents in C1 and C2 receive some attention, there are other candidates that are highlighted as well. This does not seem to indicate any particular preference. The intensity indicates a very long run.

	
5 	C1 is highlighted and some other candidates receive moderate attention (C2, C8, C12 and C15) in an otherwise even distribution. There is no clear evidence of any attention to C3 and C4.
6 	Overall a tendency towards the bottom of the screen with very little activity at the top and left side. The main hotspots are C11 as well as C10 and C15. There are no significant markers on C1 to C4 at all.
7 	This run shows a very even distribution across all candidates with C4 and C12 being slightly more emphasized than a number of other hotspots. This heatmap indicates a fairly long run without any particular preference. It is interesting that there is an emphasis on C4, but this could also be a random occurrence given the large number of other hotspots.

8 	There are three hotspots in this heatmap on C2, the boundary of C6 to C10 and C8. The hotspot on the boundary seems to align with a path that was very close to the bottom of the map. This is a glitch that only appeared a few times until I could track down the bug. All other significant markers seem to align with the grid, so I do not think that the calibration of the eye-tracker caused the anomaly.
9 	This run is another case where numerous hotspots are visible due to a long run. There is no emphasis on the top row though, with C5, C10 and C15 being the most significant markers. This plot also shows a marker off to the right of C4, which is owed to the position of the three dimensional display of the selected parents. This participant made some use of that feature.
10 	A shorter run with rather faint markers all over, and just one moderate hotspot on C2.
11 	This run has one clear preference for C8, which is one of the random candidates. Further, there are three moderate hotspots on C1 (elitist parent), C11 and C12. No particular emphasis on the top row.

12 	This run has a clear emphasis on C5 and moderate markers on C2 and C7. Otherwise a reasonably even distribution. The participant shows a tendency to prefer the left side of the grid.
13 	There are two main hotspots in this heatmap in C5 and C10. A moderate density on C2. This participant looked mostly to the left side of the grid.
14 	The overall density is lower than in (12) and (13), but there seems to be a similar tendency to favour the left side of the grid. It is the same participant as in (12) and (13) and there seems to be a preferred gaze direction. I am going to unpack this as part of the overall discussion of the eye tracking results.
15 	This run has a couple of moderate hotspots in C7 and C13 with an otherwise reasonably even distribution across other grid cells.

16 	This heatmap shows a hotspot in C11 and a moderate emphasis on C10. The distribution is spread across the whole grid without any particular attention to the top row C1 to C4.

D. Participant Information Sheet, Questionnaire and Interview Questions

Participant Information Sheet

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Date Information Sheet Produced:

21rd March 2017

Project Title

Computational Generators and Evaluators for Video Game Level Design using Cognitive Player Models

An Invitation

I am a PhD student at Auckland University of Technology (AUT) and I am currently undertaking a research study that involves procedural level generation (making game levels with help of software), which will be used to automate some aspects of game level design.

You are being invited to take part in this research study. Your participation in this study is completely voluntary and you can withdraw from this study at any time, without explanation. You also have the right to withdraw retrospectively any consent given, and to require that any data gathered on you be destroyed. If you are a student at AUT then a decision not to participate will not affect your grades or work performances in any way.

The results of this research study will be used for my PhD research. Before you decide it is important for you to understand why the research is being done and what it will involve. Please take as much time as you would like to read this information carefully.

What is the purpose of this research?

The aim of this research study is to investigate whether it is possible to generate playable game levels semi-automatically, and if they can be automatically evaluated based on real game play (your game play). Please remember that your responses are used to evaluate the implementation of the game logic in the game, they are not used to evaluate you as an individual.

How was I chosen for this invitation?

You were chosen, along with 9 others, because you've indicated your interest in participating in the experimentation by responding to an invitation circulated through groups with a direct interest in gaming and game development. We assume that you are frequently playing computer games based on the original invitation and that you have game level design experience and understanding at least at a fundamental level.

What will happen in this research?

You will first be asked to complete a short questionnaire that asks questions about your prior gaming and design experience. You will then be asked to create a game level using our software tool, until you consider the level being finished, or until asked to stop by the researcher. This will be for a maximum of thirty minutes. You will not be made aware of how the software logic has been implemented prior to designing the game level(s). Input data (mouse and keyboard) is collected during the level design, and frequent screen captures are taken using a screen capture logging tool. You are encouraged to talk aloud during your design to identify elements of the

experience that are either frustrating or interesting. When you make a comment indicating that the game is either frustrating or interesting, the researcher will record your comment taking notes. The researcher will also be observing the design process and may record the time at which relevant input events occur. No other data is collected during the observation of your game level design. You will have the option to review the screen capture, notes and other data, and you may immediately withdraw from the study if you prefer to do so. Upon finishing the design experiment, you will be asked a few questions by the researcher in an interview. The researcher will take notes during the interview and store these electronically. Electronic data will be backed up and stored on a secure harddrive. No hardcopies will be created, as interview notes will be typed on a laptop.

The data will be stored for a minimum of six years.

After the minimum storage time has elapsed, the data will be destroyed from the cloud storage server.

What are the discomforts and risks?

We do not expect you to experience any discomfort during the design experiment. You will be using mouse and keyboard standard to lab computers at AUT. If you experience any form of physical discomfort during the experiment, you are advised to stop the design immediately and withdraw from the study.

What are the benefits?

There are no tangible benefits to you for participating in this research study other than the opportunity to play test a new game.

How will my privacy be protected?

Any data collected in this research study will remain confidential. Any analysis of results that would be published will be anonymous.

You have the right to withdraw from the interview at any time without giving a reason and you can withdraw your name and interview data up to 2 weeks after the interview. Should you wish to withdraw, all data collected during the interview will be deleted from the harddrive.

What opportunity do I have to consider this invitation?

Since this research study involves spending up to an hour to complete the design testing and interview, the researcher understands that it requires time for you to consider whether you wish to participate. The researcher will appreciate if you could send a reply within two weeks of the receipt of this form.

How do I agree to participate in this research?

If you decide to take part in this research study, you will be asked to sign a consent form on the day of study.

Will I receive feedback on the results of this research?

If you wish to receive a summary sheet of the experimental findings or publications, the results are likely to be published in July 2018 and will be made available on Scholarly Commons, the AUT research repository. You will also be invited to attend a future New Zealand Game Developers Meetup at which the findings will be presented.

What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Project Supervisor, Dr. Andy Connor, andrew.connor@aut.ac.nz and Tel: +64 9 921 9999 extn 5211.

Concerns regarding the conduct of the research should be notified to the Executive Manager, AUTEC, Kate O'Connor, *ethics@aut.ac.nz* , 921 9999 ext 6038.

Whom do I contact for further information about this research?

Researcher Contact Details:

Jan Kruse, jkruse@aut.ac.nz

Project Supervisor Contact Details:

Dr. Andy Connor, email: andrew.connor@aut.ac.nz, tel: +64 9 921 9999 extn 5211, address: Colab (D60), Private Bag 92006, Auckland 1020, New Zealand.

Approved by the Auckland University of Technology Ethics Committee on <date>, AUTEC Reference number <number>.

This version was last edited on 11 July 2006

Participant ID:

Pre-Participation Questionnaire

Game Level Generators and Evaluators

SECTION 1: PERSONAL DETAILS			
1.1	Please indicate your age range		
	O 18-29	O 30-49	O 50+
1.2	Please indicate your gender		
	O Male	O Female	O Other

SECTION 2: DESIGN EXPERIENCE				
2.1	How many years have you been designing computer games?			
	O 1-5	O 6-10	O more than 10	
2.2	What is your level of design experience for games in general?			
	novice			*expert*
	1	2	3	4
	O	O	O	O
2.3	What is your level of design experience for first person games?			
	novice			*expert*
	1	2	3	4
	O	O	O	O
2.4	How many levels have you designed in this capacity?			
	O 1-2	O 3-10	O 10+	
2.5	Have you employed automated design software before?			
	O No	O semi-automated	O fully automated	
2.6	How did you control the input of these design tools (if any) ?			
	O numerical inputs	O graphical user interface	O script	

SECTION 3: GAMING EXPERIENCE			
3.1	How many years have you been playing video games?		
	O 1-5	O 6-10	O more than 10

Participant ID:

3.2	What platforms do you play on? Choose all that apply. O Xbox One O PS3 O PS4 O Xbox 360 O Wii O Wii U O PC O Other (Please specify) _____
3.3	How often do you normally play video games? O Daily O Weekly O Monthly O Never
3.4	How long do you normally play games in each play session? O 0-15 minutes O 15-30 minutes O 30-60 minutes O 1-2 hours O 2-3 hours O 3+ hours
3.5	To what extent do you enjoy playing video games? Not at all Very Much 1 2 3 4 O O O O
3.6	Do you currently play more than one game? O Yes O No
3.7	What game are you currently playing the most?
3.8	What genres of games are you most familiar with? Choose all that apply. O Action O Shooter O Action-Adventure O Adventure O Role Playing O Simulation O Strategy O Sports O Arcade O Mystery O Other (Please specify) _____
3.9	What is your favourite genre of game? Choose just one. O Action O Shooter O Action-Adventure O Adventure O Role Playing O Simulation O Strategy O Sports O Arcade O Mystery O Other (Please specify) _____
3.10	What is your favourite game of all time?
3.11	How much experience have you got with first person perspective games? None at all A lot 1 2 3 4 O O O O

Indicative Interview Questions – Game Designers

Game Level Generators and Evaluators

1. You have just completed a design experiment using a semi-automated level design tool.
    a. How would you describe the overall experience?
    b. How difficult was it to use the tool?
    c. Do you think that you have achieved the levels that you set out to create at the start of the experiment?
2. You have completed several levels.
    a. Did you find it easy to make very different levels?
    b. Did you notice a difference between the different design tests?
    c. Do you think that you created any playable levels, if so, which ones?
    d. Do you think that you made any un-playable levels, and if so, which ones?
    e. Regarding 4.c. and 4.d., why do you think that is the case?
3. These tools use autonomous agents that are small smart programs that attempt to understand your preferences during the design process. These agents were only active in some of the tests.
    a. Did you notice a difference?
    b. Do you think autonomous agents are a useful addition to this tool?
4. What would you change regarding the usability of the tool?
5. Did you like the fact that you had only simple choices available to you, e.g. just being able to choose two 'parent' levels to create new levels?
6. What would you change regarding the responsiveness of the tool with regards to your design goals?
7. Do you think that a tool using semi-automated design processes is useful or an obstacle to game level design?
8. How do you envision the ideal semi-automated design support tool?