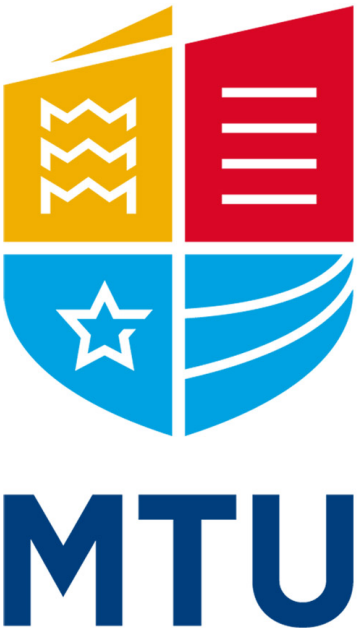

Video Game Immersion : The art of creating a virtual experience

by

Christopher O'Shea

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in Software Development

in the
Faculty of Engineering and Science
Department of Computer Science

May 2022

Declaration of Authorship

I, Christopher O'Shea , declare that this thesis titled, 'Video Game Immersion : The art of creating a virtual experience' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Munster Technological University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Munster Technological University

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Christopher O'Shea

Game development and design is very prevalent in the world that we live in today. Games display many unique features and traits in their content which uniquely identify them among others. This can range from simple mobile applications to worldwide known AAA or triple A (well marketed high quality) games such as Call of Duty.

In this Paper we will take a deep dive into what makes these games so immersive. This will include a number of steps including game genre's, game mechanics, art, music, story-lines and objectives. To achieve this we will first have to look at the games that came before us and the impact that these games had. Comparing and analysing older video games to more recent games will also be important to understand the success of games.

During the course of this work the main aim will be to discover how these games relieve the hardships of life and offer a product on a worldwide scale. The project itself will include the design and implementation of a game application. The solution includes developing a game which is easily accessible, reachable worldwide, improving individuals motor skills and creating a beneficial virtual experience. This will aim to benefit those of us who are socially awkward or may have conditions like agoraphobia.

Acknowledgements

I would like to thank my project supervisors 'Paul Davern' and 'Brian Murphy' for providing me with continuous support and assistance throughout the duration of this project. I would also like to thank both my parents "Virginia and Gerard O'Shea" for continuous proofreading of this paper. A special mention also to my dog 'Lola' for barking her support all throughout the duration of this project.

Contents

List of Figures

List of Tables

Abbreviations

HTML	**H**per**T**ext **M**arkup **L**anguage
CSS	**C**ascading **S**tyle **S**heets
IDE	**I**ntegrated **D**evelopment **E**nvironment
MDA	**M**echanics **D**ynamics **A**esthetics
iMUSE	**I**interactive **MU**sic **S**treaming **E**ngine
VR	**V**irtual **R**eality
PCG	**P**rocedural **C**ontent **G**eneration
GVG-LG	**G**eneral **V**ideo **G**ame **L**evel **G**eneration
GVG-AI	**G**eneral **V**ideo **G**ame **A**rtificial **I**ntelligence
SB-LG	**S**earch **B**ased **L**evel **G**enerators
PDP-1	**P**rogrammed **D**ata **P**rocessor
GUI	**G**eneral **U**ser **I**nterface
DDA	**D**ynamic **D**ifficulty **A**djustment
OS	**O**perating **S**ystem
NES	**N**intendo **E**ntertainment **S**ystem
UWP	**U**niversal **W**indows **P**latform
IOS	**I**Phone **O**perating **S**ystem
GML	**G**ame **M**aker **L**anguage
API	**A**pplication **P**rogramming **I**nterface
FPS	**F**rames **P**er **S**econd

For/Dedicated to/To my...

Chapter 1

Introduction

Controversies and negative outlooks on video games are very prominent and have been for many years. Many media outlets and other people believe that video games are one the main reasons for negative child behaviour and social disconnection. I want shed light on video games and display all of their positive attributes. Video games may not seem to benefit everyone but I believe that there is a video game(if not many video games) that would benefit you. Think of it like reading a book or watching a film! If you are reading this chances are you probably have done one of these. Video games can be adapted such that it resembles reading a book or watching a film. This paper will cover how creating a virtual experience in the medium of games can satisfy us all.

1.1 Motivation

Video games in the modern day continue to evolve and develop. They offer a medium of expression and depth like no other. Growing up in the late 90s and early 2000s video games were in a golden era of unique titles. From a very early age I had been exposed to quite a few games including Super Mario 64 and Sonic The Hedgehog. I was so fascinated at the time by how enjoyable and realistic these games were. Based on my positive experience of growing up with games I want to share the benefits of what I feel video games can offer all of us. My passion for games throughout my life is my motivation for this paper.

I believe it is important to do a project on video game immersion for a number of reasons. The first being Educational benefits. When I was studying in primary school we had used games as a method of learning fundamentals of core subjects. My most clear memory was developing the skills to learn the times tables using an educational

virtual game. This proved to be the most efficient, fun and social way of learning this skill. As I transitioned to second level education the notion of using games to learn was quickly removed and extremely rare. Not only can video games improve motor skills and reflexes but also provide a medium to learn new languages, stimulate creativity and even learn the times tables.

FIGURE 1.1: Online educational times table balloon game at theschoolhub.ie

1.2 Executive Summary

To summarize this project will have the goal of providing an immersive game that can both provide enjoyment and inspire, especially for individuals who lack in confidence or who are introverts. The game being created aims to relieve stress and provide a medium of enjoyment and creativity. Positive reinforcement is key to making all of us feel good and enjoy therefore I want to convey this in a virtual format through the medium of a video game. The project will aim to display video games in a positive light through the means of a 2d orientated game. One which that heavily focus on single player interaction, depth in design and heavy emphasis on immersive game-play.

1.3 Contribution

My second level education unfortunately did not offer computer studies as a subject however I got the opportunity in my transition year to do an e-technology module. I created my first game in e-technology which resembled the game Asteroids. This

foundation of creating games will contribute to the core basic mechanics of how a game should function.

Throughout my college years I realized I loved designing and implementing software. The Web Development module in first year allowed me to express my creativity in designing a website of my choosing using HTML,CSS and JavaScript. During my second and third year I enhanced my coding skills using a plethora of languages. In particular during modules such as Object Oriented Programming and Programming for Data analytics. I became very well acquainted with high level programming languages, in particular Java and Python. Many game engines use their own language which are built upon the languages mentioned above. This will give a solid basis at which to understand how games are developed in these languages.

One of my more challenging modules which came in third year involved programming mobile phones. This module contributed greatly to the learning aspect that led me to do this project. Techniques such as the creation of a user friendly GUI, interoperability of software and understanding target audiences. Mobile app creation featured prominently in my third year group project as I took on the task of developing an app from the ground up for quite a complex software application. This knowledge of app design will go hand in hand in the creation of my game.

1.4 Structure of This Document

Chapter 1 describes an introduction to our project outlining our motivation and contribution our project will have. Chapter 2 details an in depth background and previous work done that correlates to this project. Chapter 3 displays out problem statement outlining the main objectives of what we want to achieve during this project. Chapter 4 illustrates how we are going to complete our game project and pinpoints the necessary software/hardware that will be required. Chapter 7 will wrap up our research phase conclusions and provide a summary of what we have achieved so far.

Chapter 2

Background

The main point of this project revolves around the development and implementation of an immersive video game title. To achieve this we will use Gamemaker Studio 2. This is a redefined IDE built upon its successor Gamemaker Studio which was created in 1999. To understand what Gamemaker Studio can provide for us we need to understand its core concepts which directly relate to this project. This include Game Applications and Mobile Applications. We will focus primarily on the former for this report.

2.1 Thematic Area within Computer Science

The core concept of developing an immersive game relies on MDA. MDA is a formal approach to understanding games - one which attempts to bridge the gap between game design and development, game criticism, and technical game research[1]. MDA as a framework allows for the creator of the game and player of the game to have completely different views when creating/playing the game.

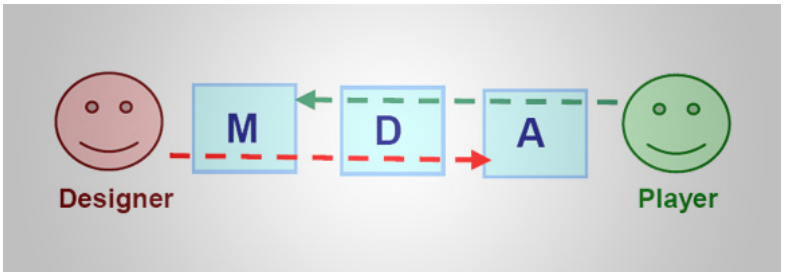

FIGURE 2.1: Core Concept of Mechanics, Dynamics, Aesthetics in games[1]

Before we can consider any other factors in the role of a game developer we must first address game mechanics. Think of this like the earths core. Video game players don't have to know what the game's rules are when they begin;unlike board and card games,

the video game teaches them as they play[2]. Based on the figure above this is our pivotal starting point for our project. As the game is further developed we delve into the dynamics portion of the game. In this case just like the earths mantle. Dynamics refer to what might be called the "behaviour" of the game , the actual events taking place and phenomena that occur as the game is played[3]. The designer begins to develop patterns into how the game should be played. Such dynamics include reaching level 2 once score of x has been met or the type of shot you take in a football game(e.g. Driven shot, Lob shot, Finesse shot) which are all based on game dynamics. Finally we have aesthetics, which is quite like the earths crust and what we are all are exposed to as players. Game aesthetics is an expression of the game experienced as pleasure, emotion, form-giving, etc.[4]. From the players point of view they will be exposed to this prior to anything else the video game may offer.

To further analyze the notion of MDA, video games can be broken into further subcategories. This will address all of the core needs in creating immersion within a game.

2.1.1 Game Physics

Game physics represent different aspects of the physical world we live in. When we talk about physics in a game, we really mean classical mechanics: the laws that govern how large objects move under the influence of gravity and other forces[5]. Ian Millington explains how physics has evolved in video games drastically in the last 30 years. If we travel back to the late 90s there are almost no game engines in which can be utilised to implement our game physics. Such basic physics such as object movement using vectors and object defined mass, volume and inertia were available, however years and decades on game physics evolved such that we had complex virtual experiences such as flight simulators and realistic driving simulator games.

To utilise game physics to the nth degree, game physics engines were developed to introduce a notion of re-usability and scalability in the development of games. A simple particle effect such as a smoke cloud could be written in a few dozen lines of code. A well developed engine may offer hundreds of such particle effects that fit perfectly into what the game developer is trying to convey as part of their game. This immediately tackles complexity and assists the game developing community to utilize its tools. A physics engine provides you with the ability to have effects interact in believable ways[5]. Constant improvements are made to the aspects of the physical world we live in that are segmented throughout a video game. Such examples including water physics, smoke and clouds.

Focusing more on game engines, the mid 1990s brought about the initial engines. At such a time, the most well developed titles involved Doom and Quake. These titles were 3d orientated to authentically create unique weapons and characters which built upon the vast amount of 2d orientated games released prior. Today, game developers can licence a game engine and reuse significant portions of its key software components in order to build games[6]. Knowing this allows for the wide range of different game genres to be implemented using our game engine.

2.1.2 Game Genre

Filtering into the topic of game genre, there is a plethora of which can be chosen from. Each of which is unique in its style of play and patterns that are followed. If we look at

Genres (15 total)	Styles
ACTION	2D Action, 3D Action, 3D Platform, Action Adventure, Ball and Paddle, Combat, First-Person Action, Fixed Screen Platform, Interactive Screen Saver, Maze, Miscellaneous, etc.
ADVENTURE	Action/RPG Adventure, First-Person Adventure, First-Person Graphic Adventure, Interactive Movie, Survival Horror, Text-Based Adventure, Third-Person Graphic Adventure
FIGHTING	2D Fighting, 3D Fighting
RACING	Aircraft Racing, Bicycling, Boat/Watercraft Racing, Demolition/Combat, Drag Racing, Extreme Racing, Formula-1/Indy Racing, Futuristic Racing, Go-Kart Racing, etc.
SHOOTER	First-Person Shooter, Fixed Screen Shooter, Overhead Free-Roaming Shooter, Platform Shooter, Shooter with Weapon Peripheral, Side-Scrolling Shooter, Squad-Based Shooter, Third-Person 3D Shooter, Vehicle Shooter, Vertical Scrolling Shooter

FIGURE 2.2: Games broken down into their different genres[7]

the broad genre of the 2D shooter (of which Space Invaders is often considered to be the first), we quickly realize that many other terms are closely aligned with it[8]. Such sub-genres exist with in the shooter genre for instance shoot em up, rail shooters and fixed shooters. This asks many questions of the developer to carefully identify what specific genres their video game falls under or has certain aspects of. To contrast we identify that a rail shooter has the player travelling through an immersive world in which they must shoot targets objects or enemies to gain score or complete a level. This sub-genre falsely allows the player to have full control as movement is prohibited and allows only the use of gun movement while the game takes care of the rest. The sub-genre shoot em up/manic shooter gives the player far more control of movement such in the game mentioned above Space Invaders. The genre features a cluster of bullets and objects which are defined in patterns to give the player a high intensity rush when playing the

game. However we also must consider the side scrolling game genre injected into Space Invaders. This sub-genre features prominently in platform games and also shooters. This genre is specified mainly by a game mechanic in that a side view camera implementation allows for the character scrolling. Even some of the earliest titles in video game history such as Space Invaders combine different game genres which fuse together to create a virtual experience for the player.

2.1.3 Game Music

Game music can be described as quite a broad topic in that we have quite a number of methods involved. Reading through the Introduction of the study of video game music describes categories such as reactive, interactive and adaptive music littered throughout video games. Quite popular reactive game music can again be seen in the earliest of game titles. The term reactive refers to a musical change in response to a singular non musical action of the player[9]. This sudden action can be identified in Super Mario Bros when the character Mario travels down a pipe or when Mario encounters a boss enemy. Usually the game asks the player to invoke a response in order for reactive game music to initialize.

Interactive and adaptive music are quite similar in that adaptive integrates from interactive music types. Focusing on interactive, some of the earliest known examples consist of a series of musical games called Musikalisches Würfelspiel/ musical dice games that came about in the mid-18th century[10]. The game represented a comprehensive range of possibilities and outcomes. The simple principles of these century old games display applicable relevancy in the interactive video games that are played today. Incorporating interactive music using looping became the best known solution to introduce musical immersion in video games. For this to work the end portion of the music track will seamlessly transition back to the start of the music track. The Ratchet & Clank video game series utilises this skill to the utmost in its original title. In the YouTube video Why The Ratchet & Clank Soundtrack Is A Perfect Fit, Aster explains "to chain these different tracks together there's these neat little transitions that segue to the next track once you step into the new location"[11]. This is quite difficult to achieve as the games composer must ensure that each track will end where it starts in order to achieve immersiveness. However developers ideally yearned for non linear music to be applied to their video game.

To carry out the optimal solution for video game music, we look to adaptive music. Adaptive music is known be quite difficult to implement into video games under software

restrictions. However in recent years it become far more applicable in video games developed today. Dynamic music is a generic term for changeable/adaptive music changes in reaction to the game state not in response to the player's actions[12]. The iMUSE system was introduced in order to have a seamless flow of music without the music having to cut out at any point. The system allows to check for specific game conditions to be met in order for a sound effect or music loop to start. The techniques linked to iMuse include vertical re-orchestration and horizontal re-sequencing. The prior allowing layers of smaller musical pieces to be layered upon one another. Thus creating an atmospheric effect or pattern prominently seen in horror games and adventure games which builds suspense. Horizontal re-sequencing uses algorithms to determine certain segments of music to be played at certain intervals during the game. There are several ways in which to approach segment selection including random selection, probability selection and specific selection from sets of audio files[12]. Some great examples are spread through games namely Tomb Raider(1996) and The Legend of Zelda(1986).

2.1.4 Game Art

Game art is a broad topic that can be broken down into two different sections. Those sections being level design and character design. Both of which are essential for the visual immersion a player will be engaged in when playing the video game. Both topics rely on one another in order for the virtual game to be playable.

2.1.4.1 Level Design

Level design relies heavily on pattern building in order to fit the genre and develop the correct experience for the game. Delving into the popular genre of platformer game titles, there is a need of pattern repetition and consistency. Rhythmic actions help the player reach a "flow" state in games, a state of heightened concentration [13]. Within our typical platformer the use of obstacles, challenging jumps and precise player movement convey a sense of challenge and complexity to the player. Reading through the 'Model of Platformer Levels'[13], the example of Super Mario game titles achieves this goal. The levels vary in difficulty, enemy spawning, object and collision placement which confront the player in a unique reshuffle every level within the game.

Since there are many design patterns a creator may chose from, it becomes quite difficult to decide on. PCG allows for the instant generation of level creation within a specified design pattern. This allows the automated levels to follow set structures. The development of new frameworks came about addressing this task such to compare and contrast to existing level design. The creation of the GVG-LG, GVG-AI and SB-LG frameworks

were introduced. The SB-LG(proved the be the most efficient of the listed framworks) is based on an evolutionary algorithm, which takes an array of tiles as input and generates a level for the game[14]. SB-LG acts as a great building block for developing the level segment of games.

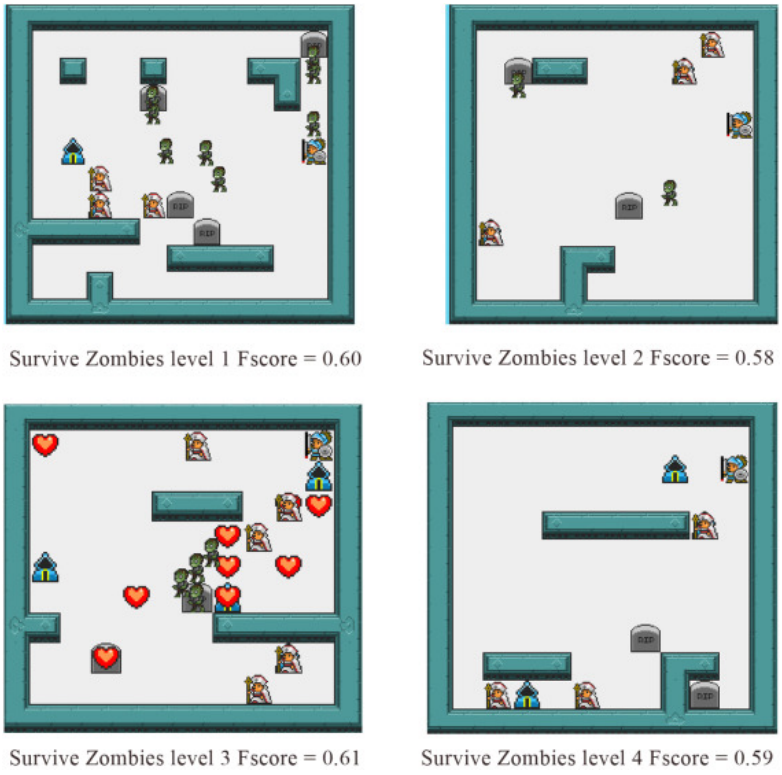

FIGURE 2.3: Search based level generation pattern[15]

2.1.4.2 Character Design

Character design has many aspects to invoke the players senses. The first of which involves creating a cognitive immersion on the character. One aspect of crafting a strong player-character is creating smooth and intuitive cognitive immersion for the player[16]. This allows the player of the game to assume the behaviours and abilities of the virtual character. Players capitalise on the characters potential in order to complete tasks, solve puzzles and navigate through the virtual world. Quite a unique example is found in the Sims games. These virtual humans come to life and are controlled by the player as to how they see fit. The game offers an ability to create a mirror image of themselves by personality looks and traits offering a very deep detailed cognitive immersion.

Animation of video game characters brings to life the sprites that are designed. 2d characters can actually show more of the physics of animation than 3d-like squash and stretch[17]. Squash and stretch can be defined as when a ball hits the ground it becomes slightly squashed and then returns to its normal spherical state. Dealing with 2d games

the z axis becomes omitted which makes it easier to give life to objects and characters through animating. Animating the game objects and characters brings about lifelike emotion and realism to the video game.

Mannerisms and habits are good ways of giving a player concrete tools to express some aspects of the character.[18]. This introduces a range of traits associated with the game character. A video game Character may have specific weaknesses and strengths. This allows the players to address problems in where situations are very difficult or impossible for the video game character. The player must then adapt and tackle these issues in an alternate way. To give the sprite more of a personality the game character may also have skills such as being athletic or being a marksman sniper. One the other end of the spectrum they may also can have frailties such as inability to swim. Mannerisms and habits convey a uniqueness to video game characters.

2.1.5 Game Graphics

To address game graphics, dimensionality is core as what we need to address. We can vary between 2d and 3d graphics in video game implementations. Many original game titles utilize 2d graphics to convey a series of images resulting in a playable title. It is argued that time has moved on and 3d games have taken over from the retro 2d style. However in a study carried out by Johanna Roettl and Ralf Terlutter "video game evaluation was not worse in the 2D condition as compared to the 3D and VR condition indicates that game developers can still be quite successful by continuing to offer "traditional" 2D video games"[19]. In actual fact people had felt sickness playing 3d games especially when game to the VR titles. The relevancy for the development of 2d games is still very prominent even in 2021.

Perspective is defined as how the player actually views and plays the game. First or third person games developed using a 3d game engine usually use the accustomed perspective camera, while some genres, like strategy or role playing games use an isometric projection model[20]. Quite a unique isometric projection camera is utilised in Sid Meier's Civilization. Seeing the uniquely created world in an isometric or birds eye view captivates the players senses. City views allow you to see a visual display of the city, and icons inform the player of his city status[21]. These simple graphical implementations reviewed by Brian Palmer brings a whole new perspective to life in virtual form. This leading to the success of the franchise with dozens of games developed and still being developed today with Civilization VI being the most recent release.

FIGURE 2.4: Graphical view of original Sid Meier Civilization game

FIGURE 2.5: Graphical view of newest Sid Meier Civilization game

2.1.6 Game Controls

Without any game controls any video game would simply not be able to function. The earliest known game-pads came about in the 1960s. The PDP-1 was used but had severe limitations to its computer design. The basic controls that the PDP-1 could utilise were some basic toggle switches. The game SpaceWar(1961) was developed as one of the first video games ever made on the PDP-1. It could utilise up to five different switches in order for player controls and features such as rockets and thrust abilities. The switches provided rotational movement but this proved flimsy and had to be adapted. This brought about the the control box.

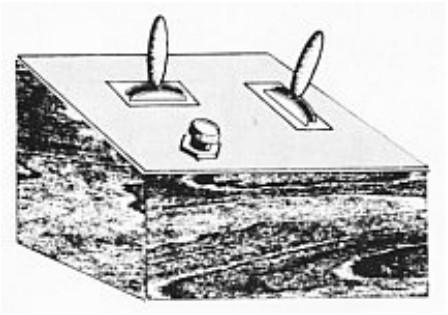

FIGURE 2.6: Sketch of control box used for video game Spacewar(1961)[22]

The control box provided stability to the players that were controlling it, prominently spread throughout arcade machines. It took the 5 switches that were needed and moved them to 2 two way switches and a single button placed on box allowing for a wired remote game controller.[22]. This development lead us to where we are now giving us the ability to control characters using modern gamepads and even motion control remotes.

Game controls today can be handled by a number of methods such as joysticks, game-pads and mouse/keyboard. In the development of 2d game titles acceleration and deceleration are utilised. By altering the attack and release phase (or, acceleration and deceleration), it is possible to create different game feel[23]. The manipulation of game character velocity releases a sense of authenticity and realism to the characters movement. The ideal scenario created ends up feeling responsive to player input. Velocity manipulation is dealt with in similar formats for certain controls. On a game-pad, holding or releasing a specific button may determine intricate velocity values. In addition to this players have access to dual game-pad sticks which can measure velocity depending on the severity of a stick being moved a certain direction. On a mouse the rotational movement of the players hands determine precise velocity values. Steve Swink determines in a visual format how manipulating velocity can introduce a fluid dynamic feel to the video game.

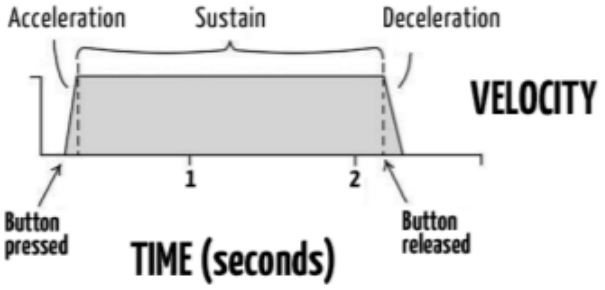

FIGURE 2.7: Responsive stiff controls for our game characters[24]

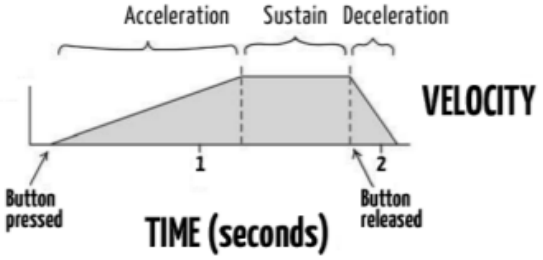

FIGURE 2.8: Loose fluid controls for our game characters[24]

Swink allows us to see similarities in human nature between human movement in the real world and how this can be transcribed over in the virtual world of video games.

2.1.7 Game Difficulty

Game difficulty contributes to how engaged the players of a game will be when immersing themselves in the title. When a player is first confronted by the game typically we see a main menu type GUI, which allows the player user input to select difficulty before initially starting the game. This is found quite frequently in most game titles. An investigation carried out involved dozens of people with their game skills ranging from casual to experienced. Several questions arose such as how much did the participants enjoy the game? This question allows us to observe correlations between player performance and their enjoyment[25]. The game was a simple endless runner that challenged players to stay alive as long as they could accumulating the most score. With a range of game difficulty. Results had displayed both an easier and hard version of the game in which both the causal and experienced players engaged in.

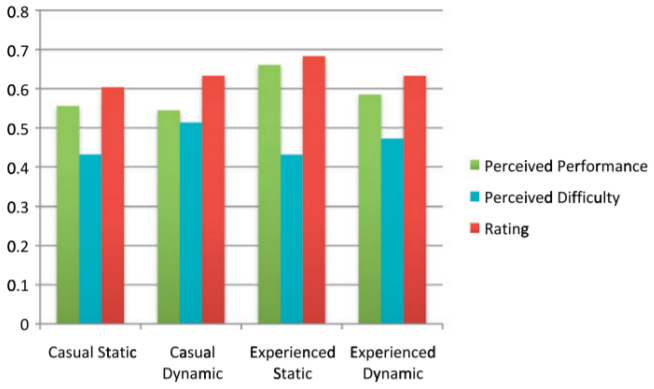

FIGURE 2.9: Casual vs Experienced players ratings on varying difficulty modes[25]

Due to the game implementing a DDA mode, results are far more stable than if the game was static in its difficulty. In simple terms ideal game difficulty is solved whereby the game gets easier for casual players the more they die while the game gets progressively harder for experienced players the longer they stay alive.

2.1.8 Storytelling

Storytelling gives developers the ability to tie all the mechanics, dynamics and aesthetics together by forming a pertinent story-line. A video game story-line provides reasoning and clarity for the implementation of the games MDA. The games core design can then shape and mould a fitting story-line, one that which can drastically immerse its players within its world.

Focusing primarily on a video games narrative, we discover the game ATUM. ATUM combines both 3d and 2d perspectives offering players both a 3d interactible point and

click scene room portion combined with a 2d orientated puzzle platforming portion. The focus in ATUM lies on this multi-layer design, both for the core game-play and for the environmental storytelling[26]. The story can then be developed and told through both the 2d and 3d strands of the game. The game includes unique storytelling in that for the player to progress they must use objects in the 3d world to help solve puzzles and traverse the 2d platforming puzzle world. All the while giving players intrigue into who the main characters of both worlds are and the characteristics tied to them. ATUM brings about a new modern twist in video games with its multi layered genre interactions and unique narrative concepts.

Characteristics for game narrative space	Function in game narrative	Function in gameplay	Examples cited (selective)
Topographical layout	Affect plot structure	Help create navigational pattern	*Assassin's Creed BioShock*
Topographical spatial opposition	Help group narrative events; Help create conventions (e.g., dangerous space versus safe space) to affect emotional experience	Help shape navigational experience; Help create conventions to affect decision making	*Assassin's Creed*; *Assassin's Creed II*
Operational character mobility object mobility	Help create the storyworld; Help the plot development; Affect the variability of narrative	Ease or intensify gameplay; affect the variability of gameplay	*Prince of Persia: Sands of Time*; *Assassin's Creed*; *Fable II*; *Heavy Rain*
Operational paths and axes	Help shape/restrain the plot structure	Help create navigational pattern; Affect the degree of repetition of gameplay	*Assassin's Creed*; *Planescape Torment*
Presentational on-screen space perspective	Stylistic/genre storytelling; Choice of first-person versus third-person help shape emotional experience	Affect the continuity of game space; Affect player control of view; Camera as a guiding device	*God of War*; *Assassin's Creed*; *Assassin's Creed II*; *Prince of Persia: Sands of Time*; *Max Payne*
Presentational acoustic space	Set the mood, create tension, and enhance the realism.	Provides feedback and hints for players' next moves; Basic game mechanic (e.g., in rhythm games)	*Fable II*; *Assassin's Creed*; *Rock Band*; *Dance Dance Revolution*
Presentational spatial segmentation	Help group narrative events	Set the boundaries of gameplay; help create game levels	*Prince of Persia: Sands of Time*; *Fable II*; *Assassin's Creed*
Presentational screen interface	Provide information; stylistic/genre storytelling (e.g., split screen)	Facilitate player control for both gameplay and camera view; provide information and instructions	*Assassin's Creed*; *Fable II*; *High Velocity*; *Heavy Rain*

FIGURE 2.10: Characteristics affecting narrative and game-play experience[27]

Huaxin Wei extracts a very detailed table of information which deeply analysis specific characteristics and functions that storytelling conveys in video games. Most titles seen above are 3d adventure titles each of which broke the glass ceiling of narrative in game development. These titles according to Huaxin followed strict characteristics which could then extract timelines for each of the games levels, chapters, cut-scene sequences and the overall game itself. Narrative gives the opportunity for players to learn the games control scheme, new abilities that unlock with progression of the game and uncover mysteries into the games protagonists/antagonists. The timeline below extracts

information from an Assassin's Creed game level and outlines the level structure with its paths and objectives. The game's narrative approach follows an indexical narration one which points the player where to go and what to do.

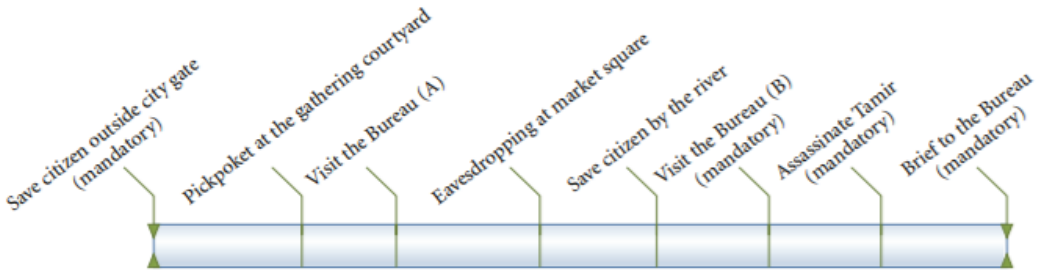

FIGURE 2.11: Assassin's Creed Damascus operational timeline[27]

Figure 2.11 determines the different paths a player may chose when playing the Damascus level. The timeline displays both mandatory and optional objectives which can be completed by preference of the player. To further progress to future levels the game outlines the player to meet mandatory objectives while allowing for other optional objectives complimenting the story of the game. This brings about a dynamic aspect to the Assassin's Creed game franchise allowing for a more expansive world with multiple possibilities. This successful narrative implementation displays that even in a linear focused game, secondary objectives can play a factor which enlarge the immersive world the video game personifies.

Following on from indexical narration, we also evaluate environmental narration. This type of narration it is a general term to refer to how spaces can evoke and construct a narrative experience while navigating a space [28]. The game Half-Life(Valve,1998) offered one of the first unique environmental storytelling aspect. The game included weapons that could shoot bullets at players, walls and other objects. The bullets would end up leaving traces on walls. Players creativity took over and painted murals and figures such as smiley faces which acted as environmental storytelling. Hence other players would discover this revealing the initial player's story being told through the games environment. While not setting the main tone of a games storytelling, environmental narration still sprinkles little storytelling aspects which improve players sense of immersion.

2.2 State of the art

This state of the art section will explain both trends we see in gaming today and provide the most relevant existing solution we have for our projects development.

2.2.1 Trends in gaming

This section will focus on relevant trends we see in video games. This will center in on trends relevant today in which will display tendencies for video game creation and its community.

Cross platform game development is a significant trend and idea today. According to "Cross Platform Game development" by Alan Thorn, he claims that cross platform games have an ability to be run on multiple platforms which can occur in one of three ways [29]. Thorn first outlines having multiple computers which provides a one to one setup of each OS instance. This method becomes fairly costly and has little efficiency in terms of testing, compiling and running each individual instance of the game corresponding to the relevant OS setup. The second method outlining a very complicated idea outlines creating hard disk partitions on one singular computer that runs each OS. The user must still reboot and restart the machine each time a switch in OS is required. The final most optimal solution being Virtualization allows for abstraction to be initiated. The goal to simulate other machines and emulate in a single system. This removes the need to reboot and restart the machine as the single OS can handle all platforms.

We cannot talk about trend in gaming and not mention eSports. eSports has become a phenomenon in today's society. The "Embodiment and fundamental motor skills in eSports" outlines the combination of cognitive,strategic, and mimetic skills are unique for eSports and can be considered a way of enhancing digital literacy [30]. The point is argued as to whether eSports is valuable for education and learning especially for young children. Developers of games have admitted that they wanted their game to be addictive for replay-ability to play a factor. This trend and idea can be translated quite dangerously as we have seen drastic measure been taken to counteract the negative aspects of eSports and video game addiction. Minors in China must now only play games between the hours of 8pm and 9pm while only being able to do so on weekends with a three hour weekly limit [31]. This drastic measure displays the negative reality of eSports and video game addiction, one that needs to be addressed extensively to suit the needs of our younger generation and portray video games in a positive and pragmatic manner.

A final trend worth mentioning is Augmented Virtual Reality. This acts as the most developed unique immersion experience we have today in video games. The players reality becomes expanded further into a virtual world. Liszio and Masuch define VR as the system's ability to shift the user's perceptive and cognitive attention from the real to the virtual world to elicit sensory and cognitive immersion[32]. However there is a need for social interaction in an alternate reality therefore VR games must rely on interactive

game mechanics to thrive. The real world around us is also consumed in background noises that influence the sensory immersive experience while in VR.

2.2.2 Existing Solution

This subsection is going to outline the most relevant existing solutions in what we want to achieve in this project. It will act as guide and inspiration to the development of my project and will setup the problem definition process we will see in the next chapter. To reiterate the aim of my project is to create an immersive video game, one which that will enhance motor skills, relieve stress by a virtual experience and piece by piece implement crucial game concepts resulting in a fully functioning video game. Many successful games resulted from Gamemaker Studio but one of the most supreme titles is Spelunky[33]. Spelunky acts as our game for the state of the art review.

To analyse Spelunky, we will map our background knowledge discovered above to the game representing each key feature that defines Spelunky as a 2d masterpiece.

Defining its genre, Spelunky is a 2d platformer adventure game. The player takes control of a spelunker avatar that explores deep underground caverns to try and win their own freedom [34]. The game takes huge inspiration from a 1989 classic 2d game Prince of Persia. The game is classified as a rogue-like genre which translates to a dungeon or cave crawler game in which levels are procedurally generated.

Spelunky is also known for its upbeat soundtrack. The game synths sounds in a basic manner, quit like NES, but its a little more sophisticated with greater polyphony[35]. The game relies on somber jazz and melodic themes to set the tone of each level in the game. Spelunky takes aspects from 90s platform game music with subtle modern twists. The game itself is known for being quite difficult with dying repeatedly being an inevitable fate. The music does a great job in contrasting the games unforgiving dungeon levels with a relaxing soothing soundtrack.

Level and character design are main features that make Spelunky stand out against its platformer rivals. Levels in Spelunky are procedurally generated in order to create a new experience for the player each time they play. Developer David Yu explains using a rogue-like prototype generation system to map grids of 4x4 that make up rooms in each level[36]. The rooms themselves have an area of 80 blocks which comprise of unique design patterns. Areas in the game include ice caves and the mines which correlate to the dungeon theme color palette of each room. The algorithm used by David is quite complex as it must figure out patterns in which the player character cannot enter an unplayable state by getting stuck or other means. A suitable level starting and ending

point must also be defined. This ensures a clear path must be generated from start to finish.

Chapter 3

Problem - Video Game Immersion : The art of creating a virtual experience

The key question to be addressed in this chapter is: "What do I want to achieve".

3.1 Problem Definition

One of the main issues with video games today is the media portrayal of the medium. Video games are presented with many positive and negative aspects each of which can be drastic in their own way. As we addressed in [31], game addiction has invoked governmental responses to take action and limit/ban video game play in China. This worrying reality may only lead to other countries following suit. I want to portray video games in a positive light and develop an application that invokes positive responses from its players.

2d games are recognised by some to be inferior to the modern day AAA 3d game. My project will also aim to convey 2d games as a form of art and entertainment employment. 3d games are indeed becoming more popular and in demand than 2d games. However we can see from examples like Spelunky and Celeste that 2d games focus on being completely unique with their attention to detail with art-style, noteworthy soundtracks and beautifully generated isometric worlds. To tackle our problem definition. objectives have been outlined below.

3.2 Objectives

The first main broad objective for this project is to create a fresh unique game developed in the 2d game engine Gamemaker Studio 2. The background portion of this report has been divided into many subsections, each of which are relevant categories that must be included in the project. To tackle this the project will involve a 2d side-scrolling platformer with a wide range of aesthetically pleasing levels each of which containing quality game mechanics. Many other sub-objective arise for this project outlined below.

The project needs an immersive narrative tied to it. This will engage its players to a distinct story that maps to the feel of the game. Voice acting may come as a difficult challenge therefore there is a purpose to add a subtitled text based narrative to tackle this.

The game must include aspects that improves motor skills of its players. For this objective I aim to implement fluid player movement alongside a plethora of easy-challenging obstacles that the player must overcome. This will engage and test the players reaction and also flow into the next objective.

This objective mainly revolves around puzzle solving. The levels in our project will follow unique patterns and structures creating puzzles for the player. This aim will allow for its players to absorb the games intrigue and become thought provoking to tackle the challenges within it.

More broadly speaking, the game that is being implemented will follow a scrolling level type structure. Such as seen in 'Mario' game titles, levels will start its initial state on the left side of the screen. The camera will work in tandem with parallax backgrounds creating a sense of depth and 3d aspects to the game. Each level will conclude on the right side of the screen after the player successfully beats the level. This structure can be modified such that levels may scroll from right to left or even climbing vertically to add some level diversity. Each level will include a theme, color palette and unique enemies which correspond to the type of level we are playing. Parallax backgrounds and tile-sets will also add to this theme. Enemies scattered throughout levels will have varying stats such as health and damage. Enemies will also get increasingly difficult as the game progresses. Music and Sound effects will also be implemented to convey the games tone. Adding sound effects will give our main character a meaningful sense of existence as well as bringing to life our other game objects within the game. Scattered throughout each level will also feature some interactible items for our player. This will range from ammo for our gun to health pickups that replenish our main player' health. Non interactible objects will feature as an aesthetic enhancer that works in coalition with our tile-sets and background to ultimately create our gratifyingly designed levels.

The game will additionally feature a level transitioning object that will serve as our main transition to take our player from level to level.

3.3 Functional Requirements

This section covers every functional requirement needed for this project in order for our goal to be achieved.

Interactible Main Menu

Implement an interactible main menu that includes save and load features for the game.

Control Scheme

Implement a control scheme feature which progressively teaches the player game mechanics as they traverse through the game.

Sprite Movement

Implement player sprite movement both horizontal and vertical for basic traversal of levels.

Sprite Animation

Animate player, enemy and other game aesthetics to bring the game to life adding a sense of realism.

Player Design

Implement suitable player design to fit the games narrative and aesthetic.

Enemy Design

Implement relevant enemies that tie in to the story, look and feel of the game.

Parallax Backgrounds

Implement parallax backgrounds creating an endless level immersion while following an applicable color palette for each level.

Non Interactible Aesthetics

Implement non interactibles for each level in the game e.g trees and bushes. This task takes into account the levels aesthetic and design.

Interactibles

Implement interactibles for the player to discover e.g coins and health-kits. Interactibles will fit game design and map accordingly.

Tile Sets

Tile sets will be used to design each individual level and act as the boundaries in which the player can roam.

Level Transitioning

Implement level transitioning for completion of each level.

Scrolling Levels

Implement side scrolling levels allowing for dynamic level size and levels of varying length and time to complete.

Camera Implementation

Implement a dynamic main camera that will follow the player through the side scrolling level.

Level Difficulty Curve

Implement a difficulty curve progressively challenging the player more the further they advance through the game.

GUI

Implement a clear GUI displaying player information including score and player health information.

3.4 Non-Functional Requirements

There a quite a few non-functional requirements in which need to be addressed for development of this project. Those relevant in the creation of this game include Availability, Reliability, Compatibility, Operability and Accessibility.

Availability

This requirement will outline who has access and permission to play the game. Each game deployed needs a corresponding age classification rating to determine its suitable player base which is compulsory for this project.

Reliability

This requirement outlines safe and secure coding and development practices. My project aims to minimise issues such as memory leaks while writing clear concise commented code.

Compatibility

This requirement highlights the ability to interact and play the video game using multiple controller types. This will include options for mouse, keyboard and gamepads to engage in playing the game.

Operability

This requirement defines the operating systems that will be able to be utilised to successfully run the project. This will include full operability for windows 10 UWP. The cross platform IDE Gamemaker Studio 2 includes interoperability between several operating systems while also releasing options for porting the project to mobile either Android or IOS.

Accessibility

This requirement will enable for save and load features for the project. A player will have the ability to save the game and load the game from a file/server to continue from where they left off.

Chapter 4

Implementation Approach

The key question to be addressed in this chapter is: "How do I plan to achieve what I have outlined in the previous chapter".

4.1 Architecture

Describe the architecture of the solution that you have in mind, including:

- Technologies involved (e.g., frameworks, programming language).
- The hardware needed to develop the project (and to support at deployment stage)

FIGURE 4.1: Gamemaker Studio 2 Cross Platform IDE

The core architecture required for the development of this project involves using the cross platform game engine - Gamemaker Studio 2. Gamemaker utilises its own programming language called GML. GML was originally written in the language C++ and

uses this at run-time, however GML combines aspects of JavaScript language and the C programming language to create an easy to use, readable scripting coding language seen to its users in the IDE. Both drag and drop alongside custom coding are initial features when starting up a new project. This project will employ use of the custom coding selection to offer far more versatility and control in development of the game. Gamemaker is available on both MAC and Windows operating systems, the latter of which we are going to be using. Gamemaker Studio allow for a wide range of supported platforms for deployment of projects, although during the duration of this particular project, deployment of the game will not be necessary. Instead usage of the zipped project feature within the IDE will be used to show results.

4.1.1 Direct3D

Gamemaker Studio 2 primarily use Direct3d for Windows users while opting for other API's for different Operating Systems. Direct3d will be at the forefront for graphic API rendering in our project. Gamemaker Studio 2 has a requirement stating the minimum DirectX version needed for game development is DirectX version 9. Integrating this API automatically will authorize numerous unique features in our games development. For instance DirectX includes graphical features such as Z-buffering and W-buffering. This will assist in creating 3d object effects and allow depth for our game objects in our 2d platformer game. Some more notable features of DirectX are listed as texture blending, color blending and alpha blending. This allows for partial/full transparent detailed images and sprites to be carefully designed and rendered resulting in our composite image. This will ultimately act as our base API for generating vector images, rendering bitmaps and magnifying images to suit the style/ patterns of our game.

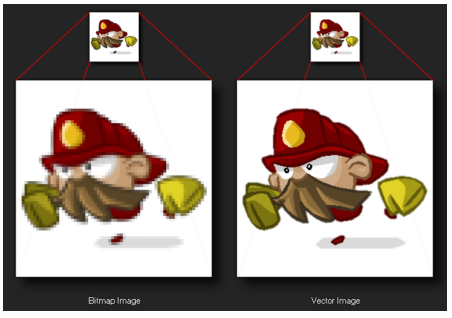

FIGURE 4.2: Gamemaker Studio 2 Bit-mapping

4.1.2 GML

Game-Maker Scripting Language will be the primary coding language used in the development of the project. GML is a very flexible language one that will define the success

of this project.

4.1.2.1 Resource management

This section of GML' language defines the main resources that will be created throughout the game. Rooms in GML act as game levels and are a core essential to allow for any Gamemaker game to run. Capitalising on GML these rooms will hold our game objects as instances. This will allow us to design and introduce sprites which will act as our visual representation of our game objects. GML sprites resource will also give us the ability to translate bitmaps to high vector images or scale the images to a defined pixel size. Once we create the images in the sprite editor we can then map these to objects. After-which these instances can be added to our game room and used as we see fit.

Rooms in GML also give us the ability to define level backgrounds, tile-maps and layers. Layers will function side by side with backgrounds in order to add depth to our level once more object instances are added. GML layers can be split up almost in packages to easily manage different resource types in our levels. This will allow us to stack layers upon other layers therefore allowing sprites, backgrounds and tile-maps each to have their own specified layer. This type of functionality will allow for precision detail and aesthetic enhancement for each of our levels. Tile-maps then permit us to take a collection of sprites which suit the aesthetic of the level we are trying to create and map them to our room/game level.

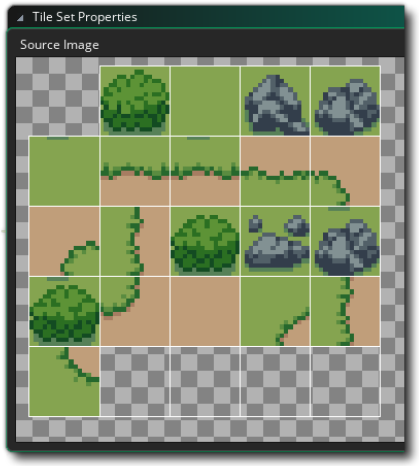

FIGURE 4.3: Gamemaker Studio 2 Tile Sets

GML also provide us with detailed audio resources. The sound formats in which Gamemaker Studio 2 capitalise on are ogg, mp3 and wav sound formats. The game engine integrates basic audio functions that can be defined in our code to play SFX and music either throughout a level or when certain conditions are met. Audio emitters are

also a key feature here which will allow us to modify categories such as pitch and Doppler effects. This also means 3d audio becomes a possibility when implementing sound in our game. GML contains audio groups for us to create categories for our sounds. This gives us a more structured project as we can map SFX and music to different categories as well as structuring different types of music we may implement. For example calm, angelic music mapped for ice themed levels and anarchic, upbeat music for lava themed levels.

4.1.2.2 Collision/Movement

GML allows for a range of different collision and movement types. At its most basic functionality we can set object instances at specific positions in a room. Game objects will have a defined collision mask set that can detect collision with another object and their collision mask. We also have the option to use path finding as well as define collision based on vector movement using a wide range of functions. This section will be very important to the project when setting up functionality of our main player in the game. Additionally this will be used to define behaviours in our enemies, translations and core object functionality such as bullet collision and player collision with ammo crates. Finally on completion of a level there will need to be collision implementation to detect when the game-state should transition to a new room/level.

4.1.2.3 GUI control/Drawing

Drawing/GUI implementation is very important in our project. We will use this section to define our main menu and other text that should be indicated to the games players. Gamemaker Studio 2 provides us with a huge range of drawing design options. This is defined and not limited by Particle effects, in depth textures and Shaders. We can also use GML' basic draw features to display stored variables on screen such as score or health. There is a healthy amount of customization with font types, color palettes, size and transparency. This will enhance our game aesthetic and provide our games users with a suitable interface to interact with and display relevant game information. A final note on Drawing is that GML has its own in built particle system. This could serve as a hidden gem in our project to code up pleasing visual aspects throughout our game levels. This will have to be used somewhat sparingly as to not clog up CPU usage and possibly reduce the defined FPS for the game.

4.1.2.4 Camera

Gamemaker Studio 2 has a view port implementation where our camera controls can be set up. The cameras in GML are quite smart in using functions to be able to detect width, height and orientation values. This means in our project we will set up our own customizable dynamic camera for each room in the game. The cameras also have the ability to be scaled at certain values to permit zooming in and out of what we want to focus on.

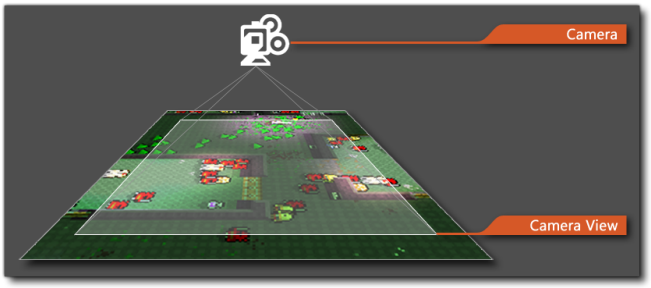

FIGURE 4.4: Gamemaker Studio 2 Camera Controls

4.1.2.5 Scrolling Levels

4.1.2.6 Control Types

Gamemaker provides several different control types that the player can interact with when playing the game. This includes keyboard input, mouse input, gamepad input, device input and even virtual key input for touchscreen functionality. Keys and buttons are binded into functions in each of these different control types. This feature will be vital to how our game will function and what devices we can make compatible for our game. Keys and buttons will also have the ability thanks to GML to check whether they are being held, pressed or combined with each other for a more dynamic control schematic. Figure 4.5 displays a detailed view of the gamepads button mapping.

4.1.2.7 Variables/Arrays

Variables in GML can be defined both in a global scope and local scope. The IDE allows for variable functions that we can use for our games specific instance variables such as score or health. Arrays on the other hand also come with array specific functions that we can use for 1d, 2d and 3 dimensional arrays. This will allow for us to structure our code far better by classifying game aspects in arrays such as level types and enemy types.

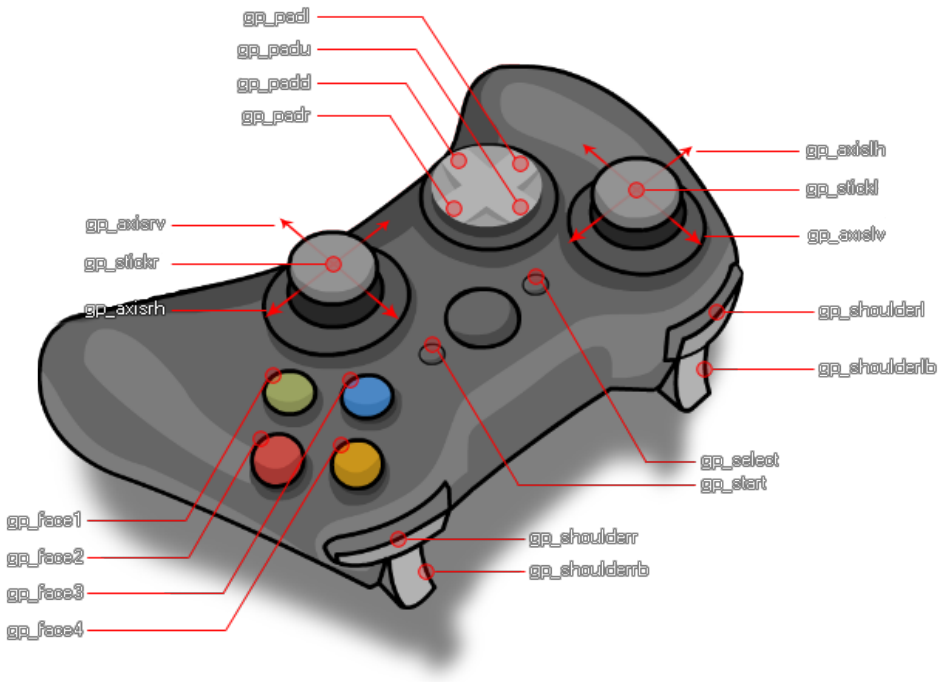

FIGURE 4.5: Gamemaker Studio 2 Game-pad control schematic

4.1.2.8 Physics

GML' Physics is very detailed and can be applied to each instance of an object in our game. Gamemaker allows us to define important physics parameters such as friction, density, damping and Restitution. When defining our game objects we can state the object type whether its static, dynamic or kinematic. There is also functions provided so that we can define accurate velocity and direction. When we accumulate all these physics aspects together this will result in functional mechanics and game objects in our game.

4.1.2.9 File Handling

File handling is pivotal for our replayability factor in our project. Our games state must be saved in order to have the ability to come back at a later point and start up again from where you left off. GML handles this by introducing a file handler. This handler can manage with INI(Initialization) files, text files and binary files. Each of which are provided with functions for read, write and open operations. Due to the fact we are working on a windows system the saved files will be stored in the localappdata/¡Game Name¿ directory. This is the location where we can perform our read/write operations in our project.

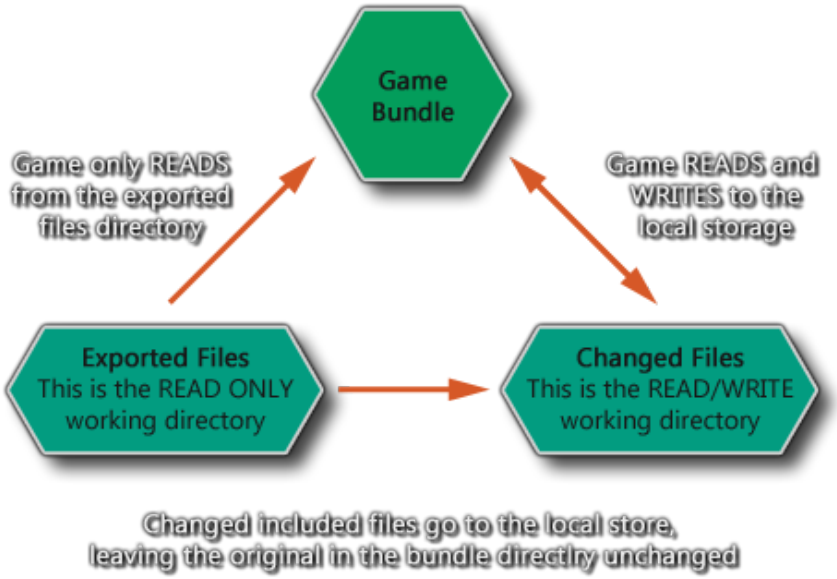

FIGURE 4.6: Gamemaker Studio 2 file system Sandbox

4.1.2.10 Debugging

After each objective is implemented in our game there will be a requirement for debugging to take place. Gamemaker studio 2 has an inbuilt debug module used to detect compiler and syntax errors. It deals with all forms of errors found in strings, arrays, functions, miscellaneous errors etc. Debugging will also give us a chance to see values in game such as rendering time, update speed and FPS. The debug module will also display hardware information for instance, CPU/GPU usage. Using this feature will ensure that we can outline any issues or bugs we encounter through the course of this project.

4.1.3 Hardware Requirements

Figure 4.7 outlines the main hardware requirements that will be required for development of this project. There will be a need to consistently update and maintain system drivers as Gamemaker Studio 2 gets numerous feature updates every couple of months. Screen resolution will also be required to be minimum of 1024x768 in order for the run-time screen to be displayed correctly.

Provide a high level view of the system you have in mind, including any package of classes, what is it responsible for and what other packages it communicates to. Provide a high level view of the database (or structure) needed to support the project, including what each table/document is responsible for and the hierarchy among them. You need

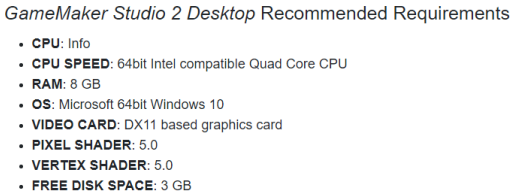

FIGURE 4.7: Gamemaker Studio 2 Hardware Requirements

to be as specific here as you can, why? Because this will aid you in identifying parts of the project you are vague on, this may be fine for some components but cause problems in term 2 for others. If you have hardware element in your project this is also where you provide a high level view of how these elements integrate into the project. So for a project that is cyber-physical you will have both a hardware and software architectural diagram. N.B. This is NOT a full system design but a high level overview of what you can credibly develop. This architecture should be informed by prototyping activity.

4.2 Risk Assessment

Table 4.1 will outline each risk associated with this project. The risks are classified in frequency of occurrence and consequence to the project. This ranges from rare to frequent frequency and also ranges from fatal to minor in respect to consequence to our project. This will give us a better understanding of the more pertinent risks that may arise during the development of our project.

TABLE 4.1: Initial risk matrix

Frequency/ Consequence	**1-Rare**	**2-Remote**	**3-Occasional**	**4-Probable**	**5-Frequent**
4-Fatal					
3-Critical					
2-Major					
1-Minor					

4.2.1 RISK1 - Time Management

Frequency == Probable

Consequence == Fatal

The most fatal risk that can happen during this project is the time management factor. The short time frame allocated for this project requires us to meet strict deadlines and

move on to further objectives regardless of incomplete previous objectives. However this may prove fatal as game objects rely on one another to function. For instance it will not be possible to move on to Parallax background implementation without functioning camera controls. A problem may then arise where a specific weeks work is inaccessible until prior weeks objectives have been met. To summarize elements cannot be completed discretely they must be completed in a defined sequence.

4.2.2 RISK2 - Source Code

Frequency == Remote

Consequence == Fatal

Source code for my project plays a massive risk factor as it acts as my primary access to develop, test and run my game. There is remote probabilities that the Gamemaker Studio 2 IDE may corrupt the source code, the unfortunate instance of my development device breaking down or a forced update that changes the layout of the code structure leaving the project in a legacy state that may have issues at run-time. Forced updates may instruct that legacy editions of Gamemaker Studio 2 has security issues therefore the possibility of having to learn new/altered GML language may cause issues. This risk however is greatly reduced with the ability to export project versions after completing objectives and also the use of Git to manage, support and backup the project so that it can be accessed at different development stages.

4.2.3 RISK3 - Memory Leak/Issues

Frequency == Occasional

Consequence == Fatal

Memory leaks in games can occur quite often at vulnerable points of the game. With thousands of lines of code human error plays a big factor where memory leaks can arise. This can be seen throughout some of the most popular games of all time such as reaching level 256 in Pacman. The 8 bit value 255 or 1111 1111 had been reached therefore no memory was allocated to generate level 256. Another potential risk could appear for eg. at a score counter that increments after conditions have been met. If the Integer defined score counter exceeds 2147483647 score then a memory leak arises. This issue primarily arises if RAM is not deallocated when the assets and objects are finished within the game.

4.2.4 RISK4 - Cost Issues

Frequency == Remote

Consequence == Critical

In a recent update in August 2021 YoYo Games the creators of Gamemaker Studio 2 stated that they will release an expanded free version of their product. There still lies a risk where the creators may incur a cost fee for some specific features within the IDE. This may prove to be a critical consequence should the risk arise during project development.

4.2.5 RISK5 - Fun Factor

Frequency == Frequent

Consequence == Minor

Many games are developed that can be classified either as successful releases or unsuccessful releases. In video games the level of difficulty and work may not always be quantified by success. A simple game such as Flappy Bird accumulated success for its game-play despite its extremely limited features. In retrospect a feature heavy title in development for years with numerous game mechanics may fall short in its fun factor. For this specific project the fun factor is taken less into account and more focus is prioritised on the objectives and feature implementation. Therefore this consequence becomes minor but important to notice that this issue is critical in the game deployment stage.

4.2.6 RISK6 - Collision/Miscellaneous Glitches

Frequency == Probable

Consequence == Major

During the development of this project it is probable that glitches will appear in the game. Bug fixes and updates are consistently required in almost every game deployed no matter the scale of the project. There will be a need to minimise and identify these glitches to provide a smooth game-play feel to the games players. Glitches themselves do not hinder the player from playing the game but if they start stacking this can be quite a major issue and place the game in an unplayable state. Therefore objective implementation will come at the cost of introducing probable glitches in our game and also filters into our next risk.

4.2.7 RISK7 - Cheating/Exploitation

Frequency == Occasional

Consequence == Minor

Due to fact that this game features single player aspects, any cheating and exploits will not directly affect other players experience playing the game. This means this risk can fall into a minor risk category. Players may try to exploit platforming games by skipping levels or finding exploits for example infinite health glitches or infinite ammo glitches. However even if such game glitches did exist it is up to the player how they want their experience of playing the game to be.

4.3 Methodology

For steady progress of my project, using the Kanban method style in Trello assists to keep track of tasks that must be completed. The trello board divides these tasks into states including ready, done, to do and doing states. Trello notes can easily be transferred from state to state allowing for clear visualization of tasks in a timely manner. Time-frames will also be defined for each objective to determine at what point we must move on to the next task. This software will give us a suitable overview and time-frame to try complete my project objectives.

4.4 Implementation Plan Schedule

This section of the paper will outline the implementation plan schedule for the upcoming semester detailing how the work will be completed over the coming months. There will be objectives each week that will be completed that in turn filter into new objectives in future weeks. The game acts as a set of building blocks each of which depend on each other. The aim being to stack as many of these objective blocks upon one another with minimal hindrances. Prior to starting week 1 there will be a need to familiarize myself with the IDE and read up on the documentation. Figure 4.8 outlines my schedule:

4.5 Evaluation

The game will be evaluated based on a bespoke survey which will have specific questions linked to all objectives and functional requirements outlined previously. This will allow

FIGURE 4.8: Planner for project implementation

an overall evaluation of the project deliverables in these key aspects. A weighting will also be used for some of the key objectives for example our play-ability, immersiveness and level transitioning.

4.6 Prototype

Figure 4.9 is a basic rough implementation of how the player character sprite will be generated in a level. The prototype also includes collision between the player sprite and

the ground block object. There is also some basic animation for left/right movements of the player sprite shown in 4.10.

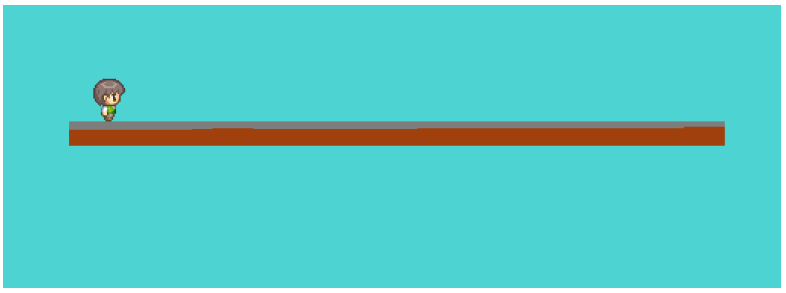

FIGURE 4.9: Game prototype showing player collision

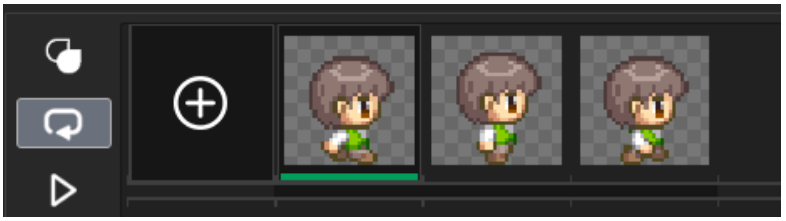

FIGURE 4.10: Sprite animation for left/right movement

Figure 4.11 displays a rough wire-frame template of the main menu screen that the player will be initially greeted with when running the game. The wire-frame includes core menu features such as the new game and load game functionality which will need to be saved/loaded to and from a file.

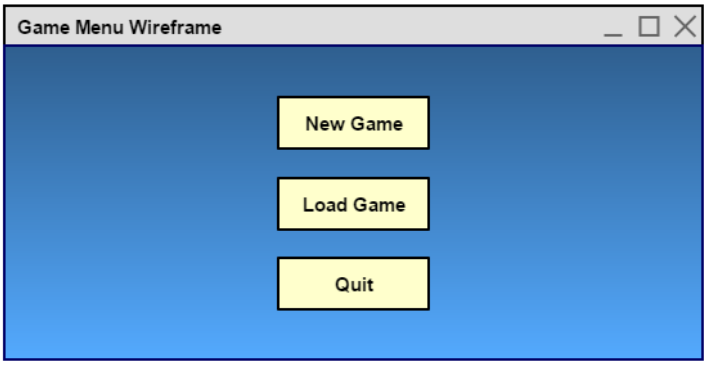

FIGURE 4.11: Template wire-frame for main menu

Chapter 5

Implementation

This chapter will detail both the difficulties encountered and solution approach of the project.

5.1 Difficulties Encountered

During the tenure of developing this project quite a large number of glitches/bugs appeared. Each of which will be detailed below with a category ranging from easy,medium and hard depending on how difficult it was to solve the issue. These difficulties encountered follow order from when they appeared starting from earliest to the most recent issues.

Week 1-5
Player sprite gets stuck in environment = medium

The first issue I encountered with the project occurred whereby the player object would get stuck inside of another object.

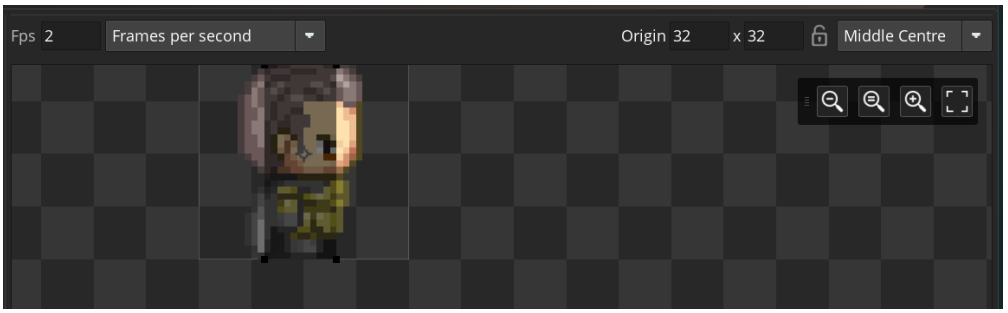

FIGURE 5.1: Gamemaker Studio 2 Collision Masking

As shown in Figure 5.1 the greyed out area defines where the player would collide with other objects in our game. The bug was challenging to discover as when the player walked the opposite direction the sprite would flip its x axis in order to successfully move left and right. When the player swapped images moving from left to right, the collision box was changing by one pixel. This caused a collision glitch with any wall objects or barriers that appeared on the x axis in our game. This proved to be quite an issue as the game would completely crash due to infinite loops of collision checks. To solve this issue I set my own manual collision masking coordinates for sprites which absolved the issue of colliding with our wall objects in the game.

Sprite animation bug for collision masking = medium

Following on from our previous bugs, another collision issue arose in the project. This was made evident when introducing enemy objects in our game. The enemies would flip their images once they collided with a wall object. However the enemy sprite would phase forwards quite a bit causing both a visual and mechanical glitch.

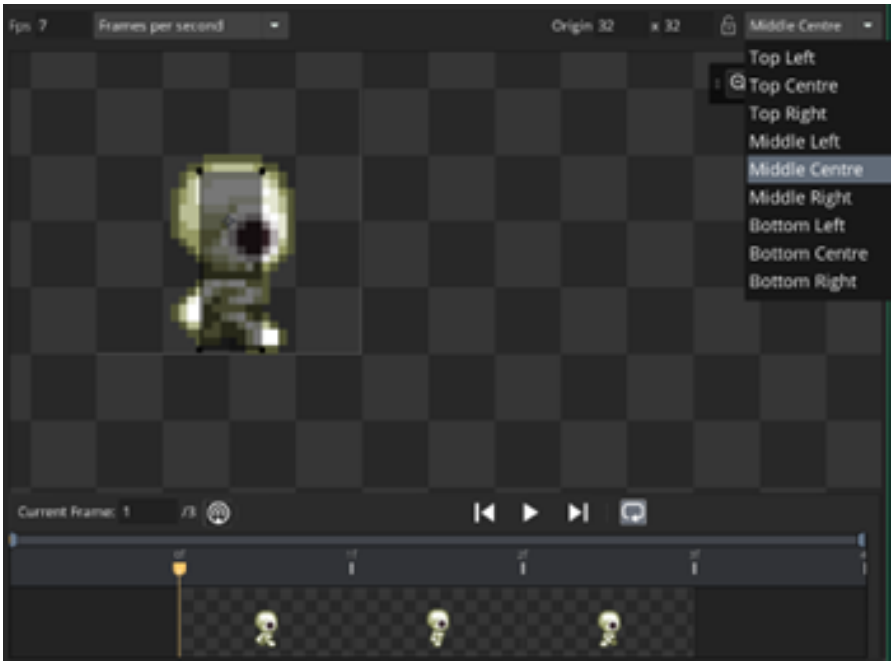

FIGURE 5.2: Gamemaker Studio 2 Sprite Origin Bug

Figure 5.2 displays the collision boundary of the enemy sprite. Gamemaker Studio 2 defaults all sprites to have a collision origin point if 'Top Left'. This issue was resolved once I changed the sprite origin point from 'Top Left' to 'Middle Centre'. Around this stage early in development, I started to realise the depth in detail that was required and portrayed in Gamemaker Studio 2' IDE.

Collision glitch with enemy as player dies instantly = easy

A Collision issue starting occurring between the player and enemy, the player would instantly die upon colliding with the skeleton enemy as the game determines collision on each frame. This issue was one of the more straightforward and simple bugs to overcome in early development of the game. Figure 5.3 shows a simple function which sets our invincible variable to 60. This gives the player 1 second of invulnerability from all damage in the game.

```
function oPlayerDecrementHealth()
{
        if (invincible = 0){
        global.hp = global.hp -33;
        invincible = 60;
    }
}
```

FIGURE 5.3: Gamemaker Studio 2 Decrement Health Function

For each second that the player is active in any of the rooms in our game, the invincible variable will count down from 60 displayed in Figure 5.4.

```
if invincible > 0 {
    invincible -= 1;
}
```

FIGURE 5.4: Gamemaker Studio 2 Player Event

Enemies would fall off the stage while reaching edge – easy

While enemies in our game could flip their sprites if they collided with a wall object, they would end up falling off the stage once walking too far over an edge. However in our game I wanted the ability to have enemies fall off platforms but to be intelligent enough not to fall off the stage completely. Solving this issue I introduced two new Boolean variables I called 'grounded' and 'stayonledge'. Each of which granted enemies the ability to make a decisions based on what confronted them. This solved the issue and introduced the first segment of enemy intelligence and adaptibility within the game.

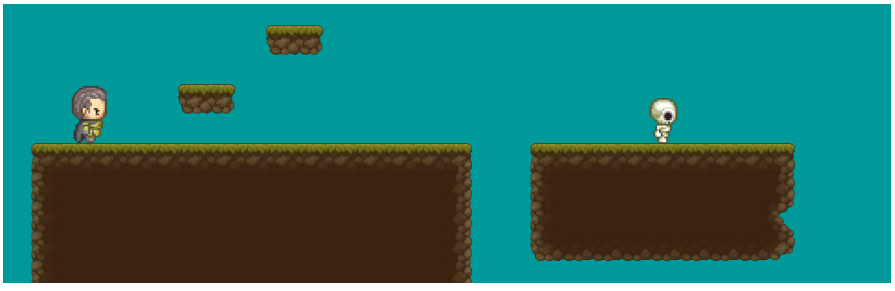

FIGURE 5.5: Gamemaker Studio 2 Beta Run-time

Week 6-9

Glitched camera GUI variables = medium

During this time period of the project, a camera implementation was introduced for our game. The camera was set up to follow the players x and y coordinates and update camera positioning every frame depending on where the player moved. This however introduced a couple of bugs, mainly GUI related issues.

The first of these coming in the form of bugged GUI elements. This being relevant game information being drawn to the screen including player and enemy HP. This glitch appeared after extending the room width and due to the camera positioning being updated to follow the player, the GUI elements started to veer off-screen. To fix this GUI issue I changed the 'DRAW-GUI' event to a 'DRAW' event. This allowed for the GUI elements to then follow its relevant object in game. Some game information such as 'Remaining Bullets' was kept in the DRAW GUI as these variables did not need to update its position and therefore could be drawn at the same coordinates persistently.

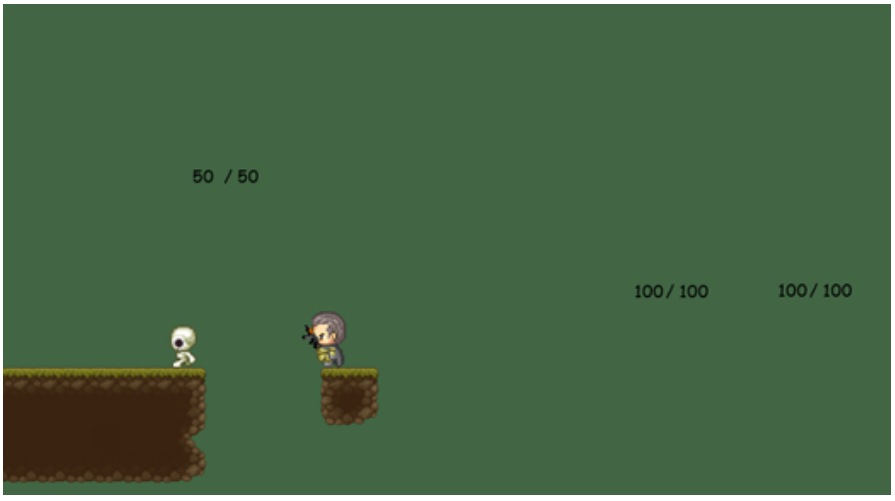

FIGURE 5.6: Gamemaker Studio 2 Camera Implementation

Glitched game sprites = easy

After fixing the previous game related issue, another one emerged. This caused a sprite related glitch whereby game objects would draw to the screen correctly. These game objects functioned correctly but appeared invisible as highlighted in Figure 5.7

This was solved by simply adding the line 'draw self()' into the draw event. This in-built function detected the relevant sprite assigned to the current object and drew it to the screen. The sprites were then correctly drawn to the screen.

Preassigned variables would not save/load – medium

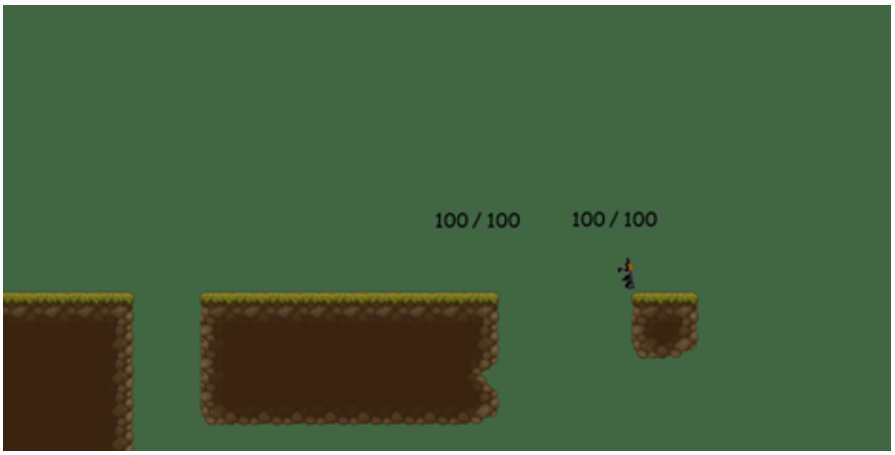

FIGURE 5.7: Gamemaker Studio 2 Sprite Glitch

After our main menu was developed, our game needed to save some other game data such as player ammo and player hp. Opting to use INI files to save game data, quite a complicated issue made itself known. Our important game data were set to global variables and pre-assigned once the player started the game. This meant variables such as player HP was being overwritten by the default specification in the games object. This was quite a significant bug as it impacted both our functional and non-functional requirements for the game both in relation to 'Interactible main menu' and 'Accessibility' respectively.

```
[Save1]
playerHP="100.000000"
room="1.000000"
gunAmmo="17.000000"
```

FIGURE 5.8: Gamemaker Studio 2 Saving/Loading Issue

Resolving this issue took a bit of a workaround and slightly changed how the game worked. I introduced a new function which set all default global variables. The game was then able to check to see if a save file existed and loaded those variables into memory. If no save file existed then global default variables would be set and the game would run as normal. This was quite an important issue to resolve both in case of my functional/non functional requirements and future game objectives such as game interactible objects(Med-kits, Ammo Boxes).

Controller button mapping issue = Hard

During the project, there was persistent development of controller functionality. This let to quite a number of issues when mapping the buttons using GML in built functions. As seen above the controller layout schematic is specified in our implementation approach in Figure 4.5. The stick and pad buttons would not map correctly and would lead to incorrect button mapping. The jump button for example was mapped using the function 'gp face2' which corresponds to the 'B' or 'circle' button. This function actually ended up mapping to the 'X' or 'A' button used to jump. The stick and D-pad buttons however would not work as intended when using the correct GML functions. D-pad buttons were completely inoperable and would not function at all while stick or axis constants would be also bugged. The axis constants would scroll uncontrollably within the games menu making it extremely difficult to chose menu options while on controller. While in-game the gun controls would be extremely bugged due to the gun being set up to follow the x,y coordinates of the mouse. This bug impacted the projects non-functional requirements in regard to 'Compatibility', while also representing risks for functional requirements of our 'Control scheme'

To try resolve these issues I mapped the 'B'/'circle' and 'X'/'square' buttons as workarounds to navigate up and down the menu while on controller. After weeks of attempting to fix the gun mechanics on controller I could not achieve a solution after several weeks of attempts. The main issue appeared to revolve around controller dead-zone issues. Due to the nature of this issue and the time consumption it held I made the decision to halt any further progress of controller development and focus on completing my other weekly objectives and requirements.

GML Shaders Issue with DirectX12 = Hard

During the development of my project I upgraded my laptop which included the new DirectX12 Graphics Card. This introduced a serious bug with my project in relation to shaders. Shaders would allow the games players to visually perceive game information such as when our player took damage or when bullets collided with enemy objects. A simple function was set up to allow for shaders to appear on screen under certain conditions for a set period of time.

Figure 5.9 displays the code necessary for GML to capitalise on our DirectX12 graphics card for enhanced graphical effects. While in run-time a fatal error message would appear crashing the game and often requiring a force quit using the task manager. After researching the issue and analysing Gamemaker Studio 2 forums it became clear that GML was incompatible with the new DirectX12 graphics card. This issue affected our functional requirements for GUI control/Drawing as players would not receive the correct feedback.

```
//
// Simple passthrough fragment shader
//
varying vec2 v_vTexcoord;
varying vec4 v_vColour;

void main()
{
    gl_FragColor = v_vColour * texture2D( gm_BaseTexture, v_vTexcoord );
    gl_FragColor = vec4(1,0,0,gl_FragColor.a);
}
```

FIGURE 5.9: Gamemaker Studio 2 Shader Boilerplate Code

To solve this issue I switched my development device to test if shaders would on older DirectX graphics cards. While pulling my most recent changes from GitHub and running the project on a different environment the issue was resolved. The project would successfully run on the DirectX12 graphics card. This however affected my ability to retrieve feedback as further down the line when publishing the game online I made the decision to discard shaders entirely as it would crash anybody who played the game using DirectX12 Graphics cards.

Week 10-13

Signpost text visual bug = easy

When introducing signposts in our game, visual bugs started to appear such as text veering off the side of the screen and text would scroll horizontally continuously. Figure 5.10 displays how signposts currently behave in our game.

FIGURE 5.10: Gamemaker Studio 2 Signpost Implementation

Fixing the signpost visual bugs involved declaring an x,y center point where the text would become center aligned around that instance of the signpost. This now allowed for line breaks in the text which would permit multiple lines of text on signposts. Text would

only veer off-screen when near a room boundary, therefore solving this secondary issue in a simple manner involved placing signposts that did not overlap any room boundaries. I made the decision to abstain from adding extra parameters to prevent off-screen text as it would not affect user experience, while also providing extra time to focus on other requirements and objectives. Tackling this problem solved some later objectives such as storytelling as NPC objects were implemented similar to how signposts function. This both solved the functional requirement of 'Interactibles' and our storytelling narrative objective.

Music looping flawed = medium

During later weeks in development, adding SFX and music introduced some issues. This was not game breaking but a serious issue impacting user experience and immersion while playing the game. The issue revolved around music not looping properly on itself once certain had finished or transitioned to a new song. I decided to use a software tool called 'Audacity' which was not previously mentioned in any capacity in our problem definition.

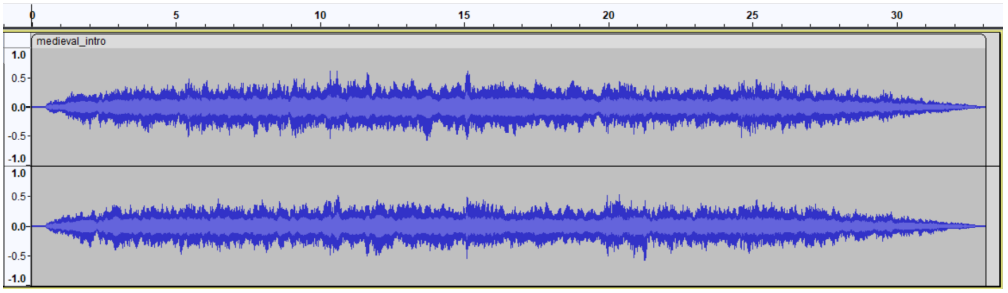

FIGURE 5.11: Audacity Editor IDE

Audacity gave me the ability to convert all audio to .OGG which compressed the SFX and music to minimise the games data size. Audacity also solved the issue of music looping poorly by adding fade in/out effects as well as necessary splices in certain tracks. This solution can effectively be noticed most prominently during the menu to Level 1 transition phase in the game.

Underwater collision glitch = Hard

Our final game related bug came in the form of further collision detection issues quite similar to bugs encountered in early development. The player character sprite would sometimes glitch while swimming underwater phasing into other objects such as wall and ground objects. This issue would effectively freeze the game and a force quit would be necessary to exit. This was caused by the sprite changing its collision mask when entering and exiting the water. This game breaking glitch was quite a risk as a fix would need to be necessary or the unique game feature would need to be discarded entirely.

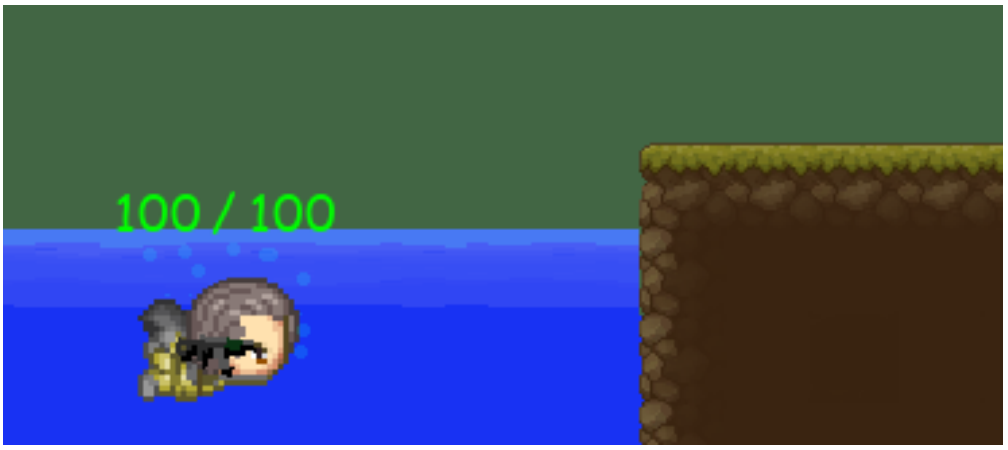

FIGURE 5.12: Gamemaker Studio 2 Swimming Animation

As a temporary fix I decided to use the same collision mask for each player animation. This included idle, walking, jumping and swimming animations. This in fact made the game slightly easier as the player hit-box for being damaged was decreased in size. However this issue was not fully resolved in time before releasing the game online for user feedback. Fortunately due to my understanding of the MDA framework detailed in the background portion of this paper I understood that this issue was masked from the games players and was indeed only exposed to the developer' perspective of the game.

Exported game issues on 'itch.io' = Hard

To publish the game, I opted to use a website called 'itch.io' which allows the free publication and use of games. Our solution approach below will go into more detail into how this was achieved. Once our game was published the game dimensions would would not fit on certain devices. This meant for some users playing the game parts of the rooms were off-screen and could not be seen. A later addition of mine objects also appeared to be bugged, both the SFX and functionality.

To act as a fix I revisited the code and reduced player HP by having the mine collide with the player rather than vice versa which was how it was originally. This bug was quite strange as the game objects worked perfectly in the IDE run-time and debug mode. Due to time constraints and requiring urgent feedback I found and option to start in full-screen mode when running the exported game instance on 'itch.io'. The published game does indeed have bugged SFX, lag issues for a few users and some GUI issues. Each of which impacted our game requirements such as Operability, Interactible objects and our GUI.

5.2 Actual Solution Approach

Initially in my previous semester I opted to follow the plan outline in Figure 5.13. This original plan outline my weekly work schedule to try meet my goals and objectives during the project.

FIGURE 5.13: FYP Original Weekly Work Schedule

However due to some outdated and inapplicable requirements combined with the fact I became more accustomed to using Gamemaker Studio 2, the weekly plan changed and was updated to the schedule shown in Figure 5.14

Each weekly objective that was specified in our updated project schedule (Figure 5.14) was implemented into the project. A further analysis of results will be provided in Chapter 6.

Figure 5.14: FYP Updated Weekly Work Schedule

Chapter 6

Testing and Evaluation

This chapter will details the metrics, system testing and results involved in the project.

6.1 Metrics

In order to retrieve suitable feedback for the project I decided to create a feedback poll using Google Forms. This gave me the option to retrieve feedback in different formats to carefully analyse and visualize feedback results. My poll included a variety of different question types, each of which having various response types. A sample of some of the more salient questions are highlighted in Figure 6.1

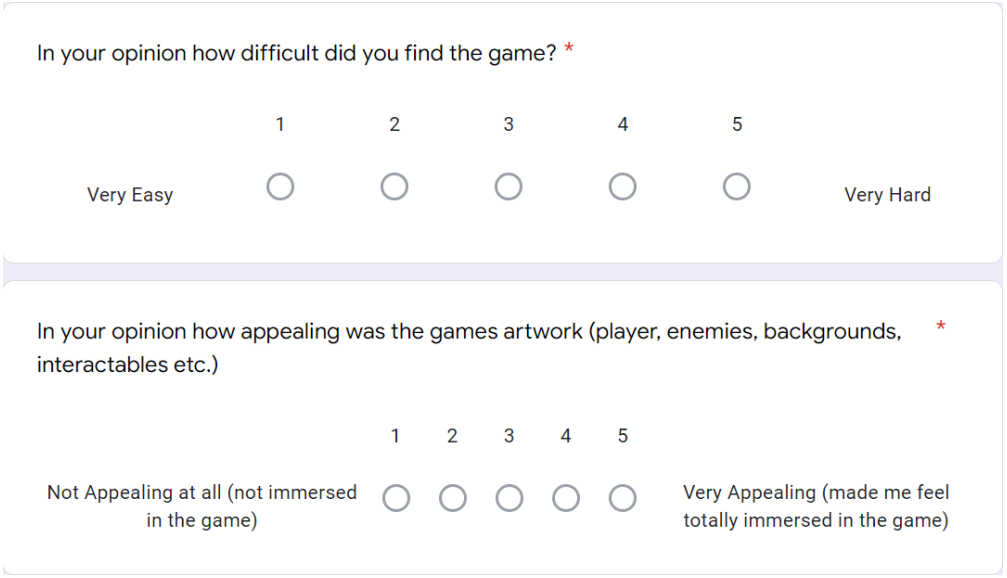

FIGURE 6.1: Google Forms Platform Game Feedback Poll

I decided to retrieve feedback from a wide range of individuals both in relation to age, gaming skill level and time spent playing games. This gave the project a larger

pool for more accurate, realistic and real world feedback. To quantify the success of our objectives the questions ranged from multiple choice questions to scaled questions. The poll would also allow us to scale how well the projects objectives, functional/non functional requirements were met. Each individual question would completely focus in on one of these requirements giving us the opportunity to identify areas of success and areas in need of improvement/revisions. An open ended section was also included towards the end of the poll to allow for individual feedback on any aspect of the game. This provided our project with numerous suggestions/considerations to further tackle our problem definition in the future.

To summarize, the feedback poll aimed to provide quantifiable answers to the following:

1. Successfully implemented control scheme?
2. Successfully implemented difficulty curve?
3. Ease of use with player movement?
4. Ease of use with gun controls?
5. Appealing use of music/SFX?
6. Appealing use of game artwork?

Overall experience of the game?

6.2 System Testing

In order for to allow individuals to provide feedback, an exported instance of the project would need to be provided. In order to achieve this some default run-time settings were altered. This involved selecting the HTML5 export platform rather than the default windows virtual machine option. Gamemaker will then allow us to open a micro web server in order for the game to be played in the browser. The snip in Figure 6.2 displays how this is set up using the Gamemaker Studio 2 IDE.

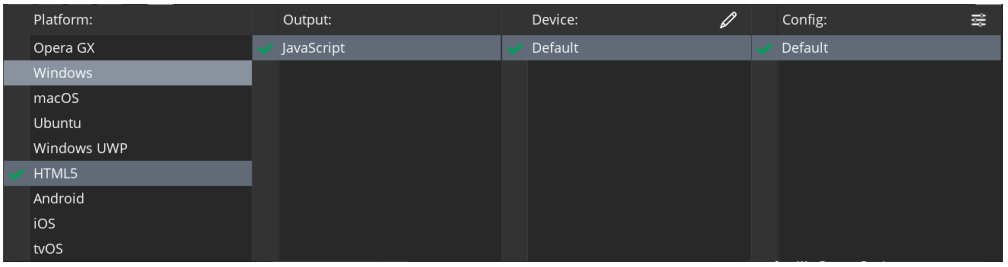

FIGURE 6.2: Gamemaker Studio 2 Run-time environment selection

The HTML5 extensions that Gamemaker provides us allows us to execute native JavaScript so that the game has the ability to become a web application. After altering the default exported instance of the project we can also define which browsers that would run

a single instance of our game. For this project, Google Chrome and Microsoft Edge were used to test the HTML5 version of our game. This allowed us to address our non functional requirements, most applicably 'Operability' and 'Availability' of the project.

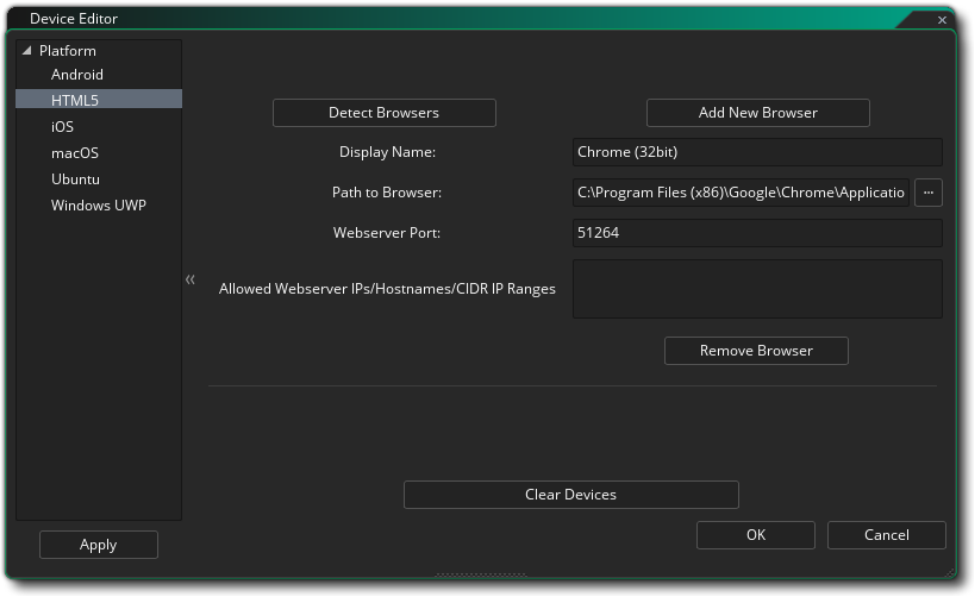

FIGURE 6.3: Gamemaker Studio 2 HTML5 Browser

To then fully export the game, Gamemaker provided a create executable feature which allowed for the project to be packaged as a zip file. This feature tied all of our project assets together and attached the necessary index.HTML file acting as a root home page for our converted JavaScript game assets. Unfortunately individuals who could provide feedback could only play the game if they had Gamemaker Studio 2 downloaded themselves. To solve this problem the popular free website known as 'itch.io' acted a quick and efficient solution to allow people to play our game. I created an account and quickly found that itch.io provided a clear dashboard to upload games and other projects. Some data was required such as inputting a unique URL for our game as well some startup options, description, access rights and general game information. From here, the project was then saved and successfully uploaded for public usage. Figure 6.4 displays the startup screen for our project found on itch.io.

6.3 Results

This section highlights the gathered feedback and results from a variety of different individuals who play tested our game. This will include our original project objectives as well as quantify the success of individual requirements for the project. Each question

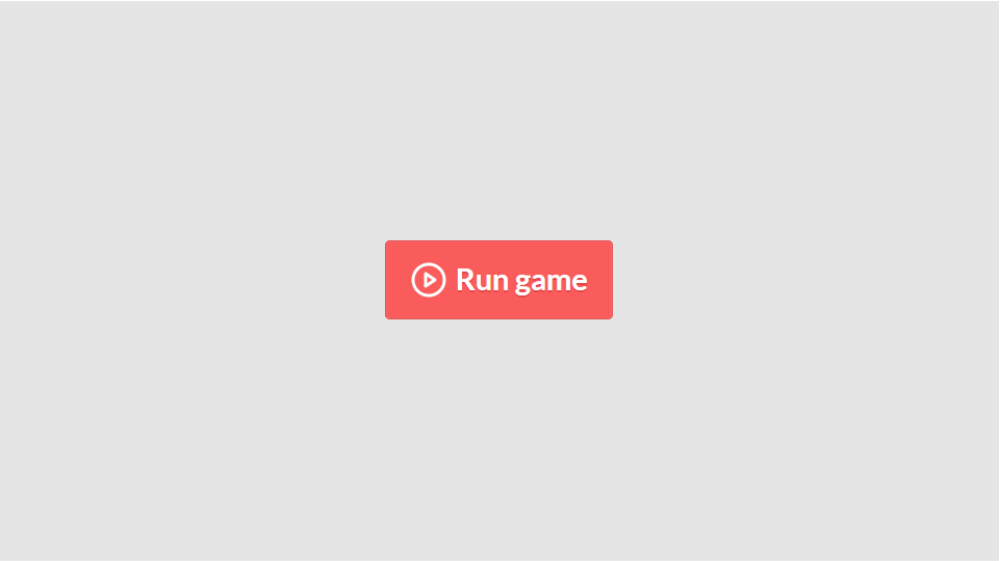

FIGURE 6.4: Itch.io default game startup screen

gathers feedback from 12 different individuals each of which vary with age range, gaming ability and likeness to games.

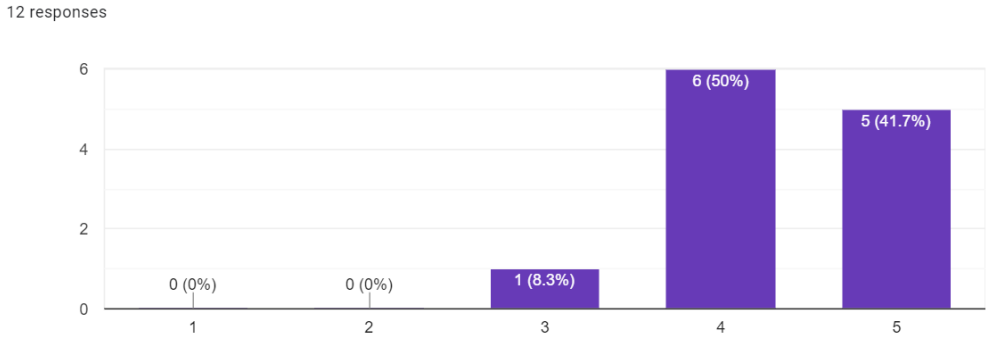

FIGURE 6.5: FYP Game Feedback Poll Artwork

Figure 6.5 represents the quality of artwork that was portrayed in the project. This ranges from 1-5 on the x axis with 1 being least favourable and 5 being most favourable. The graph displays a mode of 4 and a mean value of 4.33.

Figure 6.6 ranges from 1-5 in terms of quality of control functionality. This aspect of the project received positive feedback including a mode value of 5 and and mean value of 4.83.

Figure 6.7 ranges from 1-5 in terms of overall difficulty with 1 expressing 'very easy' and 5 expressing 'very hard'. Results were variable as most found the game slightly too easy

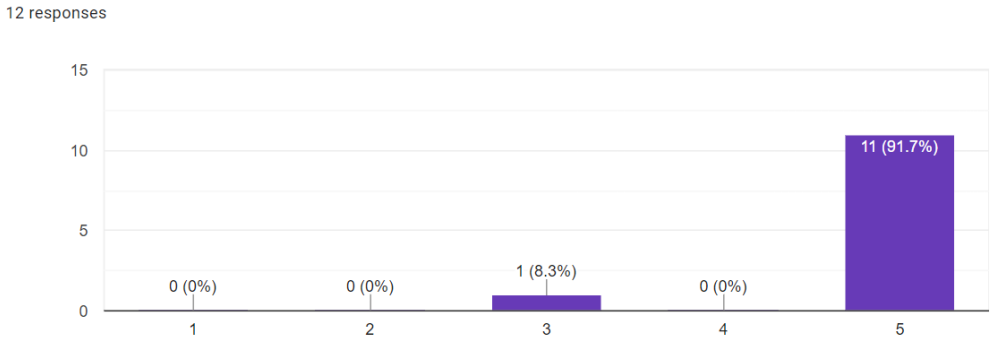

FIGURE 6.6: FYP Game Feedback Poll Control Scheme

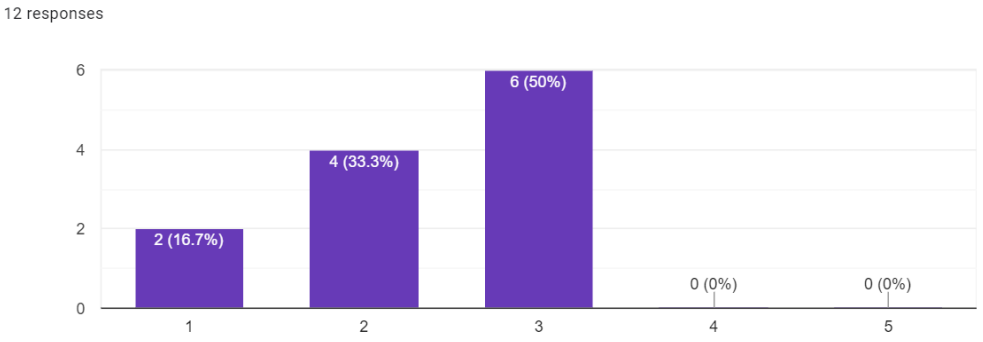

FIGURE 6.7: FYP Game Feedback Poll Level of difficulty

with a mode value of 3 and a mean value of 2.33.

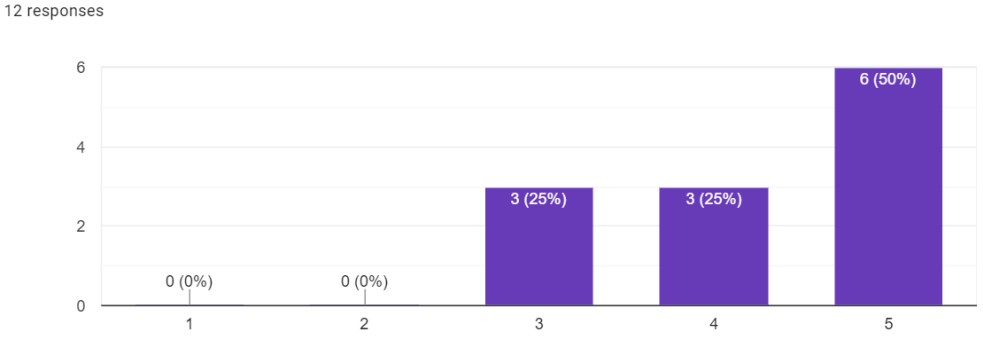

FIGURE 6.8: FYP Game Feedback Poll Gun Control

Figure 6.8 ranges from 1-5 in terms of quality of gun mechanics with 1 expressing 'clunky controls' and 5 expressing 'fluid controls'. Feedback included a mode value of 5 and mean of 4.25.

Figure 6.9 displays a pie chart with varying response into how well the player movement functioned in game. The most popular response was 'Responsive and Fluid' with a small

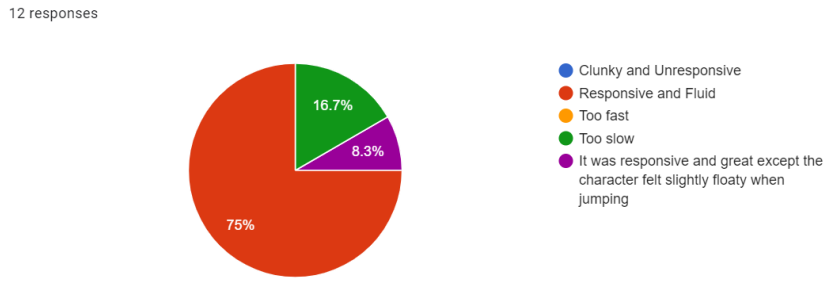

FIGURE 6.9: FYP Game Feedback Poll Player Movement

portion exclaiming movement was 'Too slow'.

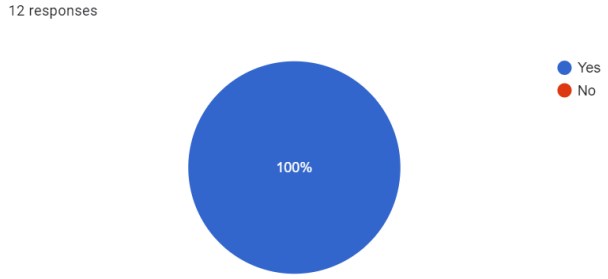

FIGURE 6.10: FYP Game Feedback Poll File Handling

Figure 6.10 Displays a simple response to if there was any issue with saving or loading the game.

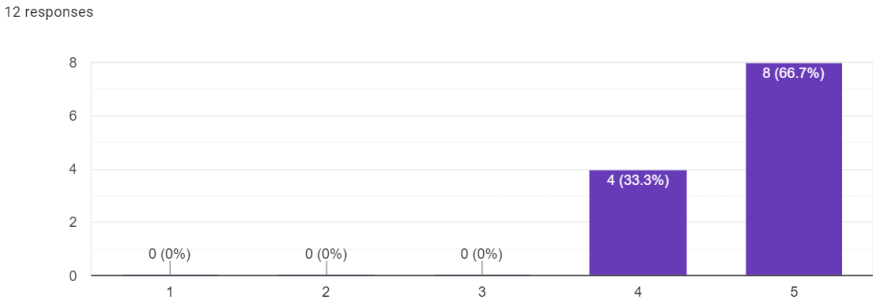

FIGURE 6.11: FYP Game Feedback Poll SFX

Figure 6.11 highlights a bar chart in which most favourable responses fall at 5 and least favourable at 1 in relation to sound effects and music within the game. Results display a mode value of 5 and a mean of 4.66.

Figure 6.12 represents a pie chart which has varying responses to convey how they felt the storytelling in the game was portrayed. The most popular answer being 'I liked the Storytelling' and least favourable answer 'The storytelling made me fully immersed in

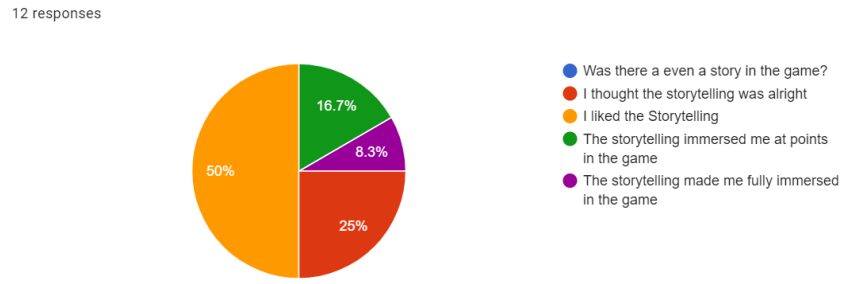

FIGURE 6.12: FYP Game Feedback Poll Storytelling

the game'.

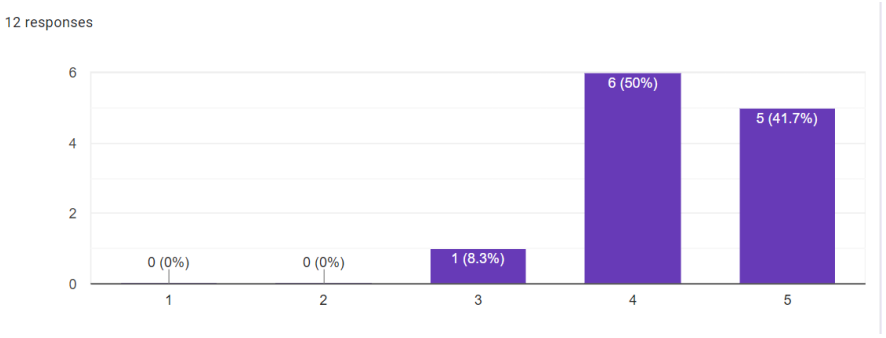

FIGURE 6.13: FYP Game Feedback Poll Overall Project Success

Figure 6.13 displays a bar graph which attempts to quantify the overall success of the project by rating it through a likeness score of 1-5. The mode value represented 4 while the mean value was 4.33.

Top Feedback Bugs/Glitches

1. Small texture glitch where water overlaps the land on collision with a signpost
2. Lag issues especially during the mine level in the game
3. Sprite would face opposite direction on occasion and appear to do a moon-walking animation

Top Feedback Improvement Suggestions

1. Implement further boss mechanics, jumping, dodge-able attack or weaponry for the boss like a rocket launcher
2. Implement more levels into the game
3. Cool down on gun fire

Chapter 7

Discussion and Conclusions

The following sections detail my conclusions and reviews while working on this project.

7.1 Solution Review

Looking into our solution for our project I believe that each objective, functional and non-functional requirement was addressed and implemented successfully in the project. There were indeed areas of our solution that could have been improved but as I defined in the risk assessment I knew that time management would be the most precarious risk.

To conclude on our results from chapter 6 we analyze each individual feature and how it conveyed a solution.

ArtWork = The artwork included in our game conveyed a simple 64x64 medievalist theme. The background artwork layered on one another creating depth in design and a layered effect throughout the levels. Based on user feedback, I believe this feature was a success.

Control Scheme = The control worked quite simple allowing for aswd controls with mouse and keyboard. The menu was also navigable by up and down arrow keys.

Difficulty Curve = The difficulty in the game was greeted by feedback users as mostly easy. I aimed for the game to be accessible for individuals of all ages while also providing a fun interactive challenge. I believe in the future there could be difficulty enhancements and further options such as difficulty modes or slightly increasing difficulty in certain portions of the game.

Gun Mechanics = The gun mechanics were generally received in quite a positive aspect. The controls in the game were slightly unique in that the gun would follow the

mouse around the screen as the player traversed the level. I believe this feature was a success but could certainly be built upon for other weapon types or even the introduction of melee weapon types.

Player Movement = The pie graph for this section exclaimed that the majority of users opted for the option of 'Responsive and fluid controls'. However there were a few votes for the 'Too slow' option which hinted that more experienced gamers wanted to play the game at a faster pace. This feature was a success but proved to be very difficult to suit the needs of as many players as possible.

File Handling = File handling worked for every user that tested our game in chapter 6. This means the functionality of this feature worked perfectly. Despite the successful feature implementation I noticed a bug after deployment where the incorrect rooms would save to memory. This has not been an issue I have been able to solve as of writing this and is still under bug fixing. Despite the glitch everything still loads and saves as defined.

Sound/Music = SFX were implemented to a satisfactory degree according the results defined in Figure 6.11. This was one of the more outstanding features that was included in our project. Despite the success of this feature it was not originally specified in our updated project schedule and therefore acted as an extra or optional feature.

Storytelling = Storytelling was met with slightly mixed reviews in our results section. This was one of the hardest features to try and implement as the knowledge required for successful storytelling would usually involve voice acting and sequence animation. Despite this we achieve a satisfactory status for this objective with the realization that this could be improved upon in the future.

7.2 Project Review

Overall, I feel that this project was a success when compared to my project objectives and requirements. I was able to meet my goals that I detailed in my project schedule with mainly optional features not completed and a couple of bugs that interrupted the project. A reinforced small bug came with object collision with other objects. This usually involved our main player colliding with the environment or other interactibles such as enemies or mines. A couple of weeks in I solved the issue of collision masking and the origin collision point of all these objects. This helped significantly for future development such as animating the player in water or colliding with objects in the boss room.

To really address the notion of immersion in video games I became obsessed with addressing even the most tiny and minute details that were in the game. When accumulating and selecting relevant game music I used the software 'Audacity' to fade in/out the start and ending of the soundtracks to add to the overall player experience. This can be seen most evidently in the main menu screen and transitioning to level 1 of the game. The music transitions from a calm serene song to an adventure starting uplifting song once 'NEW GAME' is pressed. I was quite strict with myself as to the relevance of what should be included in our game and also the ability to remove features that don't fit the game.

7.3 Conclusion

As i detailed in Chapter 1 I had previous but limited experience of Gamemaker Studio 2 in my time in secondary school, however I feel this granted me a huge advantage in feeling confident I could complete this project mainly as I knew how reliable and efficient Gamemaker Studio 2 is. One thing that hindered this project was indeed not having a game development team and only relying on myself to complete and address all aspects of the game. I found it difficult at times to manage and address an assortment of different software related tasks. Looking back on earlier development I think that my project schedules and initial research in semester 1 made it possible for the completion of this project. I also cannot stress enough how much the MDA(Mechanics, Dynamics, Aesthetics) framework helped me understand and realise how games utilise patterns to portray a sense of enjoyment and immersion for its players.

Our problem description detailed a range of different core feature to be included in our project, we conclude below if each feature was implemented and how that feature was Completed.

Interactible Main Menu - Completed using INI files(originally text files) which saved and loaded our game data using GML functions.

Control Scheme - Mostly completed using aswd controls and space-bar with mouse as core controls, gamepad controls bugged/limited in menus.

Sprite Movement - Completed using change in velocity/time formula to emulate realistic human-like movement and acceleration.

Sprite Animation - Completed using image animation tools in GML IDE.

Player Design - Completed using 64x64 assets

Enemy Design - Completed using 64x64 assets

Parallax Backgrounds - Completed using custom camera manipulation.

Non Interactible Aesthetics - Completed using 64x64 assets

Interactibles - Completed using med-kits and ammo boxes

Tile Sets - Completed using 32x32 tiles

Level Transitioning - Completed using portal asset which transitions player upon collision

Scrolling Levels - Completed using camera following player through levels

Camera Implementation - Completed

Level Difficulty Curve - Somewhat Complete as levels get harder as the game goes on, however no option to choose 'easy'/'hard' difficulty modes.

GUI - Completed displaying relevant information to the player such as remaining bullets

As noted above almost all of our project goals have been met to satisfactory degrees with only some minor inconveniences. As described in our solution review there were indeed areas in the game to improve on such as the storytelling aspect which i found extremely difficult to incorporate without the knowledge of sequencing/voice acting etc.

7.4 Future Work

For this project there are quite a number of things I would have liked to achieve or to implement in the future.

First of all I believe there is room to implement networking features in our game. Starting off it would have been great to implement a score leader-board where players can compete against one another to try and achieve the highest score. I would have also liked to have addressed in-game achievements or trophies that the game would grant the player while playing the game. This would really hit home the idea of congratulating the players and rewarding them as they play enhancing the immersive aspect. I also believe that implementing a local co-op multiplayer or hot-seat feature would benefit the game. This would bring about far more social interaction and allow players to play with friends.

I would also have liked to add machine learning algorithms to this project. This would give the game capabilities such as being able to generate random terrain for level generation which would have made the game substantially larger in scale. This could also

have been tied to enemies or interactible items such as the med-kits or ammo boxes seen throughout the game. This would have made the game dynamic as no two instances of the game would be the same.

In the games current state controller functionality is quite limited in its use. I would like to further address this feature in order to have complete controller/gamepad functionality for the project. During development of the project as mentioned in previous chapters there was issues with the button mapping such that i had to resort to using the traditional button constants rather than using the axis controls for the menu. There were also countless bugs with the gun mechanics while on controller which related to dead-zone issues.

Another feature that I would also like to work on for this project would be the option of a character selection screen. This would open up dynamic character creation/selection adding far more variety to the project. I would also have opted to implement more enemy types to be featured in the game with different stat options that builds upon the current game allowing for health, size and ledge mechanic options.

Closing the section on future work I would also like to fully redevelop the game in a 3d format in a different game engine to Gamemaker Studio 2. I would opt to try recreate this game using Unreal Engine or Unity which would grant far more versatility and game functionality options when developing the game.

Bibliography

[1] R. Hunicke, M. LeBlanc, and R. Zubek, "Mda: A formal approach to game design and game research," in *Proceedings of the AAAI Workshop on Challenges in Game AI*, vol. 4, no. 1. San Jose, CA, 2004, p. 1722.

[2] E. Adams and J. Dormans, *Game mechanics: advanced game design.* New Riders, 2012.

[3] M. LeBlanc, "Tools for creating dramatic game dynamics," *The game design reader: A rules of play anthology*, pp. 438–459, 2006.

[4] S. Niedenthal, "What we talk about when we talk about game aesthetics," in *Digital Games Research Association (DiGRA), London, UK (2009).* DiGRA Online Library, 2009.

[5] I. Millington, *Game physics engine development.* CRC Press, 2007.

[6] J. Gregory, *Game engine architecture.* crc Press, 2018.

[7] D. Arsenault, "Video game genre, evolution and innovation," *Eludamos. Journal for computer game culture*, vol. 3, no. 2, pp. 149–176, 2009.

[8] D. Clearwater, "What defines video game genre? thinking about genre study after the great divide," *Loading...*, vol. 5, no. 8, 2011.

[9] A. Aska, *Introduction to the study of video game music.* Lulu. com, 2017.

[10] D. M. Young, "Adaptive game music: the evolution and future of dynamic music systems in video games," Ph.D. dissertation, Ohio University, 2012.

[11] Aster. (2021) Why the ratchet & clank soundtrack is a perfect fit.

[12] T. Summers, *Understanding video game music.* Cambridge University Press, 2016.

[13] K. Compton and M. Mateas, "Procedural level design for platform games." in *AIIDE*, 2006, pp. 109–111.

[14] M. Sharif, A. Zafar, and U. Muhammad, "Design patterns and general video game level generation," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 9, pp. 393–398, 2017.

[15] A. Zafar, H. Mujtaba, and M. O. Beg, "Search-based procedural content generation for gvg-lg," *Applied Soft Computing*, vol. 86, p. 105909, 2020.

[16] K. Isbister, *Better game characters by design: A psychological approach.* Elsevier/-Morgan Kaufmann, 2006.

[17] B. L. Mitchell, *Game design essentials.* John Wiley & Sons, 2012.

[18] P. Lankoski, "Character design fundamentals for role-playing games," *Beyond Role and Play*, pp. 139–148, 2004.

[19] J. Roettl and R. Terlutter, "The same video game in 2d, 3d or virtual reality–how does technology impact game evaluation and brand placements?" *PloS one*, vol. 13, no. 7, p. e0200724, 2018.

[20] M. Masuch and N. Röber, "Game graphics beyond realism: Then, now and tomorrow," in *Level UP: digital games research conference. DIGRA, Faculty of Arts, University of Utrecht*, 2004.

[21] B. Palmer, "Game review: Sid meier's civilization," *not specified*, 2001.

[22] A. H. Cummings, "The evolution of game controllers and control schemes and their effect on their games," in *The 17th annual university of southampton multimedia systems conference*, vol. 21. Citeseer, 2007.

[23] G. Dahl and M. Kraus, "Measuring how game feel is influenced by the player avatar's acceleration and deceleration: using a 2d platformer to describe players' perception of controls in videogames," in *Proceedings of the 19th International Academic Mindtrek Conference*, 2015, pp. 41–46.

[24] S. Swink, *Game feel: a game designer's guide to virtual sensation.* CRC Press, 2008.

[25] J. T. Alexander, J. Sear, and A. Oikonomou, "An investigation of the effects of game difficulty on player enjoyment," *Entertainment computing*, vol. 4, no. 1, pp. 53–62, 2013.

[26] M. C. Jimenez and T. P. Buijtenweg, "Atum: applying multi-layer game design and environmental storytelling," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 2013, pp. 2623–2626.

[27] H. Wei, J. Bizzocchi, and T. Calvert, "Time and space in digital game storytelling," *International Journal of Computer Games Technology*, vol. 2010, 2010.

[28] C. Fernández-Vara, "Game spaces speak volumes: Indexical storytelling," *Proceedings of the 2011 DiGRA International Conference: Think Design Play*, 2011.

[29] A. Thorn, *Cross Platform Game Development.* Jones & Bartlett Learning, 2008.

[30] I. v. Hilvoorde and N. Pot, "Embodiment and fundamental motor skills in esports," *Sport, Ethics and Philosophy*, vol. 10, no. 1, pp. 14–27, 2016.

[31] L. Y. Xiao, "People's republic of china legal update: The notice on further strictly regulating and effectively preventing online video gaming addiction in minors (published august 30, 2021, effective september 1, 2021)," *Gaming Law Review*, 2021.

[32] S. Liszio and M. Masuch, "Designing shared virtual reality gaming experiences in local multi-platform games," in *International Conference on Entertainment Computing.* Springer, 2016, pp. 235–240.

[33] "Spelunky," 2008.

[34] T. Thompson, "" with fate guiding my every move": The challenge of spelunky." in *FDG*, 2015.

[35] A. Hall, "Spelunky's sublime music keeps me addicted to dying," *Morning Music*, 2020. [Online]. Available: https://kotaku.com/morning-music-spelunky-s-sublime-music-keeps-me-addict-1844783362

[36] D. Yu, *Spelunky: Boss Fight Books# 11.* Boss Fight Books, 2016, vol. 11.

Appendix A

Code Snippets

Put appendix material in this section e.g. code snippets

USE THE APPENDICES

Appendix B

Wireframe Models