

# 2D Game Design using GameMaker

---



Presented by:  
Liam Breytenbach

Prepared for:  
Dr Francois Schonken Dept. of Electrical and Electronics Engineering  
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town  
in partial fulfilment of the academic requirements for a Bachelor of Science degree in  
Mechatronics.

**October 24, 2024**

## Declaration

---

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced. Any section taken from an internet source has been referenced to that source.
3. This report is my own work and is in my own words (except where I have attributed it to others).
4. I have not paid a third party to complete my work on my behalf. My use of artificial intelligence software has been limited to an assistive tool and has not completed the work for me. How I have used AI can be seen in appendix B.1.
5. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
6. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Signature:



**Liam Breytenbach**

Date: **October 24, 2024**.

Word Count: **13 848**.

## Terms of Reference

---

This project is aimed at the creation of a fully functional 2D game, as well as a detailed report, within a 12 week period. The report must demonstrate a comprehensive understanding, breakdown, design, implementation, testing and analysis of the game. A Literature review should discuss established theories and practices used in today's game industry, and ethical concerns must be addressed whilst external participants are evaluated.

A game is typically designed through a **team of developers** over an extended period of time. Consequently, the game's complexity, graphics and overall length must be considered prior to commencement.

## Acknowledgments

---

I would like to thank my supervisor Dr Francois Schonken for guiding me through my project. I would also like to thank Stacey Shield, and Sampath Jayalath for giving me assistance with my report layout and content.

Thank you to all participants that helped me conduct my play testing procedure, your feedback is critical to my report and future work around the topic. I would also like to thank Sizwe Sigasa for helping me create the music and sound effects in the game.

Lastly, I would like to thank my friends and family who have been incredibly supportive throughout my engineering degree, being by my side through my best and toughest moments.

## Abstract

---

Video games have become extremely popular since the late 1970s and 1980s. In recent years, the opportunities for game development have grown due to the advancement of powerful development tools.

This project uses the game development software GameMaker to develop a functional 2D game. By reviewing literature necessary to develop an enjoyable 2D game, the design of the project is rooted existing theories and practices. To ensure that the deliverables of the project are met, each submodule of the design is targeted at a specific requirement to ensure the project's success. Through simulation in the game development software, 3D models and prototyping, the designs of the project can be finalised.

In a experimental testing procedure, every aspect of the designs were evaluated. Following a participant based play test, the acceptance test procedures could be verified, using the average score obtained from each question as a quantifiable metric. The results acquired from this experimental testing procedure were discussed, showcasing the projects performance in meeting the expected deliverables.

From the play testing questionnaire, real-world feedback was attained. By using these recommendations, improvements can be made, potentially taking the game to an industry competing level.

*“If life is a video game, the graphics are great, but the plot is confusing & the tutorial is way too long.”*

– Elon Musk

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Project Background . . . . .	1
1.2	Objectives . . . . .	1
1.3	Scope and Limitations . . . . .	2
1.3.1	Scope . . . . .	2
1.3.2	Limitations . . . . .	2
1.4	Plan of development . . . . .	2
<b>2</b>	<b>Literature Review</b>	<b>4</b>
2.1	Game Mechanics, Dynamics and Aesthetics . . . . .	5
2.2	Procedural Generation . . . . .	7
2.3	Prototyping . . . . .	9
<b>3</b>	<b>Technical Review</b>	<b>13</b>
3.1	Unity vs GameMaker . . . . .	13
3.1.1	Unity . . . . .	13
3.1.2	GameMaker . . . . .	14

<b>4 Design Objectives and Methodology</b>	<b>16</b>
4.1 Design Objectives . . . . .	16
4.2 Methodology . . . . .	16
4.2.1 Requirements . . . . .	17
4.2.2 Literature Analysis . . . . .	18
4.2.3 Module Design and Simulation . . . . .	19
4.2.4 Validation . . . . .	19
<b>5 Design and Simulation</b>	<b>21</b>
5.1 Game Design . . . . .	21
5.1.1 Paper Prototyping . . . . .	22
5.1.2 Game Mechanics . . . . .	23
5.1.3 Game Dynamics . . . . .	33
5.1.4 Game Aesthetics . . . . .	37
5.2 Controller Design . . . . .	40
5.2.1 Paper Prototype . . . . .	40
5.2.2 Component Selection . . . . .	42
5.2.3 Mechanical Design and Modelling . . . . .	44
5.2.4 Electrical Design . . . . .	44
5.2.5 Physical Prototyping . . . . .	45
5.2.6 Prototype 1 . . . . .	46

5.2.7	Prototype 2 . . . . .	46
5.2.8	Prototype 3 . . . . .	47
<b>6</b>	<b>Experimental Testing Procedure</b>	<b>49</b>
6.1	Play Testing Setup . . . . .	50
6.2	Play Testing Results . . . . .	50
6.3	Pre-Play Test . . . . .	51
6.4	Post-Play Test . . . . .	55
<b>7</b>	<b>Discussion</b>	<b>58</b>
7.1	Data Analysis and Verification . . . . .	58
7.2	SWOT Analysis . . . . .	60
7.2.1	Strengths . . . . .	60
7.2.2	Weaknesses . . . . .	61
7.2.3	Opportunities . . . . .	61
7.2.4	Threats . . . . .	61
<b>8</b>	<b>Conclusion and Recommendations</b>	<b>62</b>
<b>A</b>	<b>Additional Files and Schematics</b>	<b>68</b>
A.1	Graduate Attribute Requirements . . . . .	68
A.1.1	GA 1: Problem Solving . . . . .	68
A.1.2	GA 4: Investigations, Experiments and Data Analysis . . . . .	69

A.1.3	GA 5: Use of Engineering Tools . . . . .	69
A.1.4	GA 6: Professional and Technical Communication . . . . .	70
A.1.5	GA 8: Individual Work . . . . .	70
A.1.6	GA 9: Independent Learning Ability . . . . .	70
A.2	Project Description . . . . .	72
A.3	Ethics Approval . . . . .	75
A.4	Play Testing Questionnaire . . . . .	76
A.5	Informed Consent . . . . .	82
<b>B</b>	<b>Project Folder and AI Use</b>	<b>83</b>
B.1	Project Folder . . . . .	83
B.2	AI Use . . . . .	83

# List of Figures

2.1	Activity Theory hierarchical structure [1] . . . . .	4
2.2	Player centered game mechanics [2] . . . . .	6
2.3	Tiling using pseudo randomness [3] . . . . .	9
2.4	Frequently cited prototyping objectives [4] . . . . .	10
2.5	Iteration and the iterative process [5] . . . . .	11
4.1	Game Design and Controller Design Process . . . . .	17
4.2	Gantt Chart - 2D Game Design using GameMaker . . . . .	18
5.1	Final Paper Prototype . . . . .	22
5.2	GameMaker Co-ordinate System . . . . .	23
5.3	GameMaker Co-ordinate System at a position . . . . .	23
5.4	Procedural Generation Flowchart Example . . . . .	28
5.5	Procedural Generation Output . . . . .	29
5.6	Issue presented with removing blocks that have not been populated . . .	30
5.7	Procedural generation flowchart for creating a cave . . . . .	31
5.8	Procedural generation cave output . . . . .	32
5.9	Procedural generation cave outline output . . . . .	32

5.10 Tutorial Controls . . . . .	35
5.11 Tutorial Instructions . . . . .	35
5.12 Enemy Life-cycle . . . . .	36
5.13 Image Size fps comparison . . . . .	38
A    Blocks . . . . .	39
B    Virtual Currency . . . . .	39
C    Fluids . . . . .	39
D    Icons . . . . .	39
E    Level Icons . . . . .	39
F    Traps . . . . .	39
G    Weapons . . . . .	39
H    Player Skins . . . . .	39
5.14 Sprite Examples . . . . .	39
5.15 Sprite Animation Example . . . . .	40
5.16 Controller Paper Prototypes . . . . .	41
5.17 Micro controller dimensions . . . . .	42
5.18 Usable Space in a standard PlayStation 4 Controller . . . . .	43
5.19 Mini SMD Momentary Switch . . . . .	43
5.20 Switch with LED Circle pattern . . . . .	43
5.21 Momentary Push Button Switch Horizontal . . . . .	43

5.22 Analog Joystick Module with Switch . . . . .	43
5.23 Full Model Iteration 1 . . . . .	44
5.24 Controller Electrical Schematic . . . . .	45
5.25 Controller Prototype 1 . . . . .	46
5.26 Controller Design Changes . . . . .	46
5.27 Controller Prototype 2 . . . . .	47
5.28 Controller Design Final Changes . . . . .	47
5.29 Controller Prototype 3 . . . . .	48
6.1 Age Distribution . . . . .	52
6.2 Gender Distribution . . . . .	52
6.3 Gaming Frequency . . . . .	53
6.4 Genre of Games . . . . .	53
6.5 Familiarity with 2D Platformers . . . . .	54
6.6 What Participants Enjoy about 2D Games . . . . .	54
6.7 Example of Question Layout presented in the Play Testing Questionnaire . . . . .	55
6.8 Game Evaluation . . . . .	56
6.9 Participants that encountered bugs during gameplay . . . . .	57
A.1 Project Description Page 1 . . . . .	72
A.2 Project Description Page 2 . . . . .	73
A.3 Project Description Page 3 . . . . .	74

A.4 Ethics Approval . . . . .	75
A.5 Play-Testing Questionnaire P1 . . . . .	76
A.6 Play-Testing Questionnaire P2 . . . . .	77
A.7 Play-Testing Questionnaire P3 . . . . .	78
A.8 Play-Testing Questionnaire P4 . . . . .	79
A.9 Play-Testing Questionnaire P5 . . . . .	80
A.10 Play-Testing Questionnaire P6 . . . . .	81
A.11 Informed Consent . . . . .	82
B.1 Asking AI to reword paragraphs . . . . .	84
B.2 Asking AI to reword paragraphs . . . . .	84
B.3 Asking AI to make tables . . . . .	85
B.4 Asking AI to make text bold . . . . .	85
B.5 Asking AI for assistance with Figure layout . . . . .	86
B.6 Asking AI for assistance with GameMaker errors . . . . .	86
B.7 Asking AI for assistance with GameMaker errors . . . . .	87

# List of Tables

4.1	User Requirements . . . . .	17
4.2	Technical Requirements . . . . .	18
4.3	Acceptance Test Procedures . . . . .	20
5.1	Description of Parameters . . . . .	27
5.2	Parameter Values . . . . .	28
5.3	Game Levels and Difficulty Ratings . . . . .	34
6.1	Breakdown of Time . . . . .	50
6.2	Age Distribution . . . . .	51
6.3	Gender Distribution . . . . .	51
6.4	Gaming Frequency . . . . .	51
6.5	Genre of Games . . . . .	51
6.6	Familiarity with 2D Platformers . . . . .	51
6.7	What Participants Enjoy about 2D Games . . . . .	51
6.8	Game Evaluation . . . . .	56
6.9	Summary of Encountered Bugs . . . . .	57
7.1	Acceptance Test Procedures . . . . .	58

7.2 SWOT Analysis . . . . .	60
-----------------------------	----

# Chapter 1

## Introduction

### 1.1 Project Background

Recent years have seen corporate greed actively destroy the AAA game industry (dominating companies that produce the highest development budget games), which has been a significant motivator for developers to move to the **Indie** (Independent Game Development) space. With the availability of advanced game development tools, 2024 has been an exceptional year for Indie games. This project will take advantage of one such tool, creating an enjoyable and fully functional 2D game using **GameMaker**.

### 1.2 Objectives

This project aims to develop a 2D game with procedurally generated terrain and engaging gameplay mechanics. Using the development software GameMaker, a fully functional game will be put forward, as well as a functional gaming controller that meets the player's needs and adheres to technical constraints. In addition, the game will be made with sufficient complexity by incorporating advanced game mechanics, dynamics and an aesthetically impressive display. Furthermore, ample opportunity will be provided for testing and evaluation of the game, ensuring that the expected deliverables can be presented. The deliverables of the project include a fully functional 2D game and detailed report that demonstrates a comprehensive understanding, breakdown, design, implementation, testing and analysis of the game.

## 1.3 Scope and Limitations

### 1.3.1 Scope

The scope of this project involves creating a fully functional 2D game within a 12 week period. The design of a gaming controller and cover is outside the scope of this project, but serves to enhance the 2D game by combining the 'old-school' 2D platformer experience with a modern and familiar style of console experience. Furthermore, a play testing analysis is conducted to evaluate the game's functionality, integrity and the enjoyment player's receive from the experience. The metric used in the play testing analysis is the number of participants who respond to a specific question, providing valuable data for evaluating the game's performance.

### 1.3.2 Limitations

The project was limited to using the game engine software GameMaker for development. Given the 12 week time constraint and the fact that a single developer is designing the game, the complexity of level design, graphics and game size has been restricted to a manageable workload. In the experimental testing procedure, the sample size of participants has been limited to 20, due to the period of time required to locate potential candidates, interview each participant, and process the data obtained from each questionnaire.

## 1.4 Plan of development

Following on from the introduction, chapter 2 reviews the relevant literature required to develop a functional 2D game, focusing on meaningful play, game mechanics, dynamics and aesthetics. It also discusses importance of prototyping, emphasizing the advantages different forms of prototyping can offer to the project.

In chapter 3, two game engine's are reviewed, exploring their respective strengths and weakness in game development.

Through chapter 4, the project's objectives and methodology are addressed, breaking

#### 1.4. PLAN OF DEVELOPMENT

down the user requirements, creating acceptance test procedures and walking through the methodology used in this project.

The design and simulation of the game is then detailed in chapter 5, outlining how each user requirement is achieved through the project. The game design section discusses the process involved in the creation of the game, focusing on the mechanics including movement, procedural generation and enemy logic. The game dynamics and aesthetics are then discussed, detailing how meaningful play is brought about in the game, as well the critical role aesthetics play in system performance, while still maintaining player experience. The controller is also addressed in chapter 5, exploring component selection, the choice of microcontroller, modelling, prototyping, as well as the final design used for the project.

In chapter 6 the acceptance test procedures are verified through a participant based play test and questionnaire, using the average score obtained from each question as a quantifiable metric.

The results acquired from this experimental testing procedure are then discussed in chapter 7, showcasing the projects performance as a whole.

Finally, the conclusions of the project are revealed in chapter 8, along with future recommendations for similar work in the **Game Design** field.

# Chapter 2

## Literature Review

The exact definition of a game is heavily debated within the game design community. Adams and Dormans [6] describe a game as a type of play activity where there is at least one arbitrary, non-trivial goal that is accomplished while abiding by certain rules. The meaning of a game is further spoken about by Salen and Zimmerman [7] as a process where players participate in artificial conflict, bound by rules to obtain a quantifiable outcome. In a more general sense of the meaning of a game, Carvalho et al. [1] argue that a game is a form of activity, outlined by Russian psychologists, notably Leont'ev and Vygotsky during their research in Activity Theory in the 1920s and 1930s [8]. Carvalho et al. [1] go on to speak about an activity as a meaningful interaction between subject and object, that facilitates mutual transformations. Each activity has a motive, every action is directed at a goal, and every operation only takes place under certain conditions [1]. This can be seen below.

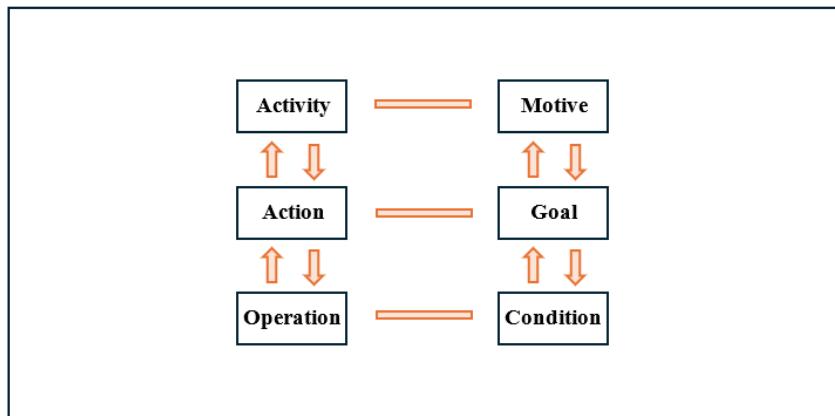


Figure 2.1: Activity Theory hierarchical structure [1]

The word 'meaningful' raises further questions in game design. What is meaningful play? Salen and Zimmerman [7] characterise the depth and quality of meaningful play as the interaction between a player's choice and a system's response to these choices. Salen and Zimmerman further mention that meaningful play does not intrinsically emerge from a game itself, but rather from how the player interacts with the game in order to play

## 2.1. GAME MECHANICS, DYNAMICS AND AESTHETICS

it. Designing a game with meaningful play, requires the concepts of a game to be broken down. Armory [9] speaks about a game being broken down into five distinct spaces: Game Space, Visualisation Space, Elements Space, Actor Space and Problem Space. While this way of thinking about a game is well-founded, Armory is relating this way of thinking to the elements of a Serious Game (a game rooted in educational benefit). While elements used entertainment games can be translated to Serious Games, not all of them apply. This concept is discussed in a paper by Arnab et al. [10], which discussed the meaning of the mechanics within serious games in the hope of providing a more systematic understanding of the relationship between pedagogical principles (concepts that inform teaching practices and learning processes) and game elements. Arnab et al. confirm that certain genres found in entertainment games including action, adventure, role-playing, and simulation genres, exhibit similar gameplay dynamics and interactions, which can be translated to games used in the educational field. In an entertainment game, Kim [11] and Hunicke et al. [12] speak about the basic core elements that make up a game as the game Mechanics, Dynamics and Aesthetics. These core elements result in the player's ability to process the game's rules, system and ultimately, the enjoyment that comes from playing.

### 2.1 Game Mechanics, Dynamics and Aesthetics

There are many definitions regarding what game mechanics entail. The mechanics of a game are defined by Adams and Dormans [6] as “the rules, processes and data at the heart of a game”. They define how a player progresses through a game and what the conditions are that result in victory or defeat. Adams and Dormans also speak about game mechanics as the rules of a game that are hidden from a player. This definition differs slightly from the meaning presented by Hunicke et al. [12] stating that a game's mechanics are the specific elements of a game at the fundamental level of data representation and algorithms. To avoid confusion, Carvalho et al. [1] define these game elements based on three layers of the gaming activity: the actions, tools and goals. Carvalho et al. define game mechanics as a specific set of rules that govern how interactions within a system lead to outcomes, involving an input, a process, and an output. In contrast, game dynamics refer to the player's reactions or responses to combinations of these mechanics within the gameplay experience. In Marczewski's book Gamification [13], game mechanics are described as the distinct set of rules that dictate outcomes of interactions within a system, based on a specific input. This is well illustrated by Fabricatore [2], where player centered game mechanics are influenced by an input, resulting in outputs that lead to new game mechanics.

## 2.1. GAME MECHANICS, DYNAMICS AND AESTHETICS

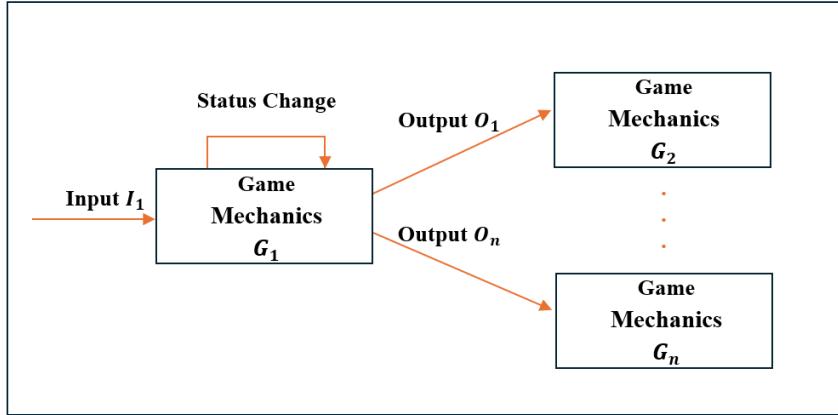


Figure 2.2: Player centered game mechanics [2]

While the exact definition of game mechanics differ, all of them have a similar idea about what a game mechanic entails. They are the fundamental elements that make up the rules governing players' interactions with other entities and objects within a game. Frasca [14] has mentioned two different categories of game rules: those that allow the player to interact with game elements and those that establish the game's objectives. These two types of rules correlate to the micro and macro choices the player is encountered with in a game. Salen and Zimmerman [7] refer to the micro-level choices as the small moment to moment choices a player is exposed to while playing the game. At a macro level, these individual choices combine to form a trajectory of experience. This trajectory of experience can be seen as the dynamics of a game, described by Kim [11] and Hunicke et al. [12] as the users' responses to a collection of game mechanics.

Papadimitriou and Piliouras [15] describe game dynamics as processes by which players move in a space of mixed strategy environments, according to specific rules, in response to what other entities are doing. This may seem very similar to game mechanics however, it differs in the fact that multiple game mechanics are required to form a game dynamic. The dynamics of a game are the result of a collection of smaller, more fundamental components. Game dynamics in an evolutionary sense, can be separated into deterministic and stochastic dynamics according to Sandholm [16]. Deterministic dynamics follow strict rules and do not involve randomness. Given a current state, one can predict exactly how the population will evolve. Stochastic dynamics however, incorporate randomness, meaning that a populations future is influenced by fluctuations and random events. By adding randomness to a game, a more complex game play strategy can be achieved. Adams and Dormans [6] state that a game should be unpredictable to a certain extent, and can be achieved by including an element of chance along with the player's choices. There are a vast array of game dynamics that can be incorporated into game design. As mentioned by Hunicke et al. [12], the dynamics of a game work to create the aesthetic

experiences that a user can visually appreciate.

According to Niedenthal [17], the aesthetics within a game refer to the sensory phenomena experienced by a player within a game, aspects of a game that are shared with other art-forms, and the expression of the game experienced as emotion, pleasure and sociability. The sensory phenomena spoken about refer to the visual, aural and haptic stimulation that a game could possess, surrounding the user with aesthetic experiences. Somerseth [18] has a similar understanding of game aesthetics, describing it as the relationship between immersion and interactivity. Somerseth goes onto say that the understanding of aesthetics must begin with the understanding of its basis in embodied sensations. An interesting concept with reference to game aesthetics is the graphical user interface (GUI). A GUI is an operating display shown on a screen, that presents options and information to a user. Jansen [19] makes an analogy of a GUI and a car's steering wheel stating, "The wheel directly binds the driver to the functionality and operation of the vehicle". Barahrom [20] points out that visual aesthetics should be focused on the art styles of GUI and game characters that surround the core of the game.

As seen, the GUI is a critical part of the game's design. Ronchi [21] points out that a well developed interaction design, links the instructions from man to computer, enhancing the interaction between them. The design of the graphical user interface must be thought about thoroughly in the design process. Dutra [22] explains that only what a user needs to know and what options they have available, should be presented on the screen. This affects how the design is developed as spacial awareness, and relevance of information must be taken into account. The player should also be familiar with the information that is displayed. Hu and Ji [23] state that the center design surrounding a familiar GUI should take the user's personal knowledge, local knowledge and general knowledge into consideration. By optimizing the space, icons, symbols, simplicity and direct navigation, a strong graphical user interface can be developed [22].

## 2.2 Procedural Generation

Creating levels using the principles of game mechanics, dynamics and aesthetics is a challenging and time-consuming process. Ashlock, Lee, and McGuinness [24] state that in the domain of level design, procedural generation can play a pivotal role in the creation of the level, challenges and rewards.

Van der Linden, Lopes, and Bidarra [25] describe Procedural Content Generation (PCG)

## 2.2. PROCEDURAL GENERATION

as “the algorithmic creation of content”. It allows content to be created automatically, significantly reducing the workload imposed on artists and developers [25]. Togelius et al. [26] have a similar view, describing PCG as “the creation of game content automatically using algorithms”. While PCG saves a considerable amount of time for developers, Van der Linden et al. [25] explain that there are other benefits that come about from using algorithms to generate levels. These include a diversity in content creation, possibly increasing a game’s replayability as well as providing a basis for game’s to dynamically adjust to suit the player [25]. A crucial element of and PCG method is the developers ability to control the output and properties. This process of control is the ability to ‘steer’ the level generation process toward a desired outcome and must consider how much effort the ‘steering’ requires [25]. Kelly and McCabe[3] make clear that the instructions for PCG should be parameterized to enable the generation of instances to have varying characteristics. This also helps the developer guide the generation of the level, as parameters can be easily adjusted and manipulated to obtain the desired outcome. One of the most basic techniques for PCG is the use of random parameters or use of random generators within the Procedural Content Generation [3].

Random number generators used in most algorithms are not truly random. Vadhan et al. [27] describe this method used in computing as pseudo randomness. Pseudo random numbers are generated through deterministic calculations to appear random, despite being constructed using little to no randomness. Vadhan et al. [27] discuss how pseudo randomness generated from deterministic methods can often replace random methods in computation. Nisan [28] describes that it is possible to obtain truly random generation using polynomial time computable pseudo-random generators, which uses time in its calculations to output a random number. While it is possible to obtain a random number this way, this method is not typically used due to computational efficiency [27].

By using a pseudo random function, Kelly and McCabe [3] state that thousands of possible permutations can be obtained through a single probability map. Tiling is a good example of how a simple procedural generation function can be applied to more complex graphical systems in game design [3]. An example of how tiling can be used is illustrated below.

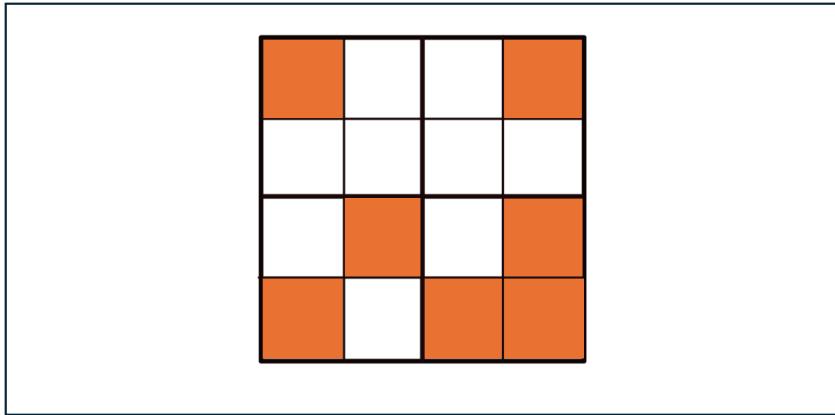


Figure 2.3: Tiling using pseudo randomness [3]

Figure 2.3 serves to represent how a two by two block can be tiled with white or orange in a pseudo random manner, to obtain different appearances. The effectiveness of PCG lies in the developer's ability to guide the generation process [3]. As demonstrated in techniques such as tiling, pseudo random functions play an important role in generating complex and visually appealing game environments, while still maintaining computational efficiency.

## 2.3 Prototyping

A critical part of the development process is prototyping. A prototype is a physical version of an idea that makes abstract concepts easier to understand. according to Beaudouin-Lafon and Mackay [29]. Jang and Schunn [30] found that teams that frequently designed and used prototypes and physical models, outperformed their peers with respect to technical quality. Elsen et al. [31] also found that graduate and professional engineers that produced prototypes early in the design phase, produced a significantly higher technical quality in their designs. Similarly, Neeley et al. [32] found that when prototypes were used in the design phase, the end functionality of the designs were greatly improved. Neeley et al. [32] imposed that the test group produced four more prototypes than the control group, displaying the significant impact prototyping can achieve. Schrage [33] states that the benefits of prototyping are achieved through engineers and designers being able to experiment and test new concepts in a way that promotes the progression of the design. There are many different potential objectives that design prototyping can achieve. This is spoken about by Camburn et al. [4] in their research reviewing full sets of articles discussing objectives of prototyping as a process. These objectives were grouped into similar categories and displayed relative to prevalence (by area).

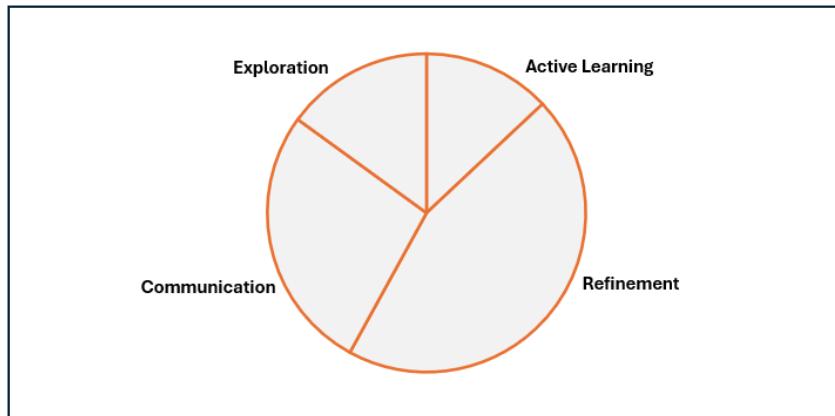


Figure 2.4: Frequently cited prototyping objectives [4]

Camburn et al. [4] speak about these frequently cited objectives that come as a result of prototyping. Active learning is the process of actively engaging with and gaining knowledge about a subject [4]. Menold, Jablokow, and Simpson [34] speak about prototypes as a catalyst for learning during the development of a product. They facilitate communication by enabling the exchange of information, through collaboration and testing [4]. Prototypes also support exploration, allowing designers to discover new ideas and concepts [4]. Additionally, they aid in refinement, helping to gradually improve designs through various iterations [4].

While Camburn et al. [4] discuss many different methods of prototyping, two main approaches stand out as relevant to the project: parallel and iterative prototyping. In parallel prototyping, multiple design concepts are measured against each other simultaneously [4]. This allows a vast array of concepts to be tested, and the positives and negatives of each are compared. While this approach is effective in the early stages of development, it can become a time costly method later on in the project. Iterative prototyping is the testing and refinement of a design through sequential repetition [4]. An illustration of the iterative design process can be seen below, based on Hansen and Deininger's [5] book titled, 'Student learning in multiple prototype cycles'.

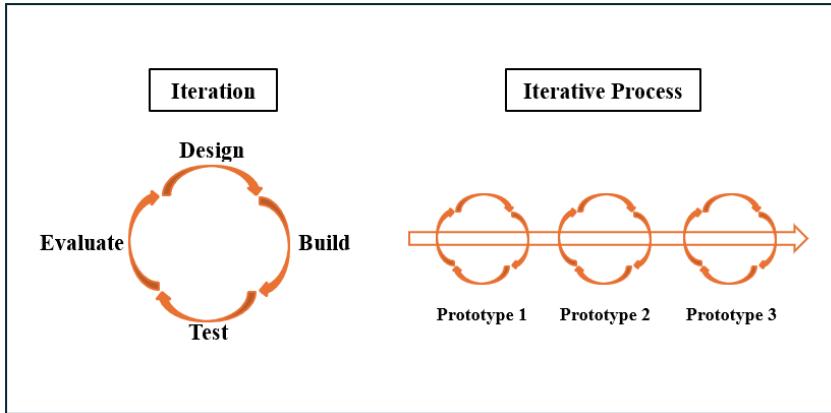


Figure 2.5: Iteration and the iterative process [5]

Zemke [35] emphasises that iteration is crucial to gathering insights into difficult problems, simplification of the design, and design flaws. In game design, the prototype that is made is often implemented as software. Software prototypes are a valuable design technique and can help break down a large problem into smaller more fundamental components. Camburn et al. [4] states that in isolated subsystem testing, a single subsystem can be explored through segmenting a prototype or concept. This technique offers programmers the ability to compartmentalize code segments, enabling them to test pieces of code, find errors and aid in the design process. While prototyping can be a lengthy procedure, there are techniques that can be used to speed up this process. Adams and Dormans [6] discuss paper prototyping as a time effective alternative to software prototypes in the early stages of development. A paper prototype is a non-digital, tabletop version of software used to represent ideas, solutions and concepts [6]. Due to a paper prototype being inherently customisable, concepts can be quickly altered and adapted to find the best possible solution to potential problems [6]. A drawback of paper prototyping is the difficulty of involving test players, as not all game mechanics can be easily translated to board games [6]. With game design especially, programming a game from scratch can be monumental task. Adams and Dormans [6] state, “To speed up the prototyping process, it is sometimes a good idea to use open-source game-engines or game developing environments such as GameMaker or Unity, even if your target platform will be something completely different”. Open-source game-engines provide a foundation on-which to start developing a game. These engines provide libraries of functions, interfaces and tools that can be used to create some incredible products.

By walking through the details that make up a game and the definitions that form the basis of meaningful play, the core elements that make up a game come to light. The game mechanics, dynamics and aesthetics are the fundamental aspects needed to create an appealing game. Going through level design, procedural generation not only

### 2.3. PROTOTYPING

to reduces the amount of time required to create a level, but also aids developers in creative approaches, and can serve as a foundation on which to build more complex and appealing levels. Prototyping and it's importance is discussed, highlighting the iterative process required for game development as well as the effectiveness of compartmentalising sections of code. Using these techniques in the engineering process, in combination with a game engine such as Unity or GameMaker, can allow developers at any level to produce remarkable designs.

# Chapter 3

## Technical Review

### 3.1 Unity vs GameMaker

There are many open source game-engines available for game development, each with their own strengths and weaknesses. While there are many options for 2D game development such as Unreal, Unity, GameMaker, CryEngine and RPG Maker, Vohera et al. [36] argue that the two game-engines that stand out specifically for 2D game design are Unity and GameMaker.

#### 3.1.1 Unity

Unity is one of the only game engines that has a specific 2D mode for development according to Vohera et al. [36]. Foxman [37] indicates that the engine has the building blocks to support both 2D and 3D games. Hussain [38] also commented that Unity has built in functionalities that support the specific needs of 2D and 3D development, as opposed to other game engines. One of the biggest advantages of using Unity over other game engines is its large and active online developer community that can offer assistance to new users according to Craighead, Burke, and Murphy [39]. This point is supported by Andrade [40], who explains that Unity is an excellent starting point for game development due to its vast community and availability of a marketplace through which, game assets and components may be traded. Foxman [37] states that the market place, "permits amateurs and professionals alike to upload homemade scenes, code, add-ons, and avatars for other users to download (for free or a fee) and populate virtual spaces". This online market place, also known as the 'Asset Store', enhances Unity's appeal for game developers as any member of the community can upload, purchase or exchange assets within the marketplace. Craighead et al. [39] also state that Unity is willing to

add features to the game engine at user request, a distinct feature that sets it apart and something that most other large engines are reluctant to offer. Another feature of Unity includes multi-platform distribution capabilities [39]. For a complete game engine, Unity engine also has a relatively low cost. Andrade [40] explains that Unity's price point is offered for free for independent game developers and to companies with an annual turnover of less than \$100 000 'US' dollars. With it's comprehensive feature set, online community and marketplace, and accessible pricing model, Unity presents itself as a strong option for game developers at all skill levels.

### 3.1.2 GameMaker

According to Andrade [40], GameMaker's advantage is it's less challenging learning entry point when compared to Unity. This is made evident by Andrade [40], stating "GameMaker is built with a non-programmer user in mind, providing a drag-and-drop editor that allows the creation of games by abstracting the underlying logic". GameMaker's core strength lies in the creation of 2D Games according to Vohera et al. [36]. While GameMaker is well equipped for 2D game development, Pönni [41] outlines that GameMaker's critical weakness is it's limited 3D support, when compared to 3D based engines such as Unity or Unreal. Andrade [40] also illustrates GameMaker's limitations, explaining that while GameMaker can support 3D functionality, the game will have to be custom scripted. GameMaker's most commonly known 2D competitor is RPG Maker. While RPG Maker is slightly more user friendly, Pönni [41] states that RPG Maker can only be used to make role-playing games. This is due to only a single camera option available, limiting the engine to a specific style of game. GameMaker however, can be used to make varying styles and genres of games, giving the engine an advantage to it's most commonly known competitor [41]. Similar to Unity, Andrade [40] and Aaltonen [42] state that GameMaker also supports a variety of target platforms for developers to export their projects seamlessly. Aaltonen [42] goes on to say that GameMaker also has an online market place where registered users may publish and sell the code, assets and sprites, much like Unity. This is particularly useful for beginners, allowing them to focus on aspects of game design they prefer, while also enabling publishers to monetise their crafted game assets [42]. GameMaker developers and the community have also uploaded many tutorials and explanations, that range from beginner to advanced. Unfortunately, GameMaker does not provide a free version option for commercial deployment according to Aaltonen [42]. With this aspect of the engine being the only major drawback, GameMaker remains a powerful and versatile tool for entry level developers to create engaging and diverse 2D games.

### 3.1. UNITY VS GAMEMAKER

A technical review of the two game development platforms **Unity** and **GameMaker**, reveals their respective strengths and weaknesses. While Unity could be considered a slightly stronger choice, due to it's expansive community, the project description clearly specifies that GameMaker must be used as the development software in this project. Since GameMaker is an excellent development platform, the subsequent chapter outlines the design objectives and methodology necessary to complete this project using **GameMaker**.

# Chapter 4

## Design Objectives and Methodology

### 4.1 Design Objectives

Using the project description and expected deliverables listed in Appendix A.1, this project is aimed at creating a fun and functional 2D game, using the development software GameMaker. Furthermore, the game is made with sufficient complexity and allows for ample opportunity for testing and evaluation.

To achieve these design objectives, the project was expanded outside the scope of expected work. To meet the deliverable of a functional 2D game, a complete game was developed along with a functional controller and cover. The description of a fun 2D game posed some challenges as the experience of 'fun' is very subjective. To achieve a quantifiable outcome of the game's enjoyment, a play-testing questionnaire was drawn up, specifically designed to analyse this aspect of the project. Creating a game with sufficient complexity required tackling the challenge of procedural generation. This design problem is still considered in today's game industry as a tremendous feat, requiring detailed planning, logic and control to produce a procedurally generated terrain. By breaking up the project into two submodules and further subcategorizing into set deadlines and objectives, the design of the system could be completed.

### 4.2 Methodology

The design of the game and controller involved analysing the relevant literature, reviewing the requirements and developing appropriate designs for simulation and testing. It outlined the iterative process required to develop effective prototypes and detailed the experimental procedure required to test the final designs of the project. The flow charts

provided in fig 4.1 give an overview of these stages, offering a clear representation of the game and controller design workflow.

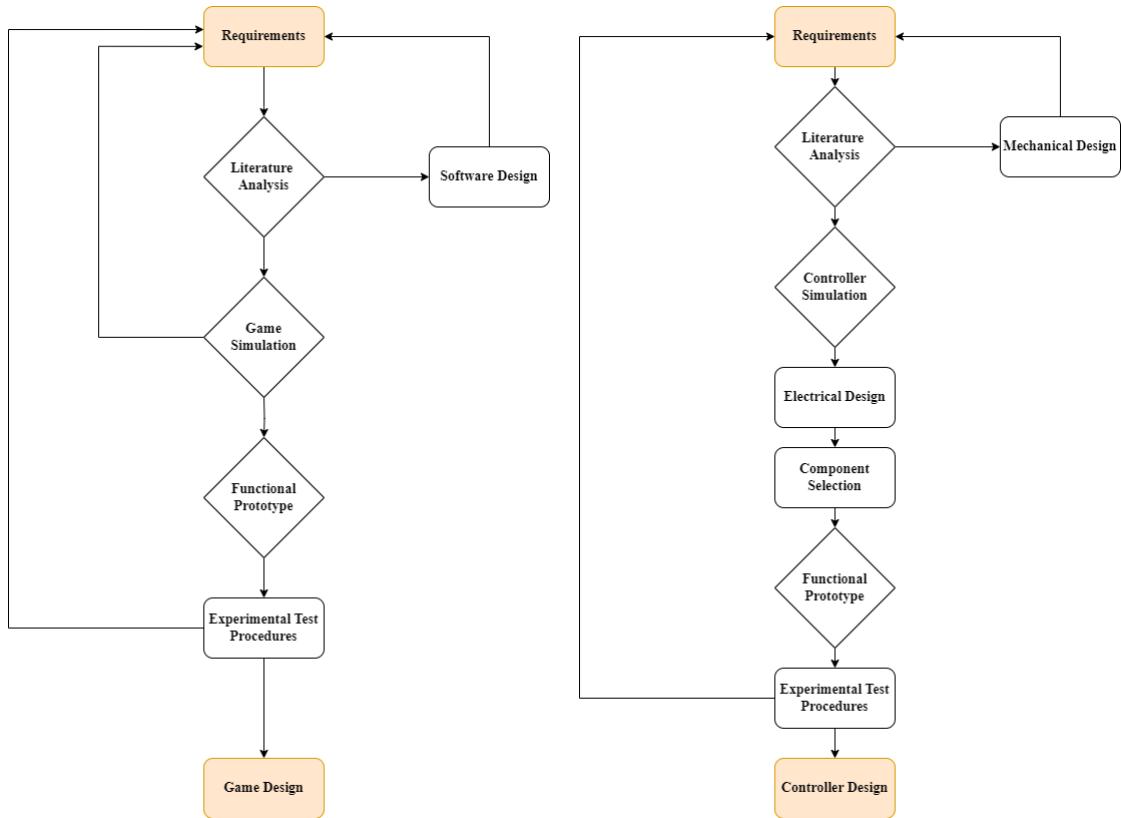


Figure 4.1: Game Design and Controller Design Process

#### 4.2.1 Requirements

The requirements were outlined prior to the project. These requirements guided the literature review, and allowed for specific focal points to be researched. They also served as a verification metric used to measure the projects success. The user requirements can be seen in table 4.1.

Table 4.1: User Requirements

User Requirement	Explanation
SR1	Must be made with GameMaker
SR2	Create a fun 2D game
SR3	Must be made with sufficient game complexity that there is enough problem solving required.
SR4	The 2D game must be functional
SR5	Must be tested and evaluated

The user requirements were utilised to determine the technical requirements of the project.

## 4.2. METHODOLOGY

These technical requirements were used in the experimental test procedures to determine if the user requirements were met. The technical requirements are shown in table 4.2.

Table 4.2: Technical Requirements

Technical Requirement	Explanation
TR1	The game must be developed with GameMaker Studio 2.
TR2	Includes complex game mechanics
TR3	A complex degree of game dynamics.
TR4	The game must have aesthetic appeal
TR5	Code should be optimized.
TR6	The game should incorporate multiple input systems and be playable on most computers.
TR7	A play-testing analysis must be conducted.

Each of the technical requirements were analysed extensively and further sub categorised throughout the project. Each section in the report details how these technical requirements were met and process involved in verifying their success. A Gantt chart presents the projects timeline, drawing attention to specific tasks, their respective duration, as well as the milestones needed to be achieved throughout the project's life-cycle. The Gantt chart used for the project can be seen in fig 4.2.

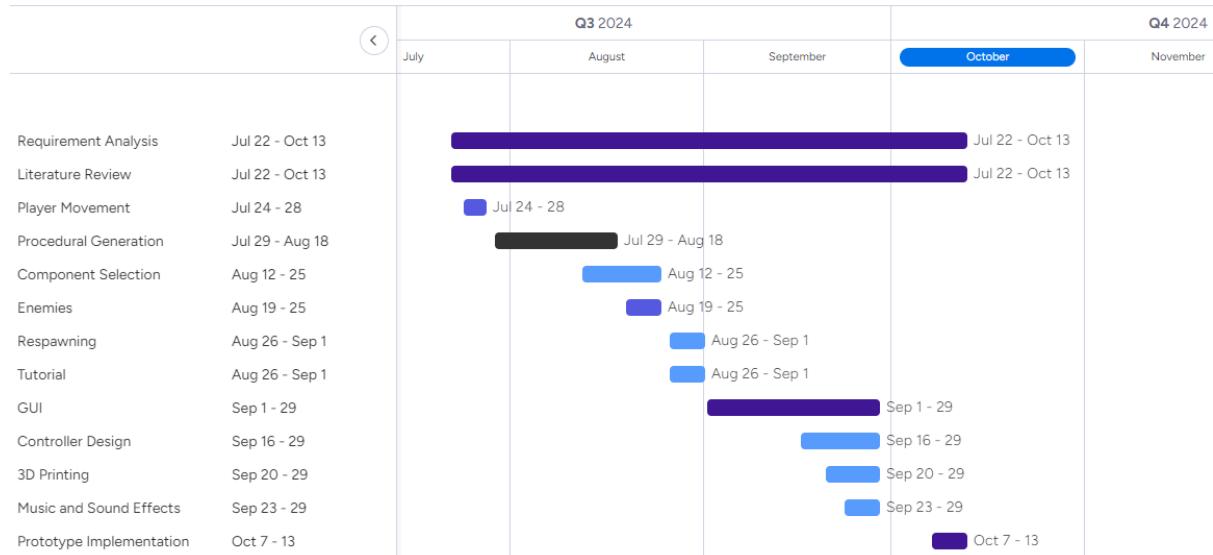


Figure 4.2: Gantt Chart - 2D Game Design using GameMaker

### 4.2.2 Literature Analysis

Using the established technical requirements as a framework, relevant literature could be sourced and focal topics could be researched. The literature review explored the game

design process and the core elements that go into creating an engaging and functional 2D game, notably: the game mechanics, dynamics and aesthetics. It also emphasised the importance of iteration, simulation and prototyping required for the project, enabling testing, evaluation, and improvement. The game and controller design were developed using established theories and practices as a guide, improving the quality of the final product to meet the deliverables expected from the project.

### 4.2.3 Module Design and Simulation

By dividing the system into two distinct sub modules (Game Design and the Controller Design), an achievable schedule for deadlines was created as seen in fig 4.2. Each process detailed in Chapter 5, involved analysing the technical requirements and finding suitable solutions to meet the projects expectations. Various simulations were conducted throughout the projects life cycle, offering insight into program optimisation, errors in code and were used to develop functional prototypes. Only through the development of prototypes could the user requirements be validated via the experimental testing procedure.

### 4.2.4 Validation

With functional prototypes developed, the modules of the project could be tested. In order to validate the results, acceptance test procedures were established to ensure that the technical requirements 4.2 and thus, the user requirements 4.1 met the project's expectations. The acceptance test procedures can be seen in table 4.3.

Table 4.3: Acceptance Test Procedures

Test	Explanation
ATP1	The game is developed using GameMaker Studio 2.
ATP2	Successful implementation of procedural generation.
ATP3	The difficulty rating of the game is sufficient.
ATP4	Players find the game enjoyable
ATP5	Players consider the game to be aesthetically pleasing with a coherent art style.
ATP6	Code runs smoothly without bugs or lag.
ATP7	The game controller has the functionality to play the game.
ATP8	There is a large enough sample size such that adequate play-testing is conducted.

Following on from the play-testing procedure outlined in chapter 6, 20 individuals were tasked with play-testing the game for a set period of time. Participants were assigned a questionnaire aimed at validating the acceptance test procedures. By analysing the data obtained from the testing procedure, the results confirmed that the technical requirements were obtained, the acceptance tests were met, allowing the project's expected deliverables to be put forward. Moreover, a SWOT analysis identified and evaluated the project's Strengths, Weaknesses, Opportunities and Threats, further providing opportunity for future research around the topic.

The next chapter details the **design and simulation** of the game and controller. Through analysing the technical requirements listed in 4.2, each subsection targets a specific requirement with the goal of meeting the projects expected deliverables.

# Chapter 5

## Design and Simulation

The project is split into two distinct subsystems, each focusing on a specific aspect of the overall design. The **Game Design** subsystem details the development of the game's core elements, including gameplay mechanics, level design, graphics, as well as GUI. Simulations are conducted throughout the subsystem design, in an iterative design process. The **Controller Design** subsystem concentrates on the design and development of the game's controller, focusing on its mechanical, electrical and software elements. As a result of simulation in electrical and 3D modelling software, a comprehensive visualisation of the controller is achieved. The division of subsystems allows for a more manageable and structured approach, guaranteeing that both areas of the project receive the necessary attention throughout.

### 5.1 Game Design

This section aims to break down the technical requirements TR1 to TR5 as mentioned in Chapter 4, with the aim of creating a fun and functional 2D game with sufficient complexity using GameMaker.

- **TR1:** The game must be developed with GameMaker Studio 2.
- **TR2:** Includes complex game mechanics.
- **TR3:** A complex degree of game dynamics.
- **TR4:** The game must have aesthetic appeal.
- **TR5:** Code should be optimized.

In this section, the game design process is detailed, with focus on the game mechanics, dynamics and aesthetics as outlined by Kim [11] and Hunicke [12]. It further explains how meaningful play is brought about, as discussed by Salen [7], and how it can be successfully incorporated into a 2D game. After many iterations and prototypes, as well as optimisation within the code, a final **design** of the **game** can be used to successfully fulfill the technical requirements listed above.

### 5.1.1 Paper Prototyping

Before any programming can be conducted, a basic idea needs to be established of the fundamental mechanics and flow of the 2D game. **Paper prototyping** is a quick and inexpensive method used to convey basic concepts and ideas as highlighted by Adams [6]. By using this method of prototyping, many rapid iterations can be produced, quickly eliminating weaker design concepts and promoting certain design strengths. The final paper prototype after a few iterations is displayed in fig 5.1.

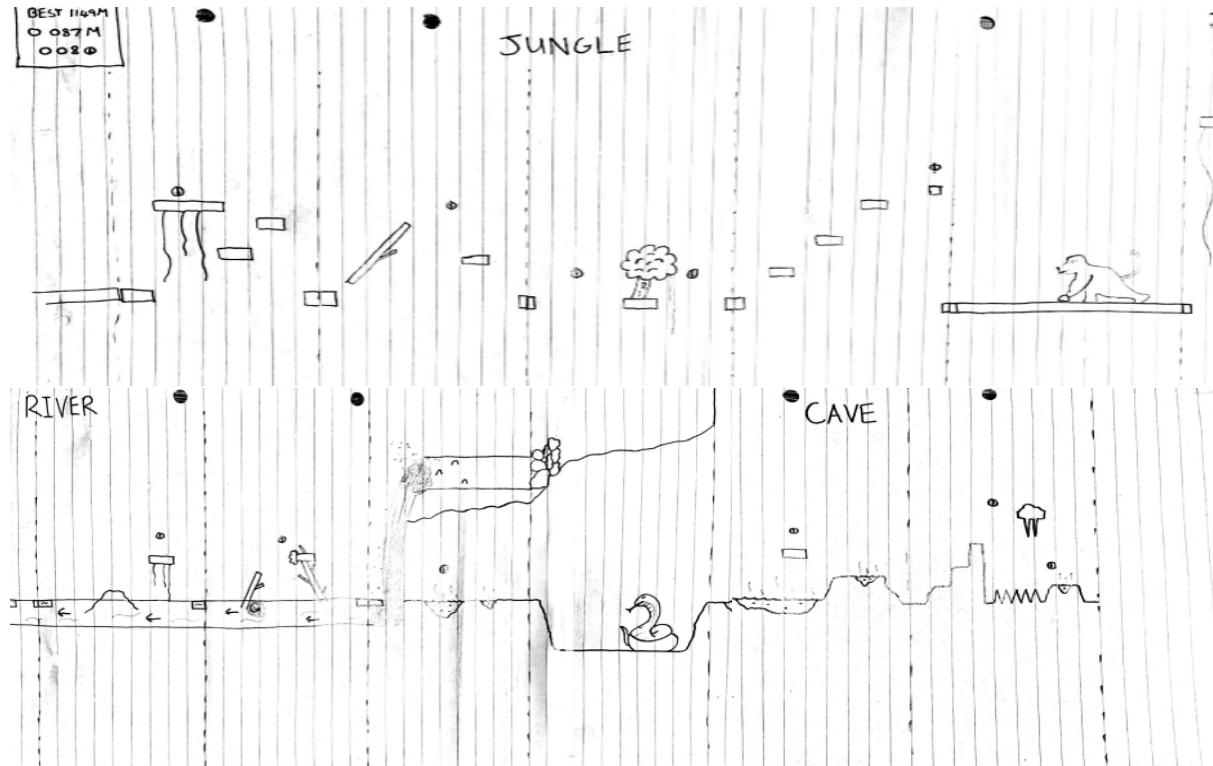


Figure 5.1: Final Paper Prototype

As a result of using this prototype as a foundation, the **Game Mechanics** can be designed.

### 5.1.2 Game Mechanics

The game mechanics refer to the specific rules and systems that dictate a game's operation. Using the paper prototype in fig 5.1, the core mechanics can be developed.

#### Player Movement

The player's movement is governed by two dimensions along the vertical and horizontal axes. The player and all other entities in the game possess the ability to move forward and backwards, as well up and down. **Position, velocity and acceleration** are taken into consideration in order to achieve this motion in two dimensions.

The **position** within GameMaker is represented in the coordinate system of x and y. Each unit along the axes represent a pixel on the screen, as seen in fig 5.2.

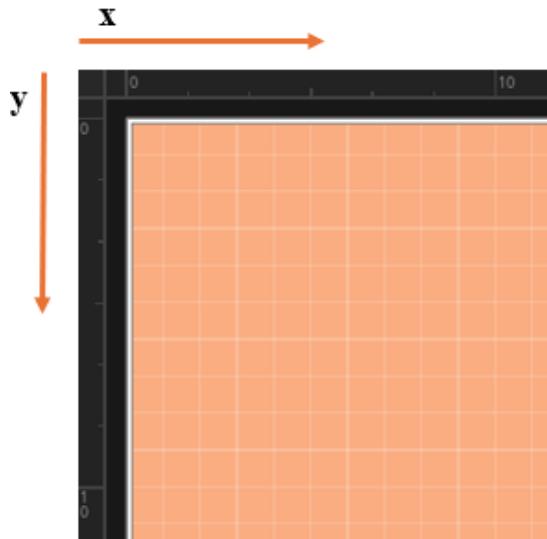


Figure 5.2: GameMaker Co-ordinate System

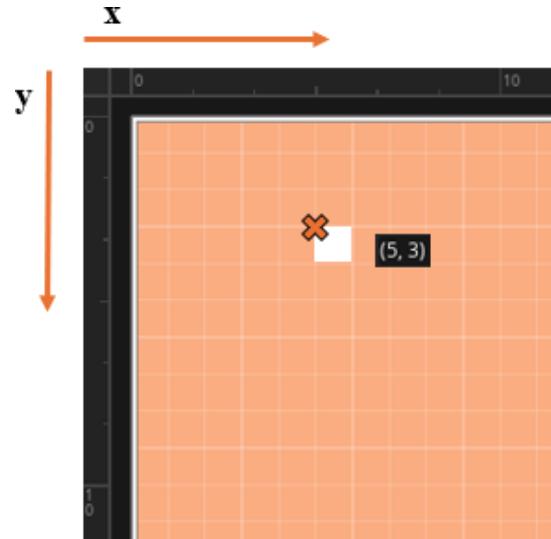


Figure 5.3: GameMaker Co-ordinate System at a position

Each block seen in fig 5.2 represents a pixel on the screen. This grid system allows the **position** of objects to be defined relative to both the game world, as well as other entities within it, using the number of pixels as a metric. GameMaker's coordinate system defines the positive y direction from the top of the screen down. An example of how a **position** is defined can be seen in fig 5.3, depicting a pixel on the screen at position (5, 3).

In GameMaker, there exists a constant update loop known as the 'step' function. This step function, runs at approximately 60 frames per second (fps) and is primarily used to create the illusion of smooth movement, however objects are not moving continuously.

Objects are shifted from one pixel's position to the next, creating the perception of fluid motion. Consequently, the position of objects in the game can be updated in each frame with their own custom scripts. An example of the player's horizontal **velocity** is displayed in equation 5.1.

$$x\_speed = (right\_key - left\_key) * move\_speed \quad (5.1)$$

This code determines the player's x velocity by taking the difference between the left and right arrow keys, and further multiplying the value by a predefined movement speed. The variables 'right\_key' and 'left\_key' return a value of 1 when pressed and 0 when released. As a result, the player's x velocity can be positive or negative based on the equation 5.1. This value is then added to the player's current x position, resulting in a change of the player's x position in the next update of the loop, as illustrated in equation 5.2.

$$x = x + x\_speed \quad (5.2)$$

By placing these two equations in the step function, the player's x position is updated in every frame based on the corresponding input from the user.

The player's vertical velocity is calculated slightly differently. In order to achieve a more realistic movement mechanic, **acceleration** along the vertical axis is required. This is achieved by applying a constant value to the current downward velocity in every frame, resulting in a constant acceleration downwards. This constant value is a predefined 'gravity' value. In addition, the code checks if the player is on the ground (checks if there are terrain blocks directly below the player), in which case the downward velocity is reset to zero. This process is illustrated in the pseudo code below.

---

#### Pseudo Code for Y Velocity (every frame)

---

- 1: Y Speed increased by gravity value
  - 2: **if** Player is grounded **then**
  - 3:   Y speed set to 0
  - 4: **end if**
- 

With this constant downward acceleration, the player is constantly brought to the surface of the terrain. By applying a vertical y velocity in the same way as presented in equation 5.1, the player is triggered to jump with user input, soon to be brought back to the surface by the constant downward **acceleration**.

## Camera Movement

A powerful feature within GameMaker is the ability to perceive the game world through a view port, acting as the game's **camera**. A specific game mechanic developed in the paper prototyping phase, is the constant movement of the camera used to add a sense of time pressure for the user. This mechanic enhances **meaningful play** within the game, outlined by Salen [7] in Chapter 2.

This constant **camera movement** is achieved in the same way as shown in equation 5.2, constantly incrementing the camera's x position every frame. The code also reviews whether the player's x position is greater than it's own, adjusting to match the player's velocity. This mechanic allows the player to move faster than the camera but never slower, creating a sense of time pressure for the user. This is displayed in the pseudo code below.

---

### Pseudo Code for Horizontal Camera Movement (every frame)

---

```

1: if Player's x position > camera's x position then
2:   camera's x position move's to player's x position
3: else
4:   camera's x position is incremented by the movement speed
5: end if

```

---

To create the illusion of the player moving through the terrain with a distant background, the **background layer** is constantly moving. If the background remained stationary relative to the player, it would give the impression that the background exists on the same depth layer as the player, which eliminates the sense of perspective. Instead, the **background** moves in the direction of the camera, based on the camera's velocity. This is illustrated in the pseudo code below.

**Note:** The pseudo code presented in these snippets are aimed at explaining relevant logic. The code used in the game combines code that follows similar logic into the same 'IF and ELSE statements' for efficiency and system performance.

---

### Pseudo Code for Horizontal Background Movement (every frame)

---

```

1: if Player's x position > camera's x position then
2:   background speed increases
3: else
4:   background speed decreases
5: end if

```

---

When the player's x position exceeds that of the camera, the camera speeds up, causing the **background** to accelerate. Furthermore, a Gaussian Blur effect is also placed over the background layer, mimicking the way in which a real-world camera experiences blur,

due to focus and movement.

**Note:** The background speed is set to be **slower** than the camera at all times, enhancing the sense of **depth** and **motion** for the user.

## Procedural Generation

Generating the terrain for each individual level is a tedious and time-consuming task. This issue can be resolved through addressing the complex engineering problem of **procedural generation**. A significant advantage of **procedural generation** as discussed by Togelius [26], is its capability to create a wide variety of terrain almost instantly.

The idea behind the **procedural generation** of the game is looping through all possible block positions, placing or removing blocks of choice. This is achieved through two main 'for loops' that iterate through all x and y coordinates in the game world, placing and removing content along a pseudo random path. An example of how this is achieved is shown in the pseudo code below.

**Note:** The **block size** used in the 'for loops' is set to 16, representing blocks of size  $16 \times 16$  **pixels**.

---

### Pseudo Code for Procedural Generation For Loops

---

```

1: for x = starting x to ending x step block size do
2:   for y = starting y to ending y step block size do
3:     Procedural Generation Instructions...
4:   end for
5: end for
```

---

As it is possible to map every position in the game, the instructions for each point must be set. The **control** over the procedural generation process is vital for creating successful terrain. By creating a function that takes in set **parameters**, as outlined by Kelly [3], the procedural generation function then has the foundation to implement its logic. The parameters used in the procedural generation function are outlined in tab 5.1 and **control** how the terrain is created.

Table 5.1: Description of Parameters

Parameter	Description
<b>startX</b>	The starting x coordinate of the generated terrain.
<b>endX</b>	The ending x coordinate of the generated terrain.
<b>y_offset</b>	The vertical distance from the top of the screen down to the first surface block of the terrain.
<b>amplitude</b>	The change in vertical height from one x coordinate to the next.
<b>flatScale</b>	The frequency in which the amplitude is changed.
<b>blockSize</b>	The size of each generated block.
<b>Depth</b>	The total depth of the generated terrain.
<b>Seed</b>	The pseudo random number generated from a particular configuration of terrain.

Using these **parameters**, the function uses the two 'for loops,' generating blocks in conjunction with the instructions laid out. A basic example is presented in fig 5.4, displaying the logic flow required to generate a simple procedural terrain.

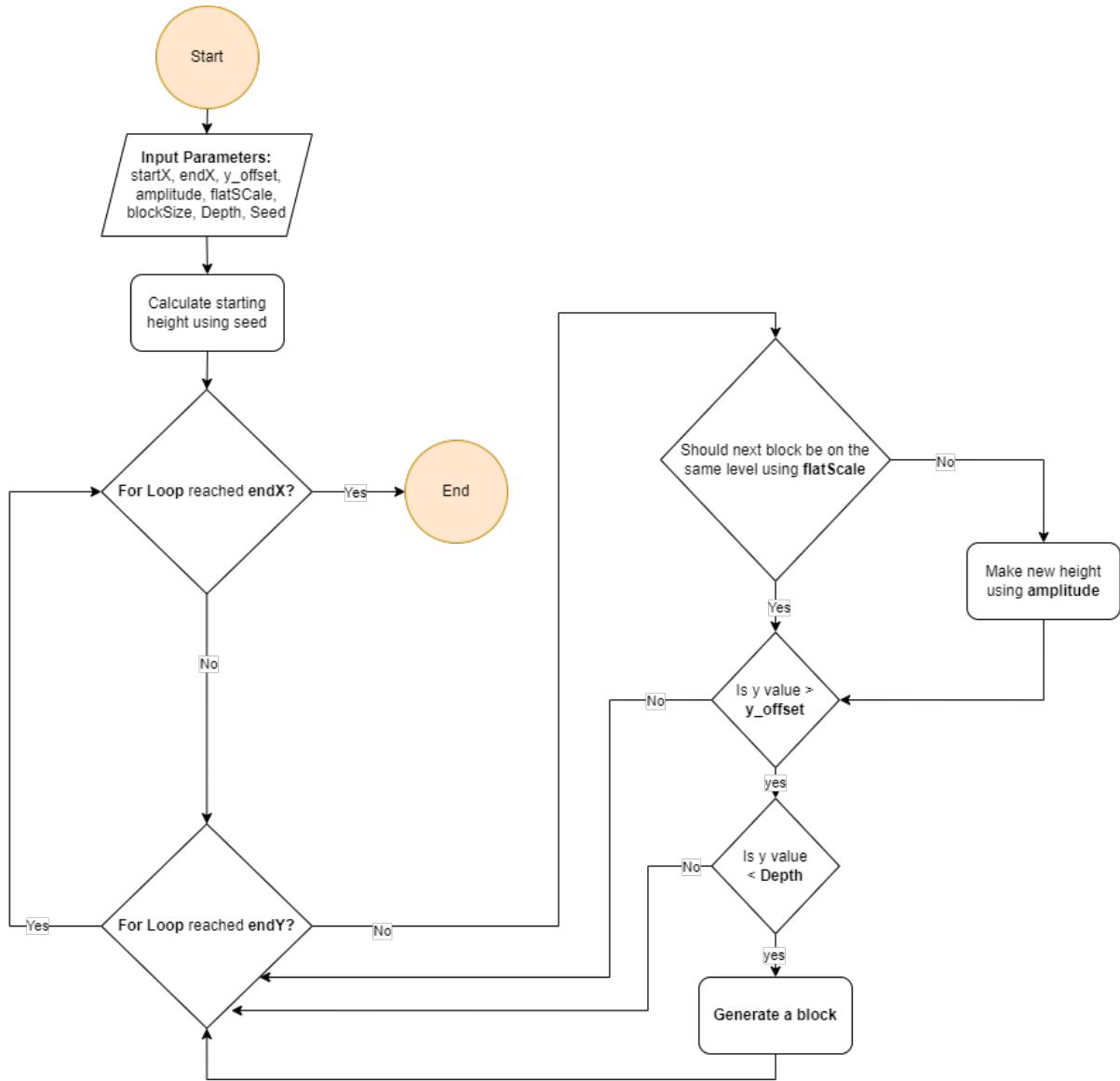


Figure 5.4: Procedural Generation Flowchart Example

Using the flowchart in fig 5.4 and the parameters listed in tab 5.2, the output is displayed in fig 5.5.

Table 5.2: Parameter Values

Parameter	startX	endX	y offset	amplitude	flatScale	blockSize	Depth	Seed
Value	0	400	100	4	6	16	300	11

The parameters of the function can be set by calling the function as shown in fig 5.4. These values can be adjusted accordingly to obtain new terrain layouts. The output of the function with these particular parameters is seen in fig 5.5.

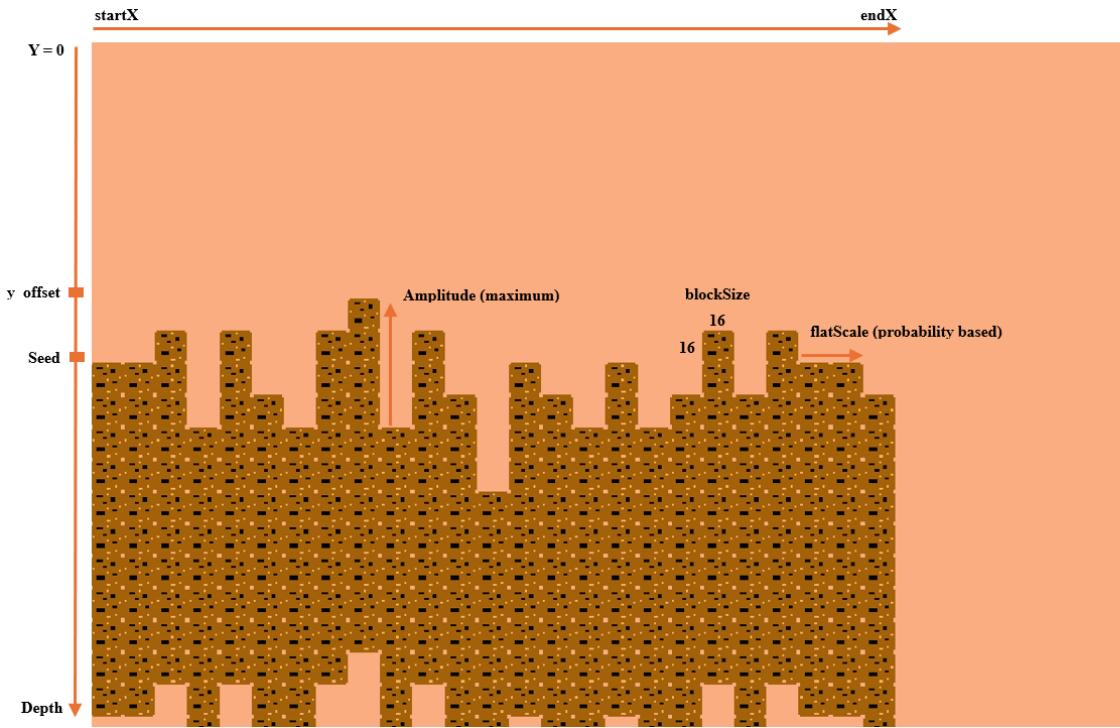


Figure 5.5: Procedural Generation Output

The beauty of this function lies in its ability to accept new parameters, allowing for the terrain to constantly generate past the initial end point. By updating the starting and ending x coordinate, this terrain can be generated indefinitely.

As a result of using pseudo random numbers, as discussed by Vadhan [27], the effect appears random, however it follows a deterministic process. Furthermore, using the **pseudo random** functions in GameMaker as probabilities, effectively 'randomises' the appearance of the terrain. Through many **iterations** and **simulations** of the procedural generation, the design of the function can be further developed. By incorporating more rules to determine the placement of new terrain blocks, the user is provided with a more authentic and detailed experience of the terrain.

The design of the **cave system** in the game, is a challenging endeavour. Using this method of procedural terrain generation requires a second function that iterates through the terrain, removing certain blocks to create the appearance of caves. The relevant issue is that blocks that do not exist yet cannot be removed, as the cutout will then be covered by blocks later on in the function. This is demonstrated in fig 5.6.

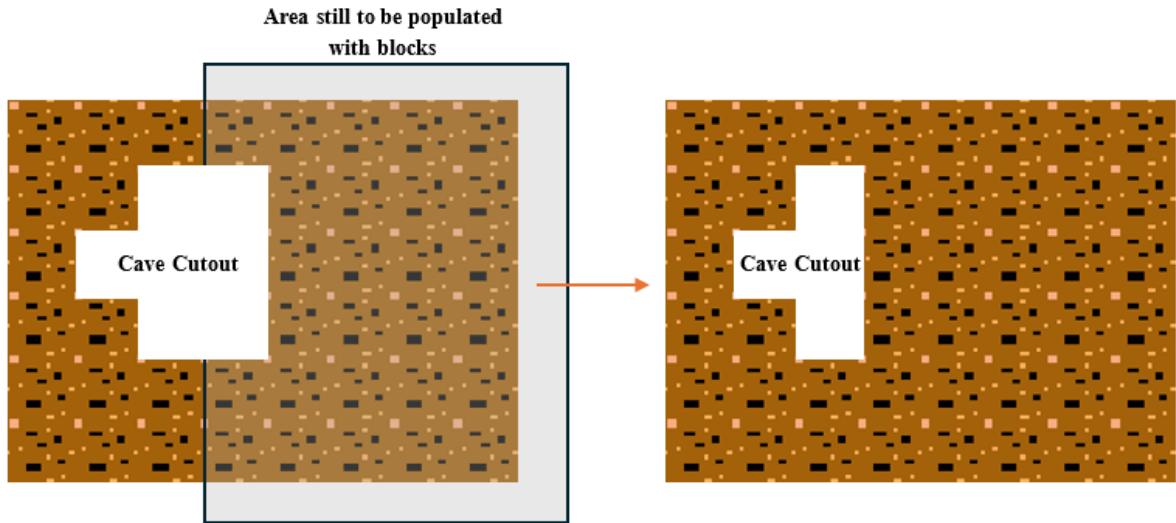


Figure 5.6: Issue presented with removing blocks that have not been populated

To address this issue, the 'for loops' iterate back towards blocks that have already been populated to prevent blocks being placed over the cave cutout. As the 'for loop' progresses through the x and y coordinates of the procedural terrain function, it is instructed to generate a cave at a designated point. The logic flow for creating a basic cave cutout is shown in fig 5.7.

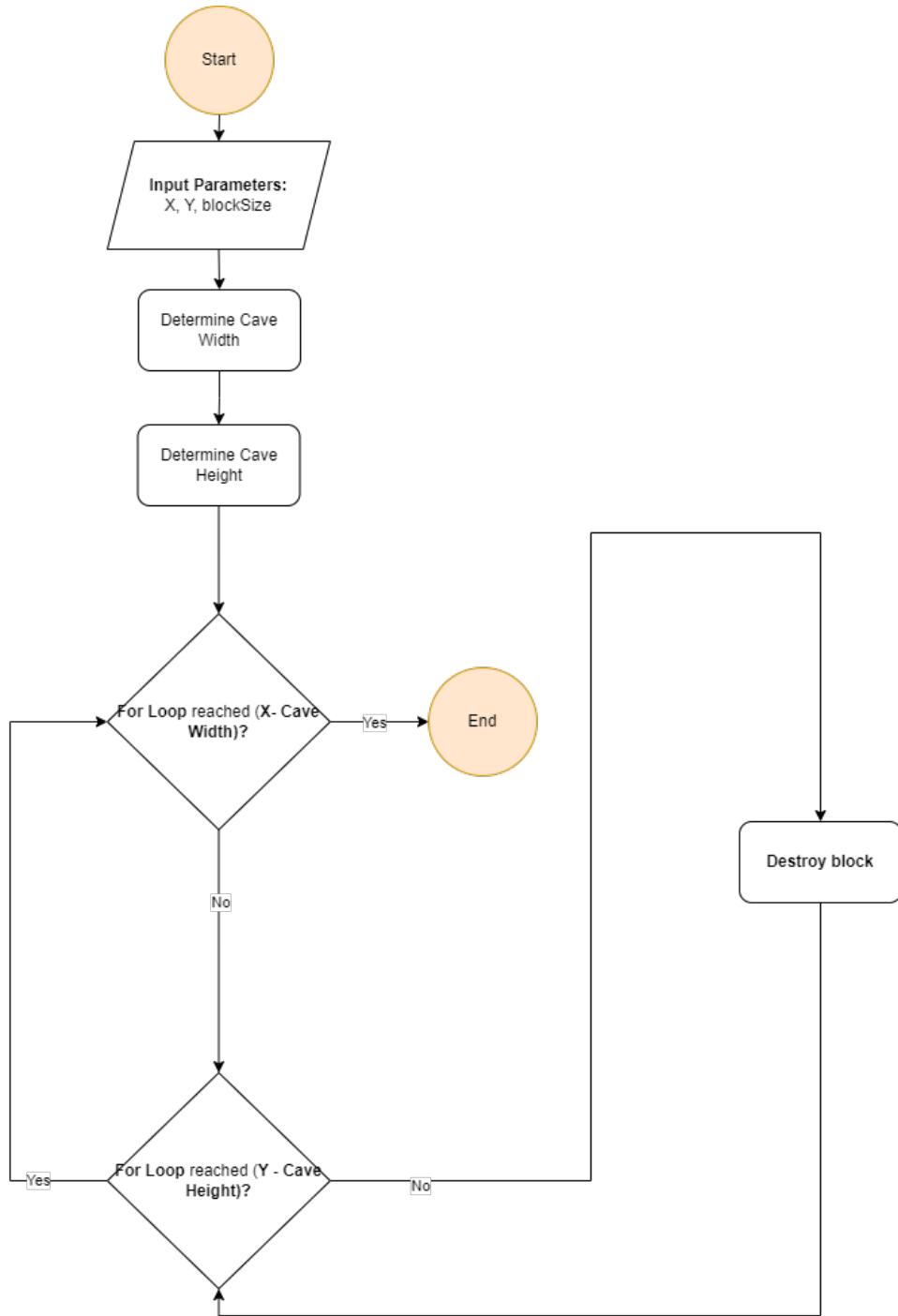


Figure 5.7: Procedural generation flowchart for creating a cave

The width of the cave is generated using pseudo random values, increasing as the depth from the surface grows. Consequently, allowing for a smaller opening at the mouth that gradually widens into a larger underground area. The resulting output is displayed in fig 5.8.

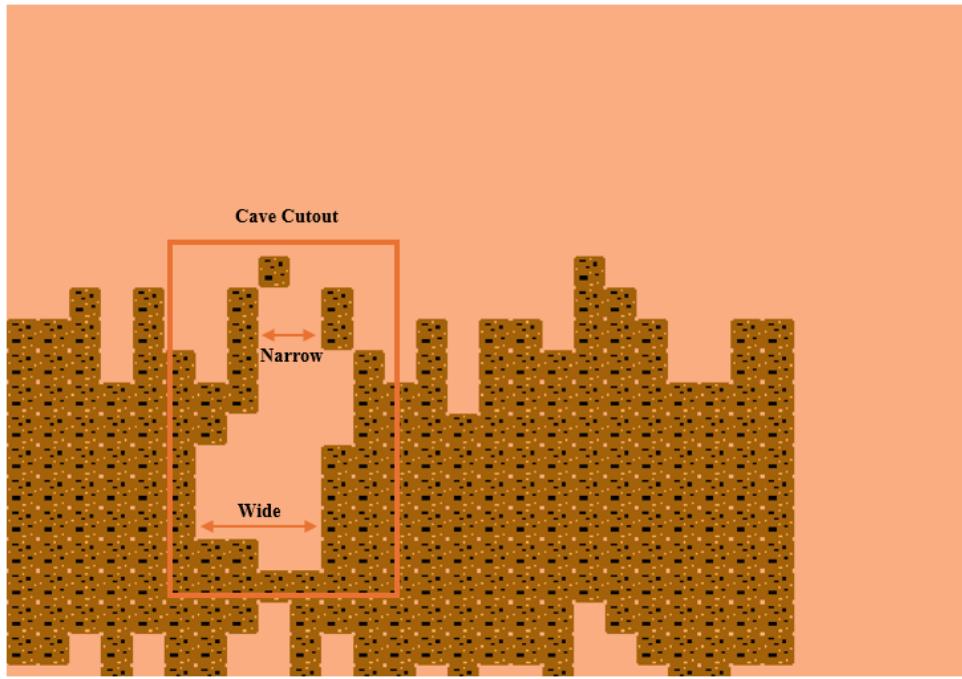


Figure 5.8: Procedural generation cave output

Once the caves have been generated, an **outline** must be created around the perimeter in order to ensure that the player has a defined path for exiting the cave upon entry. Through many iterations and prototypes, a concept emerges that enables the player to stick to cave walls while sliding down. To implement this, a function is developed to detect whether blocks along the cave perimeter have been populated. The function generates blocks, ensuring that the player has a path to escape the cave, as displayed in fig 5.9.

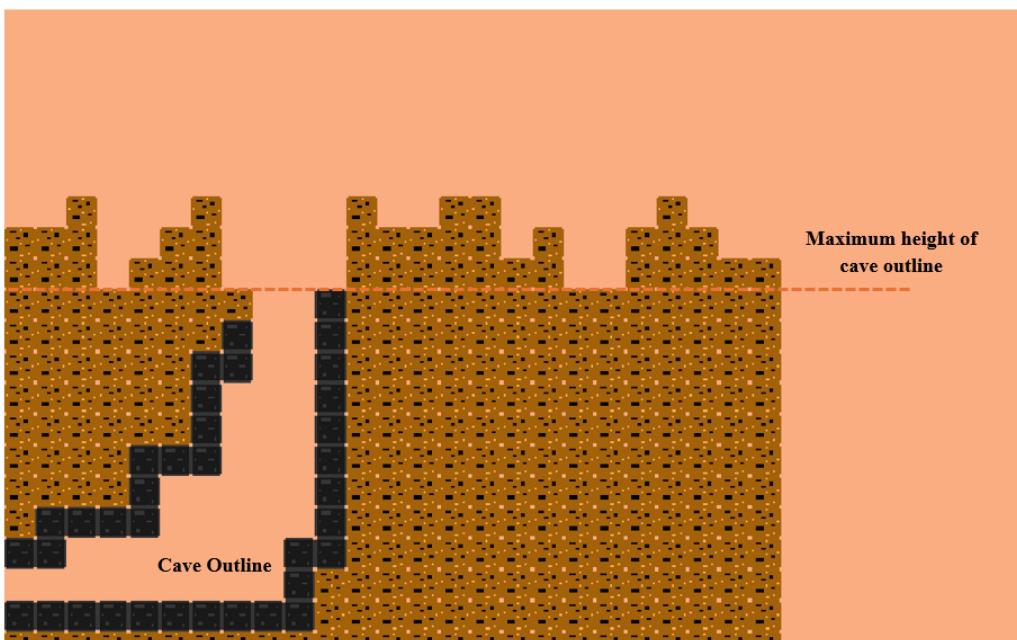


Figure 5.9: Procedural generation cave outline output

This mechanic provides the ability to scale large cave walls by jumping from side to side. This is obtained through adjusting the gravity value present in the game. While the player is on the cave wall, in water, or on a vine, the gravity value is adjusted to give the game a more realistic appearance.

The **mechanics** of the game incorporates; movement mechanics along the horizontal and vertical axes, procedural terrain generation, dynamic cave systems, as well as adjusting gravity for different surfaces. Iterating through prototypes and refining ideas, provides opportunity for the core mechanics to be solidified. The game's mechanics, are the specific rules and systems that dictate the game's operation, and are carefully crafted to ensure smooth and immersive player experiences. From generating terrain to refining player interactions, every aspect of the game's core mechanics contribute to the overall flow and playability of the game.

### 5.1.3 Game Dynamics

Game dynamics refer to how the **game mechanics** interact and evolve over time, based on the player's input. This also includes how the **player** responds to these mechanics. In addition, the game's dynamics are categorised into three primary areas: **Level Progression**, **Enemies**, and **Virtual Currency**, each featuring their own distinct functionalities.

#### Level Progression

The progression of the levels stem from the paper prototype illustrated in fig 5.1. This game dynamic provides a sense of accomplishment for the user and offers an incentive to engage with and complete the game. By combining the mechanics involved in the formulation of the game, multiple game dynamics arise, in turn allowing for variations within the design of each level.

By adjusting the parameters of the procedural generation function seen in fig 5.1, the complexity of the game can be increased with each subsequent level. This allows for an evolving gameplay experience that challenges players and encourages the development of new strategies and skills as they advance. By using the procedural generation function as the core game mechanic, the design of a particular level is never repeated. This provides players with a dynamic **Level Progression** that offers a fresh and rewarding experience

whenever a level is restarted or the next level is obtained.

The game is comprised of four distinct levels, each containing their own difficulty rating and terrain elements. This can be seen in fig 5.3

Table 5.3: Game Levels and Difficulty Ratings

Level	Difficulty
Tutorial Level	Very Easy
Jungle Level	Easy
Desert Level	Moderate
Volcano Level	Hard

Difficulty is measured by the amount of time required to pass each level, as well as the average number of deaths each user experiences in attempting to complete each level. This structure gradually introduces the player to the game mechanics while maintaining engagement through increasing challenges. This ensures that the gameplay remains captivating and rewarding throughout the duration of the game.

The **tutorial level** is designed to guide users who are unfamiliar with 2D games. Clear and concise instructions are provided at each stage, ensuring that the user grasps the basic concepts of gameplay. Figures 5.10 and 5.11 offer basic examples of these instructional elements in action.

This introductory **tutorial level** sufficiently prepares the user to attempt the next three levels of the game. As the levels progress, so do their respective difficulties with more challenging terrain, obstacles and enemies standing in the way of the player's path to success.

## Enemies

Movement patterns, health systems and attack strategies of enemies falls under game mechanics. However, how the player responds to these enemies and the evolution of the strategy involved is part of the game's **dynamics**.

Each enemy or trap in the game is scripted to inflict damage to the player upon contact. Collision detection in the update loop allows for the aforementioned. When a collision is detected, a collision flag is triggered, resulting in a decrease in the player's health value. An example of this is seen in the pseudo code below.

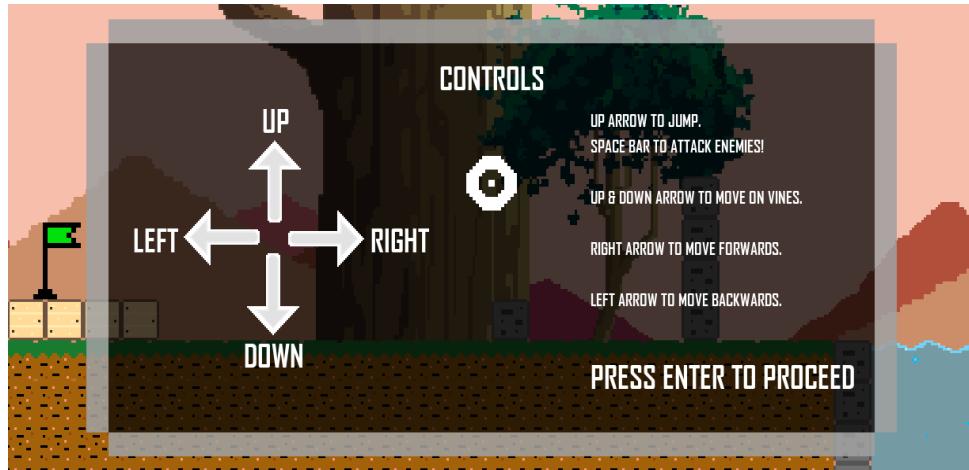


Figure 5.10: Tutorial Controls

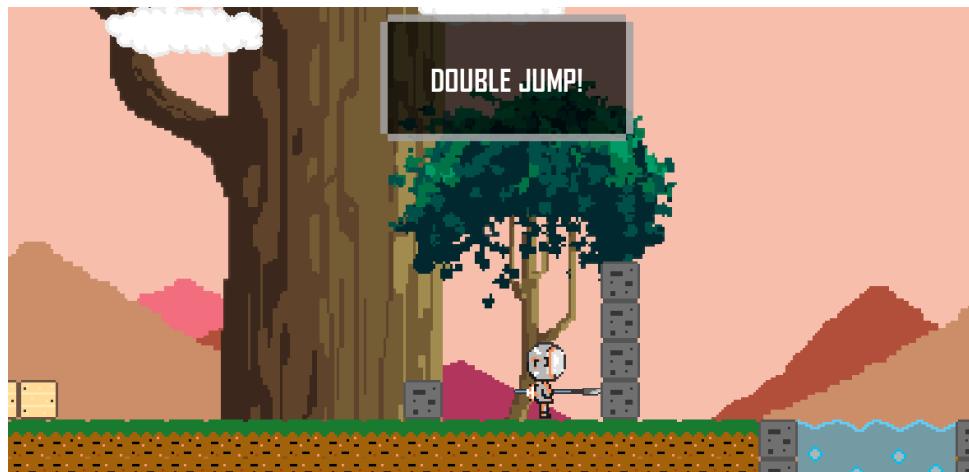


Figure 5.11: Tutorial Instructions

---

Pseudo Code for Horizontal Background Movement (every frame)

---

```

1: if a pixel on the player is in contact with an enemy pixel then
2:   Set collision flag true
3:   decrease player's health by 1
4: end if
  
```

---

The traps and enemies in the game are designed to promote **meaningful play**. The user adapts their style of gameplay to adapt to various enemies, the number of traps, as well as the movement patterns of various hazards. At the end of each sub-level, the player encounters a primary enemy, known as the **final boss fight**, which must be defeated in order to progress. Each boss is equipped with a custom script aimed at increasing the difficulty of each subsequent level, making it progressively harder for the player to advance. The scripts for each of these bosses follow the same core structure illustrated in fig 5.12.

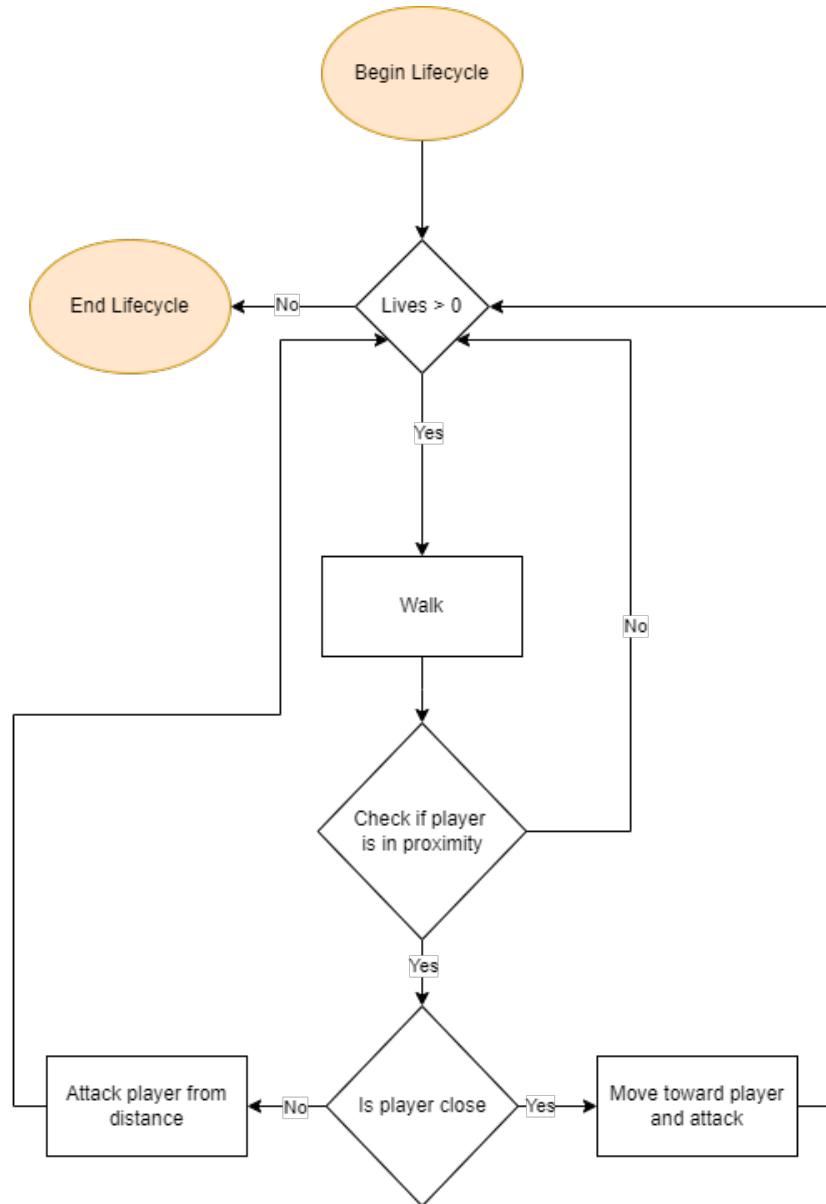


Figure 5.12: Enemy Life-cycle

By adjusting the health, speed and frequency of attacks, more formidable enemies can be faced, forcing a **dynamic** approach from players attempting to pass each level.

### Virtual Currency

In order to instill player engagement, a certain level of incentive is required. By implementing a **virtual currency**, players are motivated to progress through levels, explore the game world, and ultimately complete the game. This system encourages continued play by offering rewards and goals, that keep the experience **rewarding** and **dynamic**.

This is achieved through creating a virtual store where players can purchase new skins and weapons. Players that wish to obtain a different appearance or unlock more lethal weapons, must collect coins and diamonds throughout the game. Coins are scattered commonly throughout the maps and are obtained through time spent playing the game and repetition of levels, building up coin count. Diamonds however, are rarer and are collected through defeating bosses and exploring cave structures. This promotes heightened player engagement and exploration, keeping players motivated to continue.

### 5.1.4 Game Aesthetics

The game aesthetics play an important role in shaping the player's experience by defining the atmosphere, visual style and tone of the game. The game aesthetics are made to not only make the game visually appealing but are a fundamental part of immersion and guiding player experiences. In an engineering design context, the aesthetics must balance form and function. Visual elements are not only appealing but communicate game mechanics, feedback and objectives clearer to the user. A well designed Graphical User Interface, cohesive visual elements, and intuitive layout make it possible for a more engaging and accessible game. From an engineering perspective, integrating aesthetics into the design process requires considering how each graphical element contributes to both user experience and system performance. Larger, more detailed artwork is more CPU intensive, as rendering graphics requires processing power. This makes it important to balance visual aesthetics with system performance to ensure smooth gameplay without straining the hardware.

Initially, the design of graphics is very detailed, allowing for an improved player immersion experience. As the game gets larger, and the procedural generation function produces many blocks, system performance begins to decrease. Loading time is increased as levels take longer to load and the user experiences lags during gameplay as the processor is working hard to render these graphics. Considering these aspects, a design must be chosen that meets the following requirements:

- Improved System Performance
- Reduced Time per Image
- More Achievable Cohesive Layout

With a reduced number of pixels per image, the system performance is greatly improved.

This is tested through producing two simulations. Simulation one involves creating an idealistic,  $1024 \times 1024$  size image, downsized to a  $16 \times 16$  size, displayed many times, at random positions on the screen. Simulation two involves the same process, with the exception of displaying a  $16 \times 16$  image at regular size. This simulation can be seen in fig 5.13.

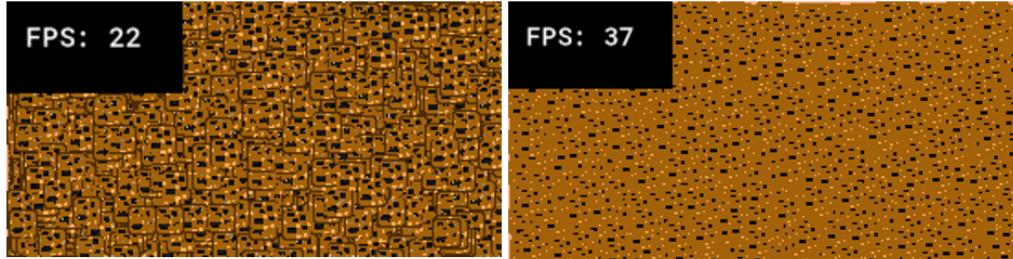


Figure 5.13: Image Size fps comparison

The 'FPS' seen at the top left of the two figures represents the rendering time GameMaker is able to produce. A lower fps indicates that more processing power is being consumed, as the system works harder to render each frame. Conversely, a higher fps reflects improved system performance, with the processor handling the rendering tasks efficiently. As can be seen in fig 5.13, the lower resolution  $16 \times 16$  performs significantly better under these conditions, highlighting the importance of lower resolution images in **optimization**.

Creating high-detail graphics is often a time consuming process however, lower resolution images reduce the time required for creating each image. With a pixelated art style, an achievable, more cohesive game layout can also be achieved within the specified time frame. For these reasons, a low resolution  $16 \times 16$ ,  $32 \times 32$  and  $64 \times 64$ , size images are used for the final design. A list of sprites (images used in the game) are presented in figures A to H. These are not all sprites used in the game, but serve to represent the art style, low resolution and tone of the game.

**Note:** Almost all the sprites created are produced in an original manner. Some sprites are derived through inspiration from others as cited in the Bibliography however, no AI or copy/pasting is used.

## 5.1. GAME DESIGN



Figure A: Blocks



Figure B: Virtual Currency

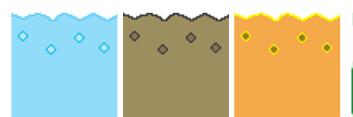


Figure C: Fluids



Figure D: Icons



Figure E: Level Icons



Figure F: Traps



Figure G: Weapons

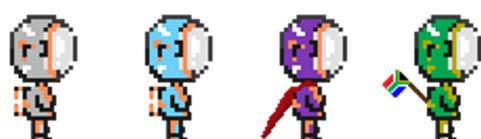


Figure H: Player Skins

Figure 5.14: Sprite Examples

Animating each of these sprites requires time and processing power. Due to the low resolution images, this time and power is significantly reduced. The animation in GameMaker

revolves around frames. Each sprite can contain many frames, and are played at an adjustable fps (frames per second). By making small adjustments to a sprite every frame, a visually appealing graphic can be created as demonstrated in fig 5.15.



Figure 5.15: Sprite Animation Example

Through reducing sprite resolution, the system can perform significantly better during the game's loading phases and gameplay. This optimization not only enhances overall performance but also shortens the time needed to create assets, resulting in a more cohesive and visually consistent **game aesthetic**.

## 5.2 Controller Design

This section addresses requirement TR6 by discussing the design and development of a gaming controller. Although the development of a gaming controller is outside the scope of this project, it has been included to enhance the 2D gaming experience.

- **TR6:** The game should incorporate multiple input systems and be playable on most computers.

The design of the controller aims to enhance the 2D game by combining the old-school 2D platformer experience with a modern and familiar style of console experience commonly used in the twenty first century. As the game's controls are characterised by a simple nature, the controller features a similar approach only placing buttons where necessary.

### 5.2.1 Paper Prototype

The design of the controller is aimed towards enhancing functionality and user experience. A few iterations of the early phase designs are seen in fig 5.16.

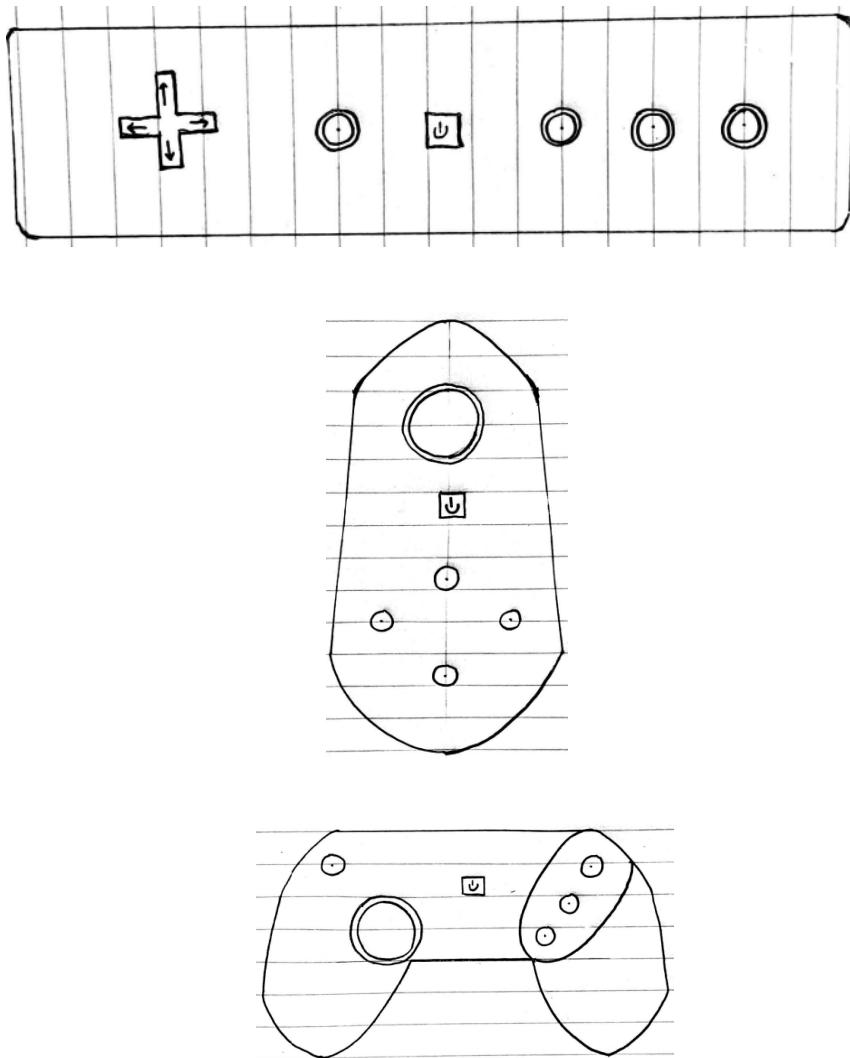


Figure 5.16: Controller Paper Prototypes

**Paper Prototype 1** draws inspiration from the well known 'Nintendo Wii' controller. This simplistic model contains the relevant functionality needed for gameplay. It's familiar style and appearance make it intuitive for users to grasp the controls with very little practice. However, a drawback to this design is the lack of comfort in the hands of the user during gameplay.

**Paper Prototype 2** features an original design aimed at exploring new controllers used in 2D games. With a joystick and relevant buttons, it is possible to play the game easily with the controller's design. However, similar to prototype 1, a drawback to this design is its lack of comfort in the hands of the user.

**Paper Prototype 3** is derived from the well known 'PlayStation' and 'Xbox' controllers. The design has been modified to meet the specific requirements of the game, allowing for

familiarity and comfort during gameplay. For these reasons, Prototype 3 serves as the foundation for the **controller design**.

### 5.2.2 Component Selection

A crucial aspect of the design process is selecting the correct components for the controller. Each component must fit within the controller's casing, and the microcontroller must have the functionality of a keyboard. In addition, buttons should be designed for user comfort to elevate the overall experience.

#### Micro controller Selection

Various types of microcontrollers can be used in the controller design. Two that stand out for this task are the Raspberry Pi and Arduino due to their cost-effectiveness and size. A key factor in this selection process is whether the microcontroller features native HID (Human Interface Device) functionality and real-time performance.

The **Raspberry Pi Zero** is a compact and powerful microcontroller, capable of functioning as a keyboard and performing the necessary tasks. The Raspberry Pi has superior computing power, which is used for more complex inputs and displaying graphics, however, it sacrifices real-time performance to achieve this. The **Arduino Pro Micro** is another viable option, offering native HID support and a more compact design. While it has less computing power than the Raspberry Pi, its real-time performance is greater. Figure 5.17 displays the dimensions of both the Raspberry Pi Zero and Arduino Pro Micro.

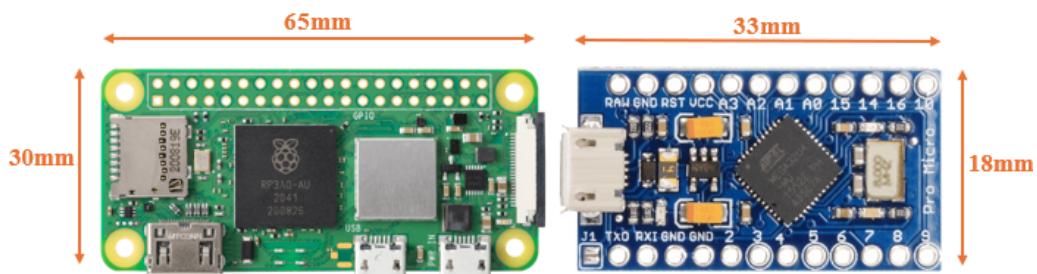


Figure 5.17: Micro controller dimensions

The size of these microcontrollers is a critical factor in the selection process. Figure 5.18 illustrates the usable space within a standard PlayStation 4 controller.



Figure 5.18: Usable Space in a standard PlayStation 4 Controller

The Arduino Pro Micro is the more space efficient choice as the usable height shown in fig 5.18 is exceeded by the Raspberry Pi Zero. The Arduino also offers superior real-time performance, which is crucial for the controller's functionality in responding quickly. As a result, the **Arduino Pro Micro** is selected as the most appropriate for the final design.

### Button Selection

Choosing the correct push buttons is essential for the overall comfort and usability of the controller. Three types of push buttons are evaluated, as illustrated in Figure 5.19, along with an analog joystick used for the design.



Figure 5.19:  
Mini  
SMD  
Momentary  
Switch



Figure 5.20:  
Switch  
with  
LED  
Circle  
pattern



Figure 5.21:  
Momentary  
Push  
Button  
Switch  
Horizontal



Figure 5.22:  
Analog  
Joystick  
Module  
with  
Switch

The tactile qualities of each push button can only be assessed through practical testing. Consequently, all push buttons are considered until they undergo evaluation in the **Prototyping Phase**. Furthermore, the joystick module is essential for player movement.

A standard PlayStation joystick is used to improve familiarity of the controller for the user.

### 5.2.3 Mechanical Design and Modelling

To aid in the design and simulation of the controller, the online 3D modeling software OnShape is chosen. This software allows for a full visualisation of the model and provides the ability to assemble different parts in order to view the controller's final assembly.

The **mechanical design** uses a base model by Amr Ashraf sourced from GrabCad.

#### [GRABCAD MODEL](#)

This model is modified in accordance with the paper prototype, to ensure the controller requirements are met, as seen in fig 5.23.

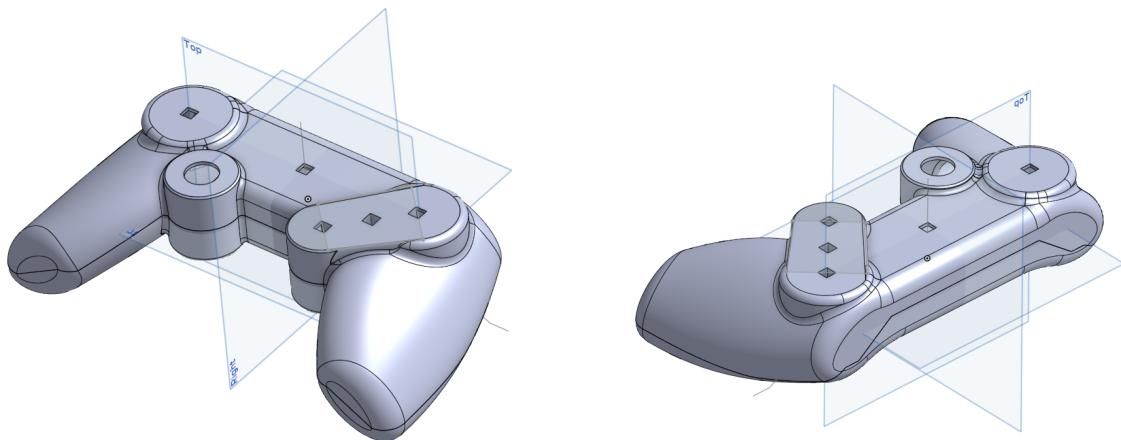


Figure 5.23: Full Model Iteration 1

### 5.2.4 Electrical Design

The design of the electronics within the controller give it the ability to interface with the computer. A full schematic of the **Electrical Design** is shown in fig 5.24.

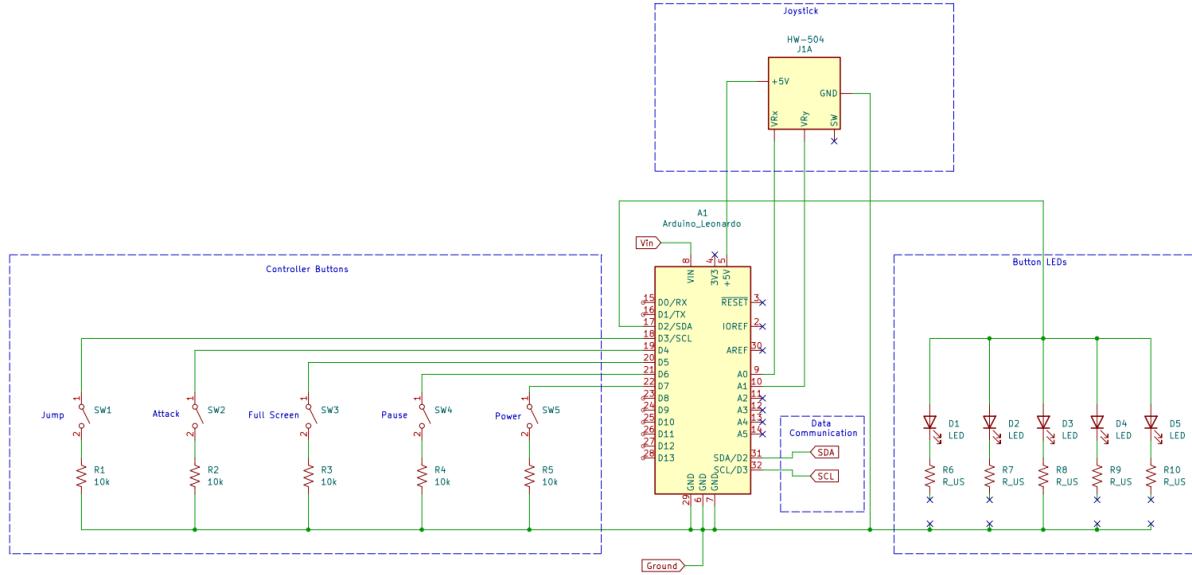


Figure 5.24: Controller Electrical Schematic

Pins D3 to D7 are connected to pull up resistors. When a push button is pressed, the value read at the pin is pulled down to zero through the connected circuitry. Each of these pins write a letter to the computer upon being pressed. The joystick is powered by the board, using 5V and ground for reference. When the potentiometers within the joystick detect movement along the X or Y axes, the value read at pins A0 and A1 change, resulting in the outputs left, right, up and down arrows being sent to the computer. Each push button is fitted with an LED. However, only the push button used for turning the power on within the controller is connected, serving to indicate that the controller has successfully powered up. Furthermore, the complete code used on the Arduino Pro Micro is presented in Appendix B.

### 5.2.5 Physical Prototyping

The controller design is implemented using a Prusa Mini 3D printer. This provides a cost-effective method to prototype designs and identify flaws, errors, and areas for improvement.

### 5.2.6 Prototype 1



Figure 5.25: Controller Prototype 1

The purpose of **Prototype 1** is to review broad categories, such as: shape, tactile feel, buttons placement and overall strength. Following on from this, the shape and tactile feel, and overall strength met the relevant criteria. However, the button placement is modified slightly to ensure user comfort. Through this prototype, the Switch with LED Circle pattern as displayed in fig 5.20 is determined to be the most suitable choice for the controller. This leads to **Prototype 2**, involving these minor adjustments in the next iteration.

### 5.2.7 Prototype 2

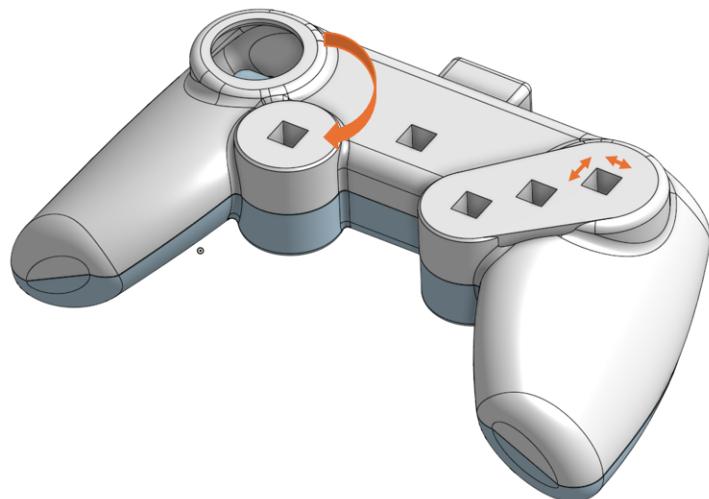


Figure 5.26: Controller Design Changes



Figure 5.27: Controller Prototype 2

Prototype 2 addresses the previously identified design flaws. The push buttons, Arduino Pro Micro and PCB of the joystick fit comfortably into the controller, however, a final design flaw is brought to light. The electrical wires within the controller cannot be accommodated comfortably within the housing. This leads to the final iteration of the practical prototype.

### 5.2.8 Prototype 3

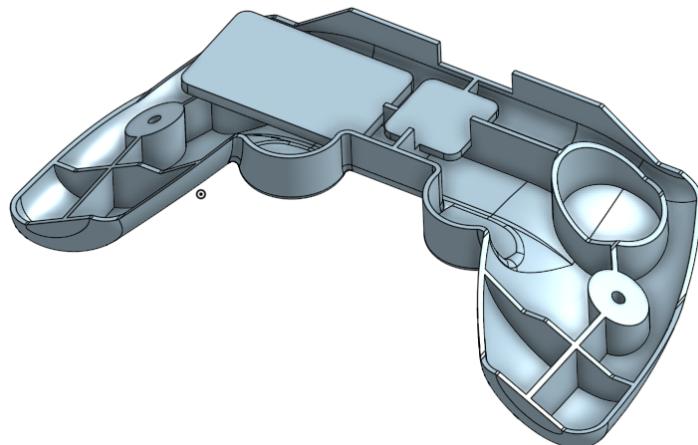


Figure 5.28: Controller Design Final Changes



Figure 5.29: Controller Prototype 3

This final prototype addresses the minor design flaws identified in Figures 5.25 and 5.27 to ensure a successful assembly of the controller. Structures within the controller are removed and platforms are added to better support components. The design is printed using a finer (0.1mm) thickness, improving the aesthetic appearance and tactile feel. This represents the final design of the controller used in the **experimental testing procedure** in the next chapter.

By evaluating the core mechanics of the game, movement and procedural generation were emphasized as some of the most critical elements. Player movement is the fundamental mechanic of the game and was thoroughly detailed in section 5.1.2. The complex engineering problem of **procedural generation** was explored in section 5.1.2, emphasising the importance of key parameters, control over the function, and the effect pseudo-randomness plays in generating effective terrain. Through an iterative process involving many simulations, the code was refined, optimized, and successfully crafted to generate a wide variety of terrain. The game dynamics in section 5.1.3 were also detailed, showing the culmination of mechanics involved to create each game dynamic. Consequently, meaningful play was discussed, and shown to emerge from the interaction between mechanics, dynamics, and player choice. Additionally, the discussion regarding the game's aesthetics in section 5.1.4 highlighted the importance of balancing form and function. The aesthetics define the atmosphere, visual style and tone of the game, but must consider the cost on system performance, as well as production time. Finally, the design of the controller was also covered in section 5.2, offering insight into early phase paper prototypes, component selection, 3D modelling, electrical modelling and prototyping. Through the iterative process described in this section, design flaws were eliminated, converging on the final design. The next chapter focuses on the **experimental testing procedure** aimed at assessing the game and controller designs by conducting a **play testing analysis**.

# Chapter 6

## Experimental Testing Procedure

In game design, the process of testing a game in order to identify design flaws, bugs and areas of improvement is known as **Play Testing**. This sections aims to meet TR7 by conducting a **Play Testing Analysis**.

- **TR7:** A play-testing analysis must be conducted.

To ensure the **acceptance test procedures** have been met, a play-test questionnaire is used to assess various aspects of the game. With minimal supervision, 20 individuals are tasked with filling in a pre-test questionnaire, which is aimed at identifying user's typical gaming habits, preferences, as well as their expectations of the game. The **Ethical Approval Form** used to grant permission for this survey can be found in Appendix A.4, along with the **Informed Consent** document used for the play test in fig. A.11 . All participants provided informed consent before play testing, and their data was handled in accordance with ethical guidelines of the University of Cape Town.

Each user is tasked with playing the game for a set period of time, after which a post-test questionnaire is completed. This questionnaire addresses key aspects of the game, and invites players to share their overall experience, as well as essential feedback necessary for further development. In Appendix fig. A.5, the pre- and post-testing questionnaires can be found.

One of the descriptions of the project was to "create a fun 2D game". This description poses some challenges as 'fun' can be subjective. To achieve a quantifiable outcome of the game's enjoyment, the play testing questions are specifically designed to analyse this aspect. The metric deduced in this analysis is the average score from the 20 participants in the survey. While more advanced methods, such as **Biometric Feedback** and **Engagement Metrics** could provide deeper insights into player enjoyment, a more invasive data collection and far longer periods of analysis are required to achieve this.

The game used in this iteration of play testing can be found online using the **link** provided below:

[APOCO PLAY- TEST VERSION](#)

## 6.1 Play Testing Setup

Each individual is given 20 minutes to complete the play-testing procedure. A breakdown of the time allocated to each activity can be seen in tab 6.1.

Activity	Time
Pre play-test questionnaire	5 min
Play-test	10 min
Post play-test questionnaire	5 min

Table 6.1: Breakdown of Time

The full play test questionnaire can be found in Appendix A fig. A.5.

During the **play test**, each participant is required to complete the tutorial and the first level of the game. Thereafter, they are given the opportunity to explore the subsequent two levels. Minimal guidance is provided to assess the game's intuitive nature and assistance is only offered if a participant is completely unsure of how to proceed.

## 6.2 Play Testing Results

The play testing evaluated the performance of the 2D game. The aim of the experimental testing procedure was to determine whether the requirements in Chapter 4 were met, as well as identify possible areas for improvement. All participants questionnaires can be accessed through the **link** provided below.

[APOCO PLAY- TEST QUESTIONNAIRE SUBMISSIONS](#)

## 6.3 Pre-Play Test

The pre-play testing questionnaire allows for insight into the participant pool, identifying age, gender, gaming habits and what participants most enjoy about 2D games. Tables 6.1 to 6.6 present the data obtained from the pre-play testing questionnaire.

Table 6.2: Age Distribution

Age	18-24	25-34	45-54	55+
Participants	17	2	1	0

Table 6.3: Gender Distribution

Gender	Male	Female
Participants	12	8

Table 6.4: Gaming Frequency

Gaming Frequency	Daily	A few times a week	Once a week	A few times a month	Rarely
Participants	3	3	3	4	7

Table 6.5: Genre of Games

Genre	Action	Adventure	Puzzle	Strategy	Role-Playing	Simulation	Sports	Other
Participants	8	5	5	8	4	5	7	2

Table 6.6: Familiarity with 2D Platformers

Familiarity	Very Familiar	Somewhat Familiar	Not Very Familiar	Not at all Familiar
Participants	4	7	4	5

Table 6.7: What Participants Enjoy about 2D Games

Enjoyment	Graphics and Art Style	Simplicity	Challenge and Difficulty	Story and Narrative	Music and Sound Effects
Participants	6	8	9	5	3

### 6.3. PRE-PLAY TEST

The data from these tables are illustrated in bar graphs 6.1 to 6.6, allowing for a visual representation of the pre-play testing data.

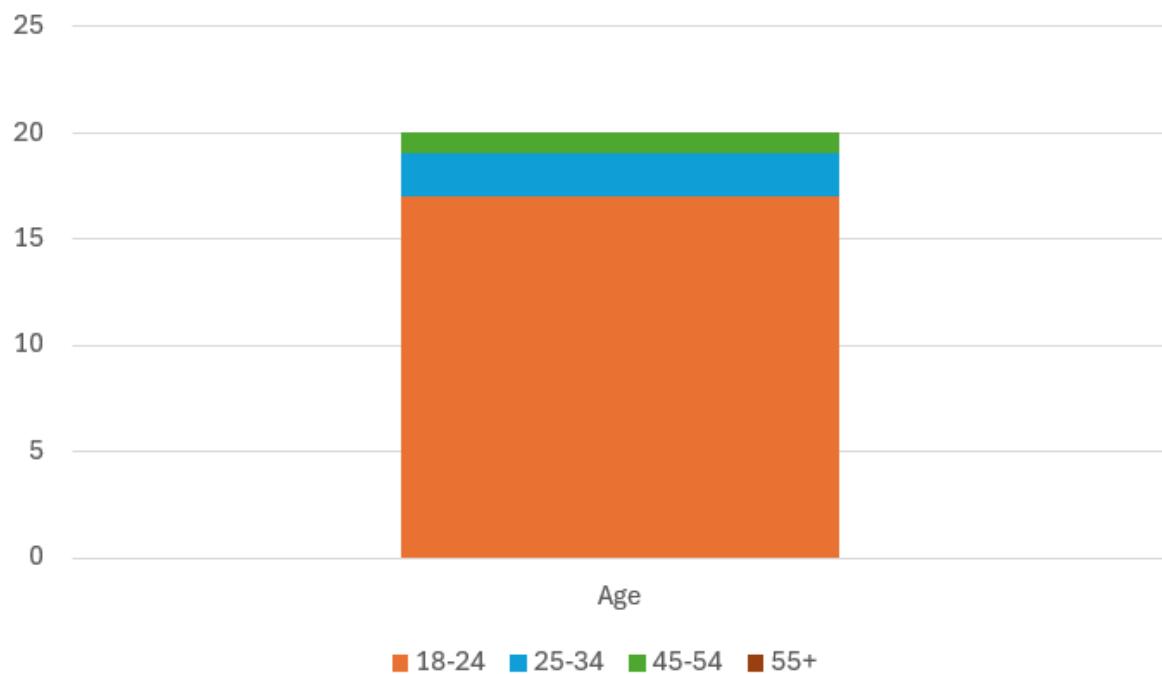


Figure 6.1: Age Distribution

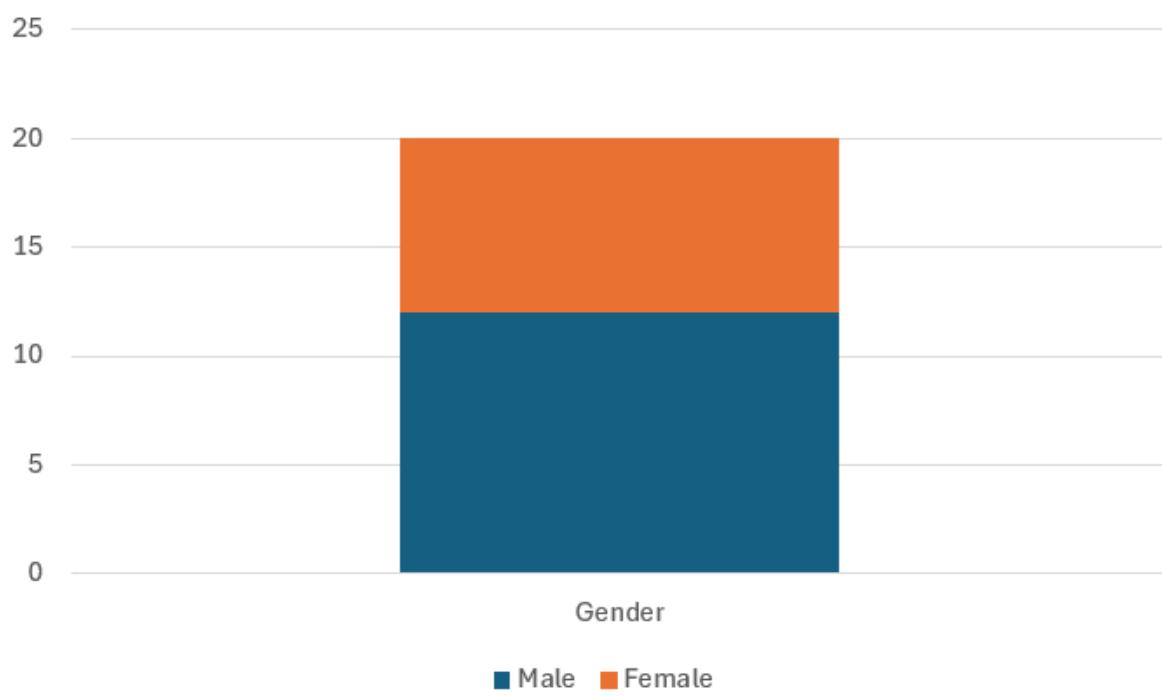


Figure 6.2: Gender Distribution

### 6.3. PRE-PLAY TEST

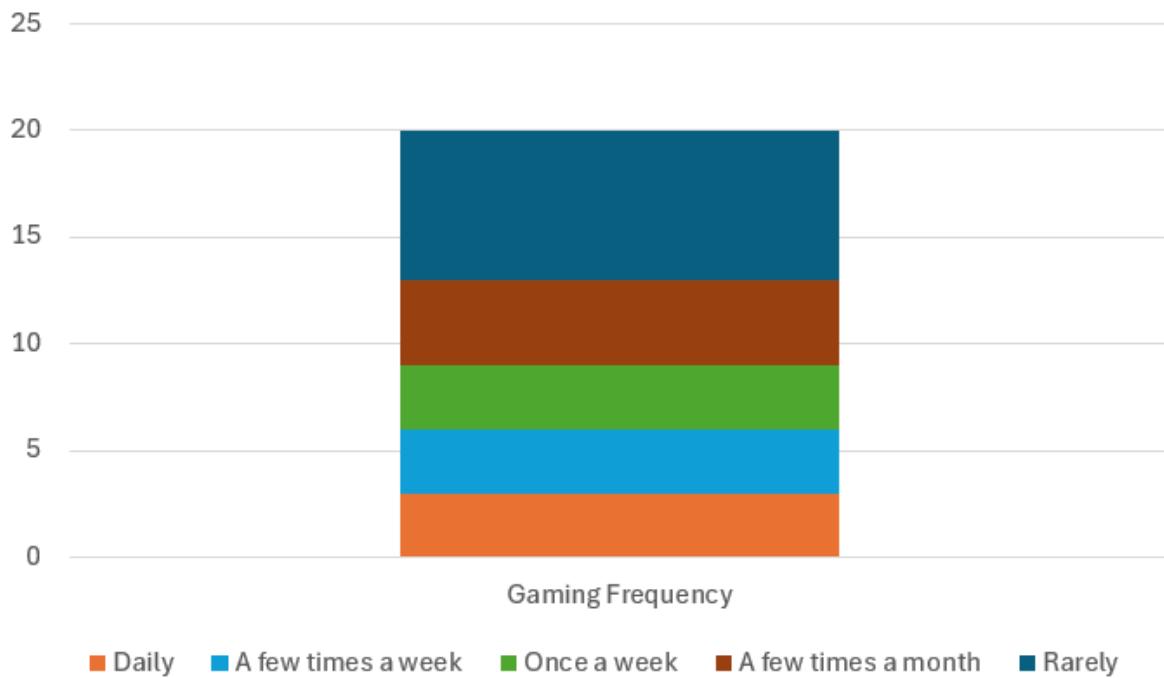


Figure 6.3: Gaming Frequency

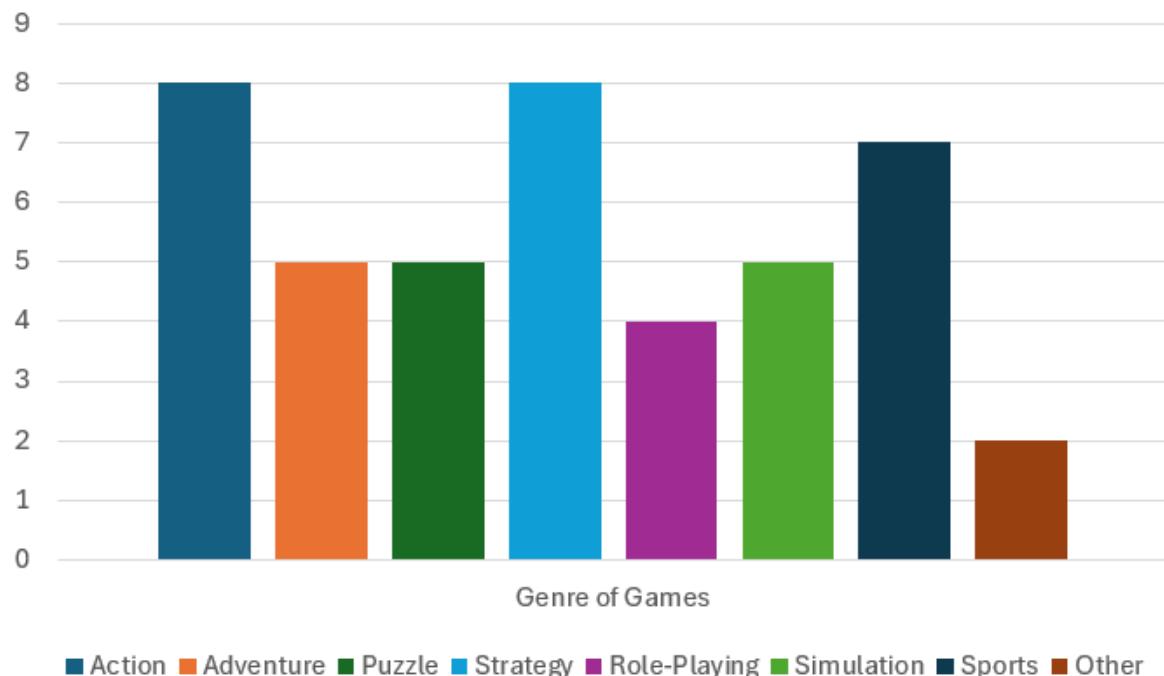


Figure 6.4: Genre of Games

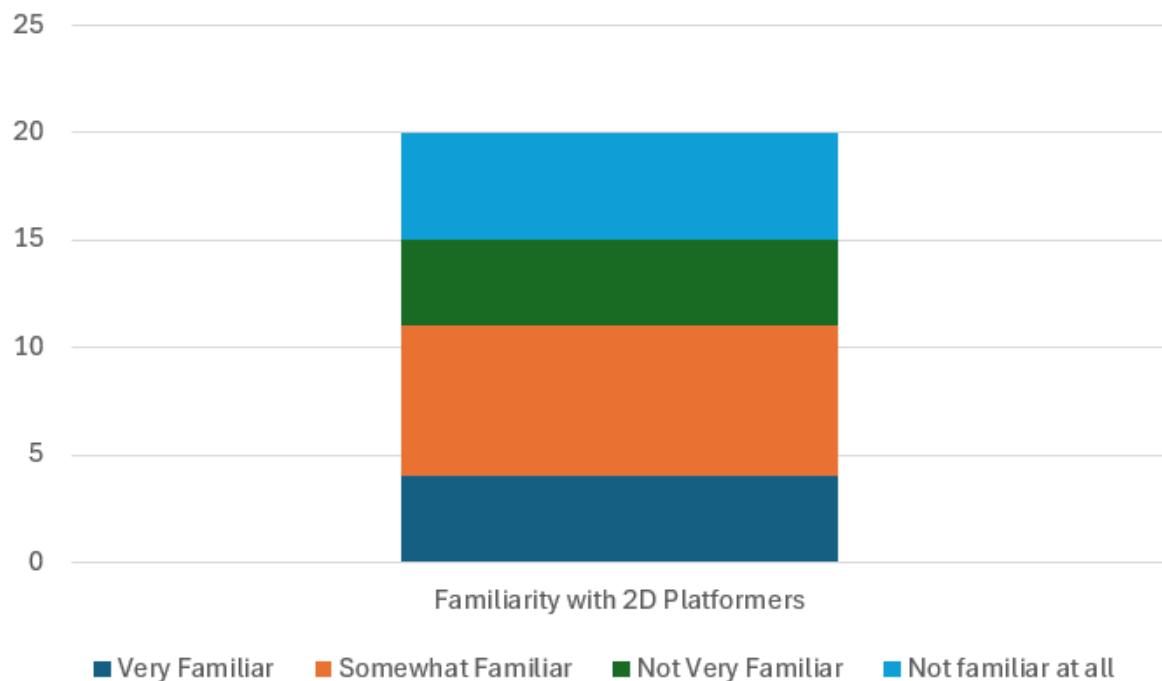


Figure 6.5: Familiarity with 2D Platformers

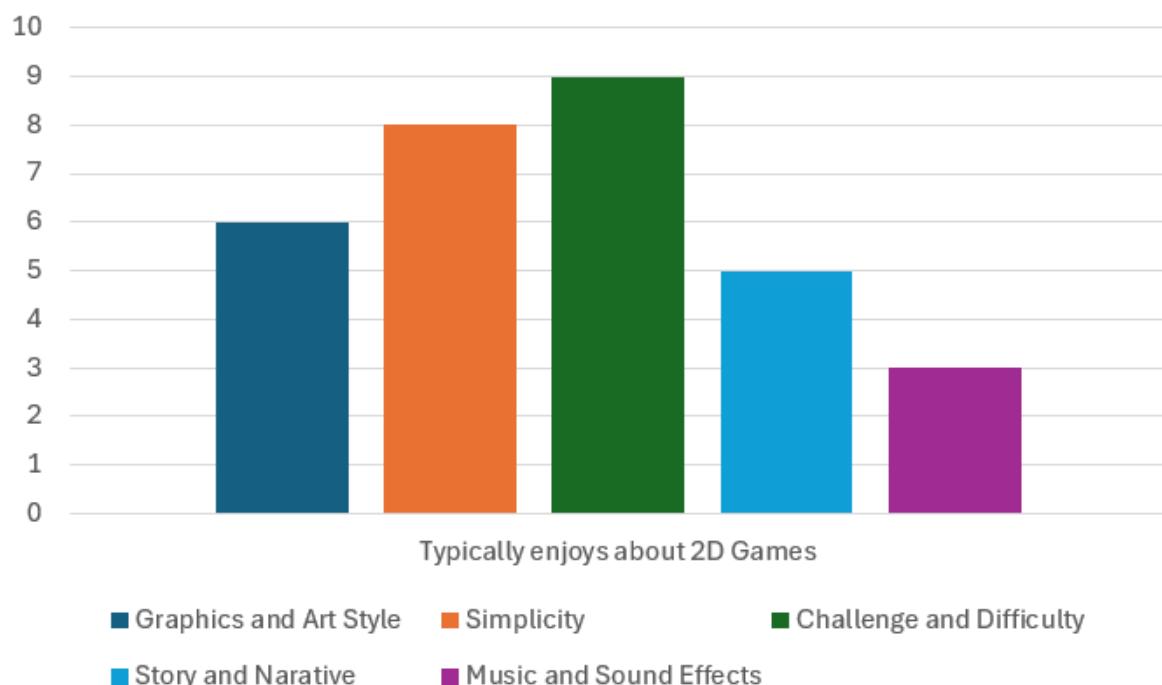


Figure 6.6: What Participants Enjoy about 2D Games

## 6.4 Post-Play Test

To obtain a quantifiable outcome of player enjoyment, the post-play testing questionnaire consists of rating-based questions, which allow for the scores to be averaged. Each of the values in tab 6.8 are based on a rating system of 1 to 5, where the average scores obtained from each question are presented. An example of the question layout is displayed in fig 6.7, illustrating how scores are obtained.

**Rate the following questions based on the scale provided**

Poor	Fair	Good	Excellent
<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4
<input type="checkbox"/> 5			

6. Graphics and Art Style:



Figure 6.7: Example of Question Layout presented in the Play Testing Questionnaire

The data obtained from the post-play testing questionnaire is displayed in tab 6.8 and visually represented in the bar graph shown in fig 6.8.

Table 6.8: Game Evaluation

Game Specifics	
Game Aspect	Average Score
Graphics and Art Style	4.4
Controls	4.6
The Tutorial Level	4.55
Graphical User Interface	4.45
Music and Sound Effects	4.5
Overall Game Evaluation	
Evaluation Topic	Average Score
Player Enjoyment	4.7
Sufficient Difficulty	4.4
Game Engagement	4.75
Addictiveness	4.1
Level Design	4.85

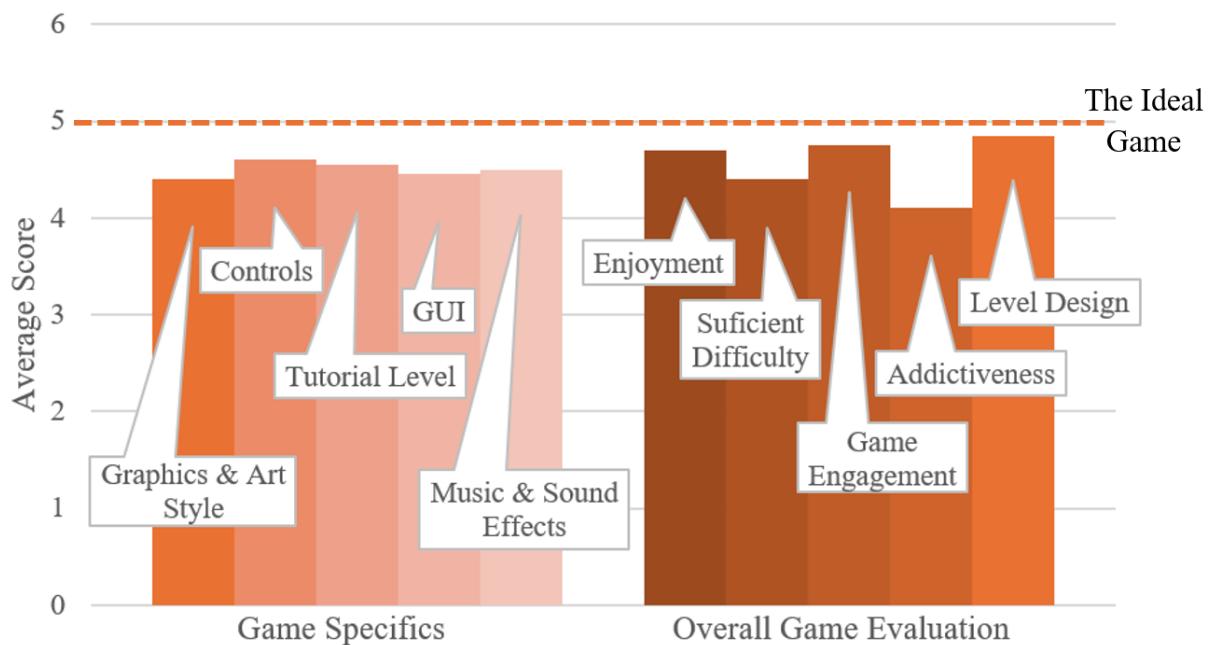


Figure 6.8: Game Evaluation

Of the 20 participants tasked with play testing the game, the majority did not encounter bugs during game play. The bugs encountered in the game are separated into two categories: **minor** issues and **fundamental run-time errors**. The results obtained from the post-play testing questionnaire are displayed in tab 6.9 and the bar graph 6.9.

Bugs	No Bugs	Minor Bugs	Fundamental run-time errors
Participants	11	8	1

Table 6.9: Summary of Encountered Bugs

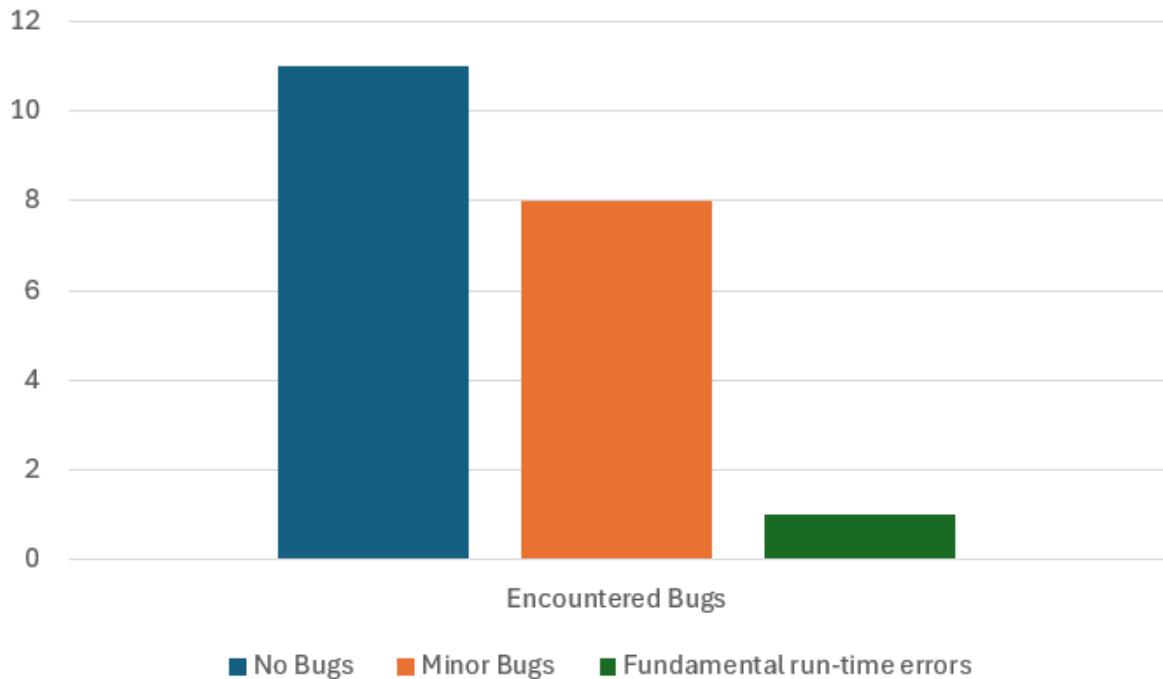


Figure 6.9: Participants that encountered bugs during gameplay

The **experimental test procedure** outlined above details the process involved in the collection of important data. Through the pre-play testing questionnaire, relevant information pertaining to the participant pool was recorded. Players were then tasked to play the game for a set duration. Consequently, each participant filled out a post-play testing questionnaire targeted specifically for analysing certain key aspects in the game's design. These results are discussed in the next chapter, showcasing how the data obtained is relevant to the project.

# Chapter 7

## Discussion

The discussion focuses on reviewing the data obtained from the **experimental testing procedure**. The responses from the pre-play testing questionnaire give insight into the demographic, gaming habits and preferences of the participant pool. By capturing the participants' responses in the post-play test, the key elements of the game can be assessed. This discussion will interpret these findings and highlight their significance to the project's goals.

### 7.1 Data Analysis and Verification

This section focuses on analysing and verifying the data obtained from the **experimental testing procedure**. By examining the response from the pre- and post-play testing questionnaires, the **acceptance test procedures** seen in tab 7.1 can be verified.

Table 7.1: Acceptance Test Procedures

Test	Explanation
ATP1	The game is developed using GameMaker Studio 2.
ATP2	Successful implementation of procedural generation.
ATP3	The difficulty rating of the game is sufficient.
ATP4	Players find the game enjoyable
ATP5	Players consider the game to be aesthetically pleasing with a coherent art style.
ATP6	Code runs smoothly without bugs or lag.
ATP7	The game controller has the functionality to play the game.
ATP8	There is a large enough sample size such that adequate play-testing is conducted.

## 7.1. DATA ANALYSIS AND VERIFICATION

**ATP1** is verified as the game is successfully developed with **GameMaker Studio 2**.

Through table 6.8 and the bar graph 6.8, it can be seen that **procedural generation** implementation is successful as participants felt that each level is carefully designed when in fact, the levels are procedurally generated through pseudo random functions and rules. This satisfies **ATP2** of successful **procedural generation implementation**.

The **difficulty rating** of the game can be seen in table 6.8 and the bar graph 6.8. On average, participants felt that the game's difficulty was sufficient for their level, with a few outliers. Player's that played game's daily, felt that the game was too easy for them. In contrast, player's that rarely played games, struggled with levels two and three, unable to complete the levels in the time frame given. While this data contains outliers, the average score obtained from the play-test verifies **ATP3**, confirming a **sufficient difficulty rating** for the game.

All survey participants found the game to be **enjoyable** to some extent. Most participants rated the game highly, as shown in table 6.8 and Bar Graph 6.8, confirming that the game is **subjectively enjoyable** and verifying **ATP4**.

The **aesthetic** quality of the game is mentioned in the post-play-testing questionnaire. Participants were asked questions relating to the Graphics, Art Style, Graphical User Interface, and Audio Sound Effects. Most of the participants rated the Graphics, Art Style and GUI highly, as seen in table 6.8 and Bar Graph 6.8, demonstrating the game's visual appeal. Similarly, the audio aesthetic reflected high average scores. Overall, the data verifies **ATP5**, as the game is considered to be **aesthetically pleasing** with a **coherent art style**.

Participants were asked whether they experienced **issues or bugs** during the play-test. Of the 20 participants, 9 found there to be issues as seen in tab 6.9. Unfortunately, 1 of these participants found a fundamental run-time error, causing the game to crash. While this definitely highlights areas for improvement, it verifies that **ATP6** has been met to a certain extent, as the majority of participants found the game to be free of **issues**. In game design, it is almost impossible for a game to be completely free of bugs. While it is theoretically possible, game design is almost never a perfect process. Through many play-tests it is possible to reduce the number of bugs and issues within the game.

The **game controller** proved to function sufficiently, affirming **ATP7**. Only 5 of the 20 participants were instructed to use the gaming controller due to it being a prototype. The controller prototype is not designed for hours of game-play and for this reason, was

not exposed to the entire participant pool.

By analysing the data from 20 different participants, sufficient information is obtained from the play-test. While this number could be larger, for the purposes of the project the sample size is **adequate**. While in Large AAA Games, the number of professional play-testers can be in the hundreds, independent game developers do not have access to these resources. Structures are provided through online game hosting sites where people can play the game in the public domain and provide feedback. Through participant pool mentioned in chapter 6, **ATP8** is verified providing the project with **adequate play-testing** data.

## 7.2 SWOT Analysis

A SWOT analysis used to identify and evaluate the project's Strengths, Weaknesses, Opportunities and Threats, paving the way for future work around the topic. Through the data collected in the **pre-** and **post-**play testing questionnaires, the SWOT analysis can be conducted.

Table 7.2: SWOT Analysis

SWOT Analysis	
Strengths	Weaknesses
<ul style="list-style-type: none"> <li>- Player Enjoyment</li> <li>- Game Engagement</li> <li>- Level Design</li> <li>- Controls</li> <li>- Graphical User Interface</li> <li>- Music and Sound Effects</li> </ul>	<ul style="list-style-type: none"> <li>- Sufficient Difficulty</li> <li>- Addictiveness</li> <li>- Bugs</li> </ul>
Opportunities	Threats
<ul style="list-style-type: none"> <li>- Large Target Market</li> <li>- Project can be further developed</li> <li>- Multi-platform launch</li> </ul>	<ul style="list-style-type: none"> <li>- Evolving Market</li> <li>- Lack of Advertisement</li> </ul>

### 7.2.1 Strengths

As illustrated in fig 6.8, the game posses a variety of **strengths**. From the overall design of the game, down to the minor details of music, sound effects, and graphics, these strengths contribute significantly to the overall gaming experience.

### 7.2.2 Weaknesses

The **difficulty** and **addictiveness** had the lowest performance rating relative to the other key game elements that were analysed. Through iteration of the design process, further simulations and thorough play testing, these weaknesses and bugs can be eliminated, improving the overall quality of the game.

### 7.2.3 Opportunities

From the pre-play testing questionnaire, insight into participant's gaming habits, preferences and frequency can be seen in figures 6.1 to 6.4. Through this data as well as the research conducted, it is evident that a large target market exists for this style of game. With the concept of procedural generation successfully implemented, the project has significant potential for further development. Using this project as a foundation, as well as a team of developers, this game could be elevated to an industry competing level. Multiple platforms can be targeted, such as PlayStation, Xbox, and potentially mobile, allowing for a broader audience reach and maximizing the game's market potential.

### 7.2.4 Threats

The game industry, as well as technology is constantly evolving. This brings to light a significant **threat** of the game becoming outdated. To keep up with this evolution, software developers must constantly match the rapid pace at which these technologies and competing games industries are changing. Consequently, the need for advertisement arises. The **threat** associated with this (especially for Indie Game Developers) is the lack of funding and resources available to compete with the AAA Game industry. Although this presents significant challenges, marketing paths through social media and game hosting websites provide Indie developers an opportunity to display their creations.

By analysing the results obtained from the **Experimental Testing Procedure**, each of the **Acceptance Test Procedures** were successfully verified. Through conducting a SWOT analysis, the **Strengths, Weaknesses, Opportunities and Threats** of the project were discussed. Drawing data from the pre-play testing questionnaire allowed insight into the strengths and opportunities of the game, paving the way for future work around the topic. The conclusion of the project is discussed in the next chapter, highlighting each milestone, success and concern discussed throughout the report.

# Chapter 8

## Conclusion and Recommendations

By reviewing the relevant **literature** required to develop a functional 2D game, meaningful play, game mechanics, dynamics and aesthetics were highlighted as the critical focus points. The literature also discussed the importance of prototyping, emphasising the advantages different forms of prototyping could offer to the project. The **technical review** explored the two game engine's, Unity and GameMaker, identifying their respective strengths and weakness in game development. Through chapter 4, the project's **objectives** and **methodology** were addressed, breaking down the user requirements, creating acceptance test procedures and walking through the methodology used in the project. The **design** and **simulation** of the game was then detailed in chapter 5, outlining how each user requirement was achieved in every phase of the project. The game design section discussed the process involved in the creation of the game, focusing on the core mechanics of movement, procedural generation and enemy logic. The game dynamics and aesthetics were then considered, detailing how meaningful play is brought about, as well the critical role aesthetics play in system performance, while still maintaining player experience. The controller was also addressed in chapter 5, exploring component selection, the choice of microcontroller, modelling, prototyping, as well as the final design used for the project. In the **experimental testing procedure** the acceptance test procedures were verified through a participant based play test and questionnaire, using the average score obtained from each question as a quantifiable metric. The results acquired from this experimental testing procedure were then analysed in the **Discussion**, showcasing the projects successful performance in meeting the expected deliverables.

This project aimed to develop a 2D game with procedurally generated terrain and engaging gameplay mechanics. Using the development software GameMaker, a fully functional game was put forward, as well as a functional gaming controller that met the player's needs. In addition, the game was made with sufficient complexity by incorporating advanced game mechanics, dynamics and an aesthetically impressive display. Ample opportunity was provided for testing and evaluation of the game, that ensured the

acceptance tests were verified and thus, the expected deliverables were presented. The deliverables of the project were successfully met including a fully functional 2D game and detailed report that demonstrated a comprehensive understanding of each element in the **Game Design Process**.

Despite this project meeting all user requirements, some improvements can be made for future work around this topic. Many of the participants from the survey provided feedback regarding their personal **recommendations** for improving the game. These recommendations were diverse, with many participants displaying conflicted views on how to improve the game. Some participants suggested that more action sections and a faster pace would make the game more exciting, while others thought that the difficulty was sufficient and the calming pace added to the style of the game. The most common recommendation was the inclusion of more levels, bosses, power ups and weapons in the game, creating a more complex, longer and enjoyable gaming experience. Some participants felt that the game was lacking a storyline, which could make it stand out in comparison to similar games in circulation. With the tutorial level aimed at taking a user with no experience to a capable level of playing the game, certain participants felt that the tutorial could have been more explanatory with regards to how to maneuver cave walls, the function of the health bar and how to use different weapons. Taking all of these recommendations into account will allow this project to be further developed in the hope of **successfully launching** to game hosting sites, mobile Android devices and the App Store.

# Bibliography

- [1] Maira B Carvalho, Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, Carolina Islas Sedano, Jannicke Baalsrud Hauge, Jun Hu, and Matthias Rauterberg. An activity theory-based model for serious games analysis and conceptual design. *Computers & education*, 87:166–181, 2015.
- [2] Carlo Fabricatore. Gameplay and game mechanics: a key to quality in videogames. 2007.
- [3] George Kelly and Hugh McCabe. A survey of procedural techniques for city generation. *The ITB Journal*, 7(2):5, 2006.
- [4] Bradley Camburn, Vimal Viswanathan, Julie Linsey, David Anderson, Daniel Jensen, Richard Crawford, Kevin Otto, and Kristin Wood. Design prototyping methods: state of the art in strategies, techniques, and guidelines. *Design Science*, 3:e13, 2017.
- [5] Camilla Arndt Hansen and Michael Deininger. Novice designers’use of partitioning strategies to navigate the prototyping process. *Proceedings of the Design Society*, 1:2267–2276, 2021.
- [6] Ernest Adams and Joris Dormans. *Game mechanics: advanced game design*. New Riders, 2012.
- [7] Katie Salen and Eric Zimmerman. Game design and meaningful play. *Handbook of computer game studies*, 59:79, 2005.
- [8] Yrjo Engeström. Learning by expanding: An activity-theoretical approach to developmental research. *Orienta-Konsultit Oy*, 1987.
- [9] Alan Amory. Game object model version ii: a theoretical framework for educational game development. *Educational Technology Research and Development*, 55:51–77, 2007.
- [10] Sylvester Arnab, Theodore Lim, Maira B Carvalho, Francesco Bellotti, Sara De Freitas, Sandy Louchart, Neil Suttie, Riccardo Berta, and Alessandro De Gloria. Mapping learning and game mechanics for serious games analysis. *British Journal of Educational Technology*, 46(2):391–411, 2015.

- [11] Bohyun Kim. Game mechanics, dynamics, and aesthetics. *Library technology reports*, 51(2):17–19, 2015.
- [12] Robin Hunicke, Marc LeBlanc, Robert Zubek, et al. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, page 1722. San Jose, CA, 2004.
- [13] Andrzej Marczewski. *Gamification: a simple introduction*. Andrzej Marczewski, 2013.
- [14] Gonzalo Frasca. Ludologists love stories, too: notes from a debate that never took place. In *DiGRA conference*, volume 8, 2003.
- [15] Christos Papadimitriou and Georgios Piliouras. Game dynamics as the meaning of a game. *ACM SIGecom Exchanges*, 16(2):53–63, 2019.
- [16] William H Sandholm. *Population games and evolutionary dynamics*. MIT press, 2010.
- [17] Simon Niedenthal. What we talk about when we talk about game aesthetics. In *Proceedings of DiGRA 2009 Conference: Breaking New Ground: Innovation in Games, Play, Practice and Theory*, 2009.
- [18] Hanna Sommerseth. Exploring game aesthetics. In *Proceedings of DiGRA 2009 Conference: Breaking New Ground: Innovation in Games, Play, Practice and Theory*, 2009.
- [19] Bernard J Jansen. The graphical user interface. *ACM SIGCHI Bulletin*, 30(2):22–26, 1998.
- [20] SNB Baharom. *Player-centric Emotional Design for Digital Games: An Empirical Exploration in Visual Aesthetics*. PhD thesis, Doctoral dissertation, Sultan IDRIS Education University, 2017.
- [21] Alfredo M Ronchi. *eCulture: cultural content in the digital age*. Springer Science & Business Media, 2009.
- [22] Joatan Preis Dutra. Hci/gui design applied for adaptable context-aware mobile game. In *Clap Conference, Bremen, Germany*. Retrieved from [http://mutacao.com.br/joatan/docs/20120211-DUTRA\\_Joatan\\_Preis-HCI\\_GUI-Design\\_for\\_Mobile-CLAP\\_Paper.pdf](http://mutacao.com.br/joatan/docs/20120211-DUTRA_Joatan_Preis-HCI_GUI-Design_for_Mobile-CLAP_Paper.pdf), 2012.
- [23] Fei Hu and Lixia Ji. Gui design: From information architecture to interactive experience. In *2008 9th International Conference on Computer-Aided Industrial Design and Conceptual Design*, pages 913–917. IEEE, 2008.

- [24] Daniel Ashlock, Colin Lee, and Cameron McGuinness. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273, 2011.
- [25] Roland Van Der Linden, Ricardo Lopes, and Rafael Bidarra. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):78–89, 2013.
- [26] Julian Togelius, Emil Kastbjerg, David Schedl, and Georgios N Yannakakis. What is procedural content generation? mario on the borderline. In *Proceedings of the 2nd international workshop on procedural content generation in games*, pages 1–6, 2011.
- [27] Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.
- [28] Noam Nisan. Pseudorandom generators for space-bounded computations. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 204–212, 1990.
- [29] Michel Beaudouin-Lafon and Wendy E Mackay. Prototyping tools and techniques. In *The human-computer interaction handbook*, pages 1043–1066. CRC Press, 2007.
- [30] Jooyoung Jang and Christian D Schunn. Physical design tools support and hinder innovative engineering design. 2012.
- [31] Catherine Elsen, Anders Häggman, Tomonori Honda, and Maria C Yang. Representation in early stage design: an analysis of the influence of sketching and prototyping in design projects. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 45066, pages 737–747. American Society of Mechanical Engineers, 2012.
- [32] W Lawrence Neeley Jr, Kirsten Lim, April Zhu, and Maria C Yang. Building fast to think faster: exploiting rapid prototyping to accelerate ideation during early stage design. In *international design engineering technical conferences and computers and information in engineering conference*, volume 55928, page V005T06A022. American Society of Mechanical Engineers, 2013.
- [33] Michael Schrage. Cultures of prototyping. *Bringing design to software*, 4(1):1–11, 1996.
- [34] Jessica Menold, Kathryn Jablokow, and Timothy Simpson. Prototype for x (pxf): A holistic framework for structuring prototyping methods to support engineering design. *Design Studies*, 50:70–112, 2017.

- [35] Steven C Zemke. Student learning in multiple prototype cycles. In *2012 ASEE Annual Conference & Exposition*, pages 25–1185, 2012.
- [36] Chaitya Vohera, Heet Chheda, Dhruveel Chouhan, Ayush Desai, and Vijal Jain. Game engine architecture and comparative study of different game engines. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2021.
- [37] Maxwell Foxman. United we stand: Platforms, tools and innovation with the unity game engine. *Social Media+ Society*, 5(4):2056305119880177, 2019.
- [38] Afzal Hussain, Haad Shakeel, Faizan Hussain, Nasir Uddin, and Turab Latif Ghouri. Unity game development engine: A technical survey. *Univ. Sindh J. Inf. Commun. Technol.*, 4(2):73–81, 2020.
- [39] Jeff Craighead, Jennifer Burke, and Robin Murphy. Using the unity game engine to develop sarge: a case study. In *Proceedings of the 2008 Simulation Workshop at the International Conference on Intelligent Robots and Systems (IROS 2008)*, volume 4552, 2008.
- [40] António Andrade. Game engines: a survey. *EAI Endorsed Transactions on Serious Games*, 2(6), 2015.
- [41] Oona Pönni. Starting a beginner game project with gamemaker studio 2. 2021.
- [42] Eetu Aaltonen. Implementing multiplayer core features for gamemaker studio 2 using node. js: game area instances and simulation layer. 2024.

# Appendix A

## Additional Files and Schematics

### A.1 Graduate Attribute Requirements

#### A.1.1 GA 1: Problem Solving

I have demonstrated my problem-solving ability in various technical aspects of the game. Procedural generation is one of my major engineering problems, discussed in section 5.1.2. I have solved this problem and can successfully procedurally generate terrain including, blocks, water, traps, enemies and other obstacles in such a way that the player can navigate through the terrain effectively. This was achieved through many mathematical operations that randomly assign objects to positions on the screen based on a set of rules. Another major challenge is the loading time of levels, discussed in section 5.1.4. I have solved this problem by rearranging functions, reducing image sizes, eliminating ‘for loops’, as well as removing unnecessary objects to improve game speed. There have been many other smaller problems encountered during the game design and implementation process, including bugs (solved by stepping through and compartmentalizing code sections), runtime errors, and plot flow issues. The design of a functional controller was another engineering problem that was solved in section 5.2. Through using my engineering degree and thorough research in the topic, I have been able to solve these problems.

Problem solving is primarily covered in chapter 5 (**Design and Simulation**).

### A.1.2 GA 4: Investigations, Experiments and Data Analysis

Through research into the theory and formulae surrounding procedural generation, I have successfully solved this engineering problem. Using other people's research, I have gathered enough information to write a function to procedurally generate terrain within the game. I have conducted a game testing experiment using other people in my research, discussed in chapter 6. I have been granted ethical clearance and have used willing participants to play-test my software. This has highlighted fundamental bugs, errors and design flaws within the software. I have conducted speed tests during the loading phase and run-time of the game and have implemented changes to the software for optimisation. Using research methods obtained from experts within the field and the information gathered from this experiment, I have analysed the data and discussed the results obtained from the play test, discussed in chapter 7.

Experiments and data analysis is primarily covered in chapter 6 (**Experimental Testing Procedure**), while Investigations are conducted in chapter 2 (**Literature Review**) and chapter 5 (**Design and Simulation**).

### A.1.3 GA 5: Use of Engineering Tools

The following engineering tools have been used:

- GameMaker (A 2D game development platform). This is a requirement for my research project.
- OnShape (A 3D model designer tool).
- Kicad
- Prusa Slicer
- 3D Prusa Mini Printer.
- Arduino IDE

Using these engineering tools, I have been able to develop the necessary software, models and prototypes to complete my research project.

#### A.1.4 GA 6: Professional and Technical Communication

A professional and technical report has been detailed, discussing 2D Game design using GameMaker. Overleaf has been used to write the report, and Microsoft Teams has been used for communication, both considered professional communication tools.

#### A.1.5 GA 8: Individual Work

I have done almost all work involved in this project on my own and have demonstrated my ability to complete a 2D Game without the assistance of a game development team. I have learnt how to use GameMaker with **no prior knowledge** of this game development engine. I have also obtained ethical clearance, consequently conducting an investigation using other people for my research. I have researched the topic by myself and have learnt sufficient information to successfully complete this project with the guidance of Dr Schonken.

#### A.1.6 GA 9: Independent Learning Ability

I have watched hours of tutorial, programming and game development videos as well as read numerous articles and book chapters related to game design. I have also learnt how to use OnShape (a 3D modelling software) to create a controller for my game. I have learnt how to use GameMaker, as well as a new programming language (GML) with **no prior knowledge of the game development engine or language** as it was not covered in my Engineering Degree. Using this knowledge I have obtained sufficient information to be able to create a 2D Game by myself. While I have created this 2D game, there are limitations to the complexity that can be achieved as this is typically done by a team of developers.

## A.1. GRADUATE ATTRIBUTE REQUIREMENTS

## A.2 Project Description

<b>Student Proposed?</b> <i>If Yes, add student name</i>	No
<b>ID:</b>	FS24-03
<b>SUPERVISOR:</b>	WPF Schonken
<b>TITLE:</b>	2D Game Design using GameMaker™
<b>DESCRIPTION:</b>	<p>Recent years have seen corporate greed actively destroy the AAA game industry, which has been a significant motivation for developers to move the Indie space. Independent games are having a spectacular year in 2024 (at least so far), with some very good game development tools.</p> <p>This project will use one of these tools, specifically GameMaker™ (<a href="#">Make 2D Games With GameMaker   Free Video Game Maker</a>), to create a fun 2D game. Game development concepts like core gameplay loop, player choice in the form of divergences, juxtaposition gratification, etc. must be addressed. The game must be complex enough to satisfy GA 1 and GA 4, which will make this a challenging project.</p> <p>For more information on the technical side of game design, have a look at the Design Delve videos on Second Wind's YouTube channel: (<a href="#">674) Design Delve - YouTube</a></p>
<b>DELIVERABLES:</b>	<ul style="list-style-type: none"> <li>• Functional 2D game</li> <li>• Detailed Report</li> </ul>
<b>SKILLS/REQUIREMENTS:</b>	<ul style="list-style-type: none"> <li>• Strong programming skills</li> <li>• An interest in video games and game development</li> </ul>
<b>GA 1: Problem solving</b> <i>Identify, formulate, analyse and solve complex* engineering problems creatively and innovatively</i>	With sufficient game complexity there should be enough problem solving required.
<b>GA 4: Investigations and analysis</b> <i>Demonstrate competence to design and conduct investigations and experiments.</i>	Ample opportunity for testing and evaluation.

Figure A.1: Project Description Page 1

## A.2. PROJECT DESCRIPTION

GA 5: Use of engineering tools: Demonstrate competence to create, select and apply and recognise limitations of appropriate techniques, resources and modern engineering and IT tools, including prediction and modelling, to complex engineering problems	GameMaker™
EXTRA INFORMATION	<b>Warning:</b> This is a <b>dangerous, open-ended</b> project and should only be considered by students with strong programming skills, a serious interest in gaming/game development, and an artistic streak. If you do not meet these requirements at the start of 2 <sup>nd</sup> semester, this topic will be scrapped and a new random topic will be assigned to you.
BROAD Research Area	Game development, programming
Suitable for ME/ECE/EE/ALL?	ECE

### Risks clearance questionnaire

		Yes	No
<b>Q1</b>	Does this project involve data collection	x	
<b>Q2</b>	Does this project involve utilizing a third-party data set		x
<b>Q3</b>	Does this project utilize machine learning (ML) or artificial intelligence (AI)?	x	
<b>Q4</b>	Does it exceed the minimum risk defined here: <a href="#">Link</a> [Answer is No here if your project does not utilize ML and AI]	?	?
<b>Q5</b>	Does this project involve external parties, funders, etc		x

Answer the following questions if you answer "Yes" to any of the above questions.

If the answer is "Yes" to **Q1**, please answer the following questions:

		Yes	No
<b>Q6</b>	Are there humans or animals directly involved in the data collection process or contains any identification information		x

If the answer is "Yes" to **Q2**, please answer the following questions:

		Yes	No
<b>Q7</b>	Are the third-party data used anonymous (data does not contain human or animal-related information?)		x

Figure A.2: Project Description Page 2

## A.2. PROJECT DESCRIPTION

<b>Q8</b>	Are the third-party data used from an open source?		x
<b>Q9</b>	Are the third-party data used from a different research group?		x
<b>Q10</b>	If the answer to <b>Q9</b> is "Yes", do you have the approval to use third-party data sets? Attach the proof to PSQ application.		

If the answer is "Yes" to Q5, please answer the following questions:

		Yes	No
<b>Q11</b>	Have you signed an MOU between the parties [If Yes, attach the proof to PSQ application.]		
<b>Q12</b>	Will there be a chance for any conflict of interest between the parties? [If Yes, provide details of the issue and your plan to solve it]		

Figure A.3: Project Description Page 3

## A.3 Ethics Approval



2024/08/28

EBE/01157/2024

RE: Research Ethics Committee Project Approval Letter

Dear Liam Breytenbach,

Your application for ethics review of your project titled  
2D Game Design using GameMakerTM

has been reviewed and evaluated by the  
Engineering & Built Environment Committee.

You may proceed with your research project titled:  
2D Game Design using GameMakerTM

Expiration date of approval: 2024/12/31

Please note that should:

- (i) any serious or adverse effects to participants occur and/or,
- (ii) aspect(s) of your current project change and/or
- (iii) any unforeseen events that might affect continued ethical acceptability of the project occur then you should immediately report this to the approving REC. You may be required to submit an amendment to this application, in order to determine whether the changed aspects increase the ethical risks of your project.

Based on the information supplied your application has been successful and is approved.

Please note the following additional conditions associated with this approval:

- (i) If UCT students are to be used for this research, permission needs to be applied for from the Dept. of Student Affairs.

Regards,

Engineering & Built Environment Committee.

Figure A.4: Ethics Approval

## A.4 Play Testing Questionnaire

### 2D Game Design using GameMaker

#### Playtesting Questionnaire

---

### **Pre-Playtest Questions**

#### **Participant Information**

*(Please note that this section is optional, and your identity will remain anonymous. No personal data will be shared or published.)*

**Participant Number:**.....

Please circle your answer:

**1. Age:**

- 18-24
- 25-34
- 35-44
- 45-54
- 55+

**2. Gender:**

- Male
- Female
- Other
- Prefer not to say

**3. How often do you play video games?**

- Daily
- A few times a week
- Once a week
- A few times a month
- Rarely

Figure A.5: Play-Testing Questionnaire P1

#### A.4. PLAY TESTING QUESTIONNAIRE

4. What types of games do you usually play? (Select all that apply)

- Action
- Adventure
- Puzzle
- Strategy
- Role-Playing (RPG)
- Simulation
- Sports
- Other: \_\_\_\_\_

5. How familiar are you with 2D platformer games?

- Very familiar
- Somewhat familiar
- Not very familiar
- Not familiar at all

6. What do you typically enjoy about 2D games? (Select all that apply)

- Graphics and art style
- Simplicity
- Challenge and difficulty
- Story and narrative
- Music and sound effects
- Other: \_\_\_\_\_

7. Considering this is an undergraduate level final year project for Mechatronics, what are your expectations before playing this game?  
*(Open-ended question)*

.....  
.....  
.....

---

Figure A.6: Play-Testing Questionnaire P2

## Post-Playtest Questions

### The Game

Rate the following questions based on the scale provided

No	Sort of	Yes
<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3
<input type="checkbox"/> 4	<input type="checkbox"/> 5	

1. Overall, was the game enjoyable?

2. Was the difficulty of the game sufficient for your level?

3. Was the game engaging?

4. Was the game addictive?

5. Did the levels in the game appear to carefully designed (excluding the Tutorial)?

Figure A.7: Play-Testing Questionnaire P3

#### A.4. PLAY TESTING QUESTIONNAIRE

Rate the following questions based on the scale provided

Poor	Fair	Good	Excellent
<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4
<input type="checkbox"/> 5			

6. Graphics and Art Style:

7. Controls:

8. The Tutorial Level:

9. Graphical User Interface:

(Main Menu, Level Selection, In-game Store, Pause Menu)

10. Music and Sound Effects:

Figure A.8: Play-Testing Questionnaire P4

#### A.4. PLAY TESTING QUESTIONNAIRE

**11. Did you encounter any issues or bugs during the game?**

*(If yes, please describe them in detail.)*

- Yes
- No

.....  
.....  
.....  
.....

**12. What did you like most about the game?**

*(Open-ended question)*

.....  
.....  
.....

**13. What did you like least about the game?**

*(Open-ended question)*

.....  
.....  
.....

**14. Do you have any suggestions for improving the game?**

*(Open-ended question)*

.....  
.....  
.....

**15. Would you recommend this game to others?**

- Definitely
- Probably
- Not sure
- Probably not
- Definitely not

**16. Any additional comments?**

*(Open-ended question)*

.....  
.....  
.....

Figure A.9: Play-Testing Questionnaire P5

### **The Controller**

**17. How intuitive was using the controller?**

- Very
- Some what
- Not at all

**18. Were you able to play the game with the controller?**

- Yes
- No

**19. Did you encounter any issues with the controller?**

*(If yes, please describe them in detail.)*

- Yes
- No

*(Description:)* .....

.....  
.....  
.....

### **The Game Cover (Game Packaging)**

**20. Did the game cover intrigue you?**

- Yes
- No

**21. Was the art style of the cover coherent with the game?**

- Yes
- No

**22. Was the User Manual sufficient for this game?**

- Yes
- No

---

APOCO PLAY- TEST VERSION

<https://gx.games/games/ublc23/apoco-playtest-version-/tracks/234c3db2-351c-4571-9b28-00ba1450d87a/>

Figure A.10: Play-Testing Questionnaire P6

## A.5 Informed Consent

### **INFORMED CONSENT DOCUMENT**

#### **INFORMATION SHEET & CONSENT FORM**

##### **2D GAME DESIGN USING GAMEMAKER**

Hello,

My name is Liam Breytenbach, and I am an undergraduate Mechatronics Engineering student at the University of Cape Town. I am researching 2D game design and would like to invite you to participate in the project.

This project focuses on creating a 2D game that is engaging, complex, and strategic. The goal of my research is to assess whether the game could potentially be launched on a game/app store. The purpose of this questionnaire is to get feedback from individuals who enjoy playing games to test the game, identify problems, and rate the overall experience.

Please understand that your participation is voluntary and the choice to participate is yours alone. If you choose to participate, there are no negative consequences. You are also allowed to withdraw at any time. However, I would be grateful if you would assist me by allowing me to interview you.

The process will take approximately 20 minutes – 5 minutes for the pre-game questionnaire, 10 minutes of gameplay, and 5 minutes for the post-game questionnaire. No travel arrangements are necessary and there are no costs involved.

Each participant will remain anonymous throughout the research. If you would like to be informed about the results, an email can be sent to you after the conclusion of the project.

No personal questions are asked in this survey, so each participant cannot be identified by the public. Thank you for your participation.

This data is only going to be used to improve my game.

#### **Consent:**

Name of participant.....Date .....

Signature.....

Figure A.11: Informed Consent

# Appendix B

## Project Folder and AI Use

### B.1 Project Folder

This folder contains all relevant files pertaining to the project.

APOCO

### B.2 AI Use

ChatGPT has been used as an assistive tool throughout the project. ChatGPT has **not** done this work for me. I have asked ChatGPT for advice with regards to report layout, paragraph construction and flow. ChatGPT's response have not been copied and pasted but **selectively** chosen and added to the report to aid in the quality of the work put forward. Furthermore, ChatGPT has been used to create tables, bullet points and other such displays in Latex due to it's effiency in doing so. During the 2D game development process, ChatGPT assisted me with commenting sections of my code, as well as working along side me in debugging certain aspects of the game. A few examples of how ChatGPT was used can be seen below.

This is better: The beauty of this function lies in its ability to accept new parameters, allowing for the terrain to keep generating past the ending point. By updating the start and ending x coordinate, this terrain can be generated indefinitely.



Here's a refined version:

---

The elegance of this function is in its flexibility to accept new parameters, enabling continuous terrain generation beyond the initial endpoint. By adjusting the starting and ending x coordinates, the terrain can be generated indefinitely.

Figure B.1: Asking AI to reword paragraphs

**MY COMBINED VERSION:** The beauty of this function lies in its ability to accept new parameters, allowing for the terrain to constantly generate past the initial end point. By updating the starting and ending x coordinate, this terrain can be generated indefinitely.

Write this better: Before any programming can be conducted, a basic idea needs to be established about what exactly is happening in the 2D game. \textbf{Paper prototyping} is quick and inexpensive method used to convey basic concepts and ideas as outlined by Adams \cite{adams2012game}. By using this method of prototyping, many rapid iterations can be produced. The final paper prototype after a few iterations is displayed in fig \ref{Final Paper Prototype}.



Before any programming can begin, it is essential to establish a clear understanding of the fundamental mechanics and flow of the 2D game. \textbf{Paper prototyping}, as highlighted by Adams \cite{adams2012game}, is a fast and cost-effective technique used to communicate core concepts and ideas. This approach allows for rapid iterations, enabling designers to refine their ideas quickly. The final version of the paper prototype, developed after several iterations, is shown in Figure \ref{Final Paper Prototype}.

◀ ▶ ⌂ ⌂ ~

Figure B.2: Asking AI to reword paragraphs

**MY COMBINED VERSION:** Before any programming can be conducted, a basic idea needs to be established of the fundamental mechanics and flow of the 2D game. **Paper prototyping** is a quick and inexpensive method used to convey basic concepts and ideas as highlighted by Adams [6]. By using this method of prototyping, many rapid iterations can be produced, quickly eliminating weaker design concepts and promoting certain design strengths. The final paper prototype after a few iterations is displayed in fig 5.1.

Make a latex table from this: Table 3.1: Technical Requirements  
 Technical Requirements  
 Explanation  
 TR1 Design of a electromagnetic system within the size of a real tadpole.  
 TR2 Tadpole capable of undulatory lateral oscillations in air, and in water.  
 TR3 Design driving circuit capable of powering the electromagnetic system.  
 TR4 Design of the soft body shell.  
 TR5 Multi-objective optimisation.  
 TR6 Thermal Analysis.

To create a LaTeX table from the provided information, you can use the `tabular` environment. Below is an example of how to structure your table, including the title and the technical requirements along with their explanations:

**LaTeX Code for the Table**

```
latex
\begin{table}[h]
\centering
\caption{Technical Requirements}
\label{tab:technical_requirements}
\begin{tabular}{|c|p{0.6\textwidth}|}
\hline
```

**Message ChatGPT**

Figure B.3: Asking AI to make tables

How do you make a word bold in Latex

To make a word bold in LaTeX, you can use the `\textbf{}` command. For example:

```
latex
\textbf{This is bold text.}
```

This will make the text inside the curly braces appear bold.

Figure B.4: Asking AI to make text bold

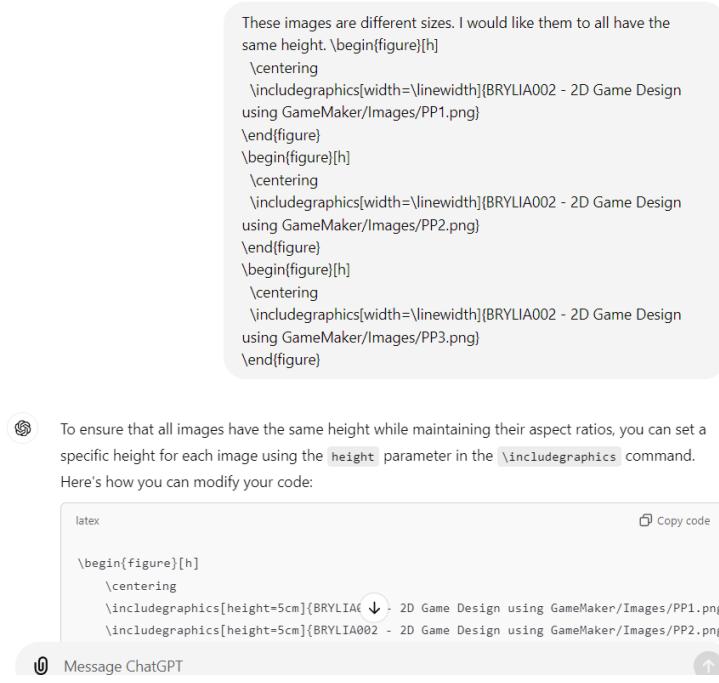


Figure B.5: Asking AI for assistance with Figure layout

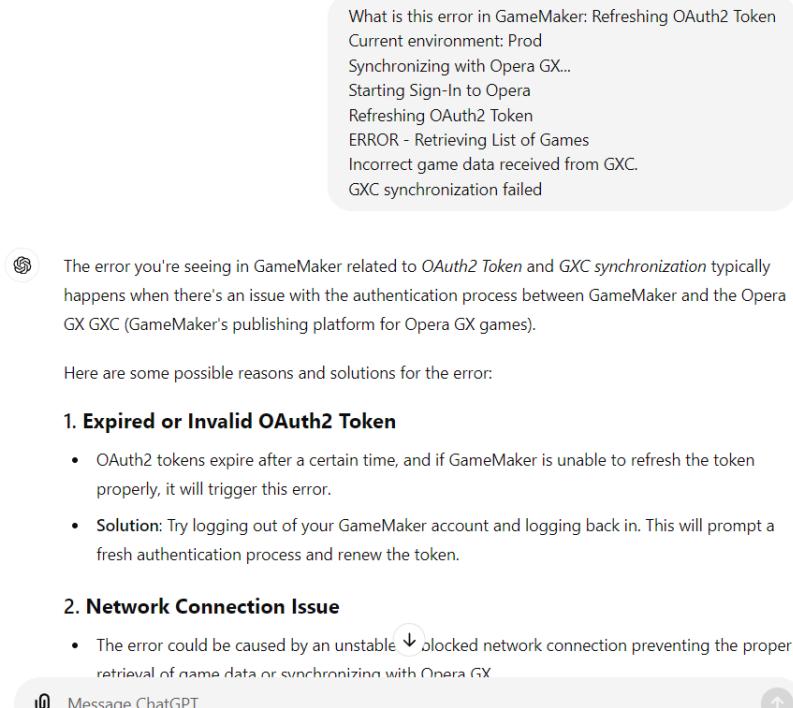


Figure B.6: Asking AI for assistance with GameMaker errors

The screenshot shows a GameMaker Studio error message in a light blue box:

```
ERROR in action number 1
of Step Event0 for object obj_baboon:
Variable <unknown_object>.spawner_timer(100017, -2147483648) not
set before reading it.
at gml_Object_obj_baboon_Step_0 (line 72) -      if (spawner_timer >
0) {
#####
##### gml_Object_obj_baboon_Step_0 (line 72)
```

Below the error message is a circular icon with a gear symbol. To its right is a paragraph of text:

The error you're seeing indicates that the variable `spawner_timer` is not initialized before you try to access it in the `Step` event. GameMaker requires variables to be initialized before use to avoid these kinds of errors.

To fix this, you need to initialize `spawner_timer` in the **Create Event** of `obj_baboon`. This ensures that the variable exists and has a valid value before the **Step Event** accesses it.

Figure B.7: Asking AI for assistance with GameMaker errors

I have used AI as an assistive tool. Through report layout, paragraph construction and flow, ChatGPT's responses have been **selectively** chosen and added to the report to aid in the quality of the final work put forward.