Run N Jump

CE301 Capstone Project Final Report

By Bartosz Markiewicz

Supervised by Dr. Doctor, Faiyaz and Voudouris, Alexandros A

1704256

# Acknowledgements

Throughout the writing of this report I had received support and assistance from numerous people whom I’d like to thank. Firstly, I would like to thank my project supervisor Dr. Doctor, Faiyaz who was invaluable in providing me advice and challenging me to make the project as best as I could. Allowing me to ask many questions and providing me clear answers when I was feeling uncertain on how to proceed.

Secondly, I’d like to thank my CE301 course supervisors; Dr Vishanathan Mohan and Professor Anthony Vickers who provided a great deal of insight and information into the creation of a final year project, their support throughout the year was vital. Their lectures were very useful in explaining everything.

Furthermore, I would also like to thank my parents who gave me a sympathetic ear and being there for me in times of struggle. Them answering my calls in the morning after an all-nighter of work was always reassuring. Lastly, I’d like to thank my fellow Computer Science students at the University of Essex, school of Computer Science and Electronic Engineering department who gave me encouragement and a time of relaxation in-between the work on this project. A distraction was vital to rest my mind between times of work.

# Abstract

The Arcade has been called the grandfather of modern video-gaming. The influence of arcade games can be seen even on modern games. The video-game market has been growing rapidly and is projected to continue growing. This had encouraged me to recreate the classic side-scrolling Super Mario Bros experience with my own twist on the formula.

My project goals have been to create a Mario game with light RPG-elements such as friendly NPCs, a text-based story narrative and a basic inventory system. My focus has been on having a working game with collisions, enemy AI agents and other dynamic world features such as unlockable doors. I aim to have it be playable on Windows and Android platforms. For Android I intend to have the game published on the Android Google app store in the future. To create this game, I had utilised mainly the LibGDX Java game development framework based on OpenGL using the Android Studio IDE. I had used TMX tiled maps for the creation of my game levels. Furthermore, I intend to add an online high-score system for the game to allow the players to compare their score to others. Additionally, I had added a wide range of animations and SFX into the game to give the game a more professional feel to it.

Table of Contents

[Acknowledgements 2](#_Toc70669149)

[Abstract 3](#_Toc70669150)

[Introduction 5](#_Toc70669151)

[Chapter 1 Literature Review 5](#_Toc70669152)

[1.1 Context 5](#_Toc70669153)

[1.2 Problem Definition 5](#_Toc70669154)

[1.3 Serious Games 6](#_Toc70669155)

[1.4 Design 6](#_Toc70669156)

[Chapter 2 Project Description 8](#_Toc70669158)

[2.1 Aims and Objectives 8](#_Toc70669159)

[2.2 Scope 10](#_Toc70669161)

[Chapter 3 Project Methodology 13](#_Toc70669164)

[3.1 What? 13](#_Toc70669165)

[3.2 Why? 14](#_Toc70669166)

[Chapter 4 Project Requirements 14](#_Toc70669167)

[4.1 User Group 14](#_Toc70669168)

[4.2 Functional Requirements 14](#_Toc70669169)

[4.3 Other Requirements 19](#_Toc70669170)

[4.4 Use Case Diagram 19](#_Toc70669171)

[Chapter 5: Design and Implementation 20](#_Toc70669172)

[5.1. Implementation Plan 20](#_Toc70669173)

[5.2 Package Diagram 20](#_Toc70669174)

[5.3 Class Diagrams 22](#_Toc70669175)

[5.4 Tools Utilised 26](#_Toc70669180)

[5.5 Game Description 27](#_Toc70669181)

[Chapter 6: Testing 37](#_Toc70669185)

[6.1 Debugging 37](#_Toc70669186)

[6.2 Devices Tested On 38](#_Toc70669187)

[6.3 Known Bugs 38](#_Toc70669188)

[6.4 Testing Strategy 38](#_Toc70669189)

[Chapter 7: Summary of Technical Documentation 39](#_Toc70669190)

[7.1 Outline of the Working System 39](#_Toc70669191)

[Chapter 8: Project Management 40](#_Toc70669192)

[8.1 Work Done 40](#_Toc70669193)

[8.2 Cumulative Flow Diagram 41](#_Toc70669194)

[8.3 Weekly Meetings 41](#_Toc70669195)

[8.4 Legal Issues 42](#_Toc70669196)

[8.5 Risk Management 42](#_Toc70669197)

[Conclusions 44](#_Toc70669198)

[Summary 44](#_Toc70669199)

[Critical Evaluation 45](#_Toc70669200)

[References 47](#_Toc70669201)

# Introduction

The objective of the product is to have a fun and engaging 2D side-scrolling platforming game for people who enjoy casual videogames. A 2D side-scrolling platformer videogame is a game where the user sees the world through an orthographic projection, this can be a side-view, the front, or the top of the imagined 3D world. In my case the projection is from the side – the user goes from left to right with gravity pulling the player down on the x, y coordinate system, down being the ground, up being the sky, this is typical of a 2D platformer. I chose to re-create Super Mario because I realized the classic experience was lacking RPG elements which could make it better.

# Chapter 1 Literature Review

## 1.1 Context

In the early 1980s the video game industry was growing steadily. There was a ‘golden age of arcade video games’ from 1978 to 1983. With popular releases such as Space Invaders (1978), Pac-Man (1980) and Donkey Kong (1981) [6], following this economic boom, the video games industry crashed in 1983 with many game developers going bust. [1] Many saw it as a failure of the industry but following the bust the industry began to recover globally with classic hits such as Tetris (1984), Super Mario Bros (1985) and Blasteroids (1987). Later in the 90s there was a boom of the fighting genre of arcade games.[7] The release of the Mortal Kombat and Street Fighter games shaped the overall arcade game market for years to come. Those big Arcade games of the 80s and 90s have inspired countless modern re-iterations or similar games. Currently the video game industry is the fastest growing software industry segment. [3. p25]

## 1.2 Problem Definition

Despite the size of the video-gaming industry there isn’t really a major notable modern alternative to the Super Mario franchise which has been dominating the 2D platforming genre since its conception. Super Mario Bros U for the Wii U released in 2012 sold 5.8 million copies despite the console itself not being very successful, showing that the Super Mario franchise is still alive and well after 40 years.[8] Furthermore Super Mario games are still being made, some with innovative 3D graphics and gameplay such as Super Mario 3D World but others with the classic 2D side-scrolling formula intact such as Super Mario Maker 2. There hasn’t been much innovation in the franchise since the original Super Mario Bros, aside from the move towards 3-dimensional gameplay. The 2D Super Mario Maker 2 plays the same as Super Mario Bros. There are not many new interesting features, increased complexity, or fundamental changes to the formula. Aside from more levels (allowing user creation of levels), most of the enemies, world features and gameplay elements haven’t changed as much as one would expect in over 40 years, aside from being modernized. Some people would say, ‘if it isn’t broken don’t fix it’, but it’s clear there is room for innovation in the genre. I personally found things missing and I am aiming to add these things into my own Mario-inspired game. The rest of this report will specify what these things are.

## 1.3 Serious Games

Arcade games are not just used for fun. The development of games has often been used by new software engineers to teach themselves computer science principles they can use in all spheres of software development. But it is not just the process of creation of a video game that can have an educational benefit.

Playing a video game can be very educational and useful, most arcade games can be in some way adapted for an educational purpose. Even a simple game like Tetris can solve an educational purpose, at MIT researchers created a web-based vocabulary-drill version of Tetris. It works using voice recognition in the form of Web Accessible Multimodal Interface software [4. p4]. The game begins like classic Tetris with a piece appearing at the top of the game board. The twist is that also an image appears every time, if it’s the first time, the image appears with the corresponding word. If the image is appearing for a subsequent time it appears without the word and this is when the player has to say the word associated with the image out loud to unlock block rotation. [4. p3] Through this the player learns new words and their meaning.

There are limitations to the ability of games to teach, and certainly adding learning to a game can often lead to lesser overall enjoyment of the game. [4. p5] This necessitates the developer to often choose between either high learning effectiveness or fun when choosing how to implement features.

Games with a ‘higher’ purpose besides entertainment are often called serious games. This can be a game with an educational/learning aspect or a game promoting health/therapy. Currently the biggest challenge in developing serious games is the difficulty in assessing whether the game possesses actual learning value/benefit, which is why there is a lot of research going into game related learning analytics. [5. p23]

## 1.4 Design

When designing the product, I had conducted research into game design and have learnt that games are meant to arouse meaningful immersive experiences, furthermore a game consists of various elements such as mechanics, story, aesthetics and technology - these are equally important. Each player’s experience is totally unique. [1. p1] There is a subset of serious games which can be educational, educational games have to be designed properly to incorporate engagement that integrates with educational effectiveness - the challenge for the game designer is in finding a balance between game-play and learning objectives. The goal of a designer is to balance the five elements - flow, immersion, presence, arousal, and engagement. [1. p2] If a designer wants to teach something, the learning should impose a cognitive load. If the learning objectives are disconnected from the gameplay the game may fail to produce educationally effective experiences. [1. p4] Additionally the difficulty of the game should be increased when the player is bored. [1. p7] Simply put player engagement is important, the experience of the player needs to be rewarding, non-player characters need to resemble real players as much as possible - since gameplay alongside real players had much higher user engagement. Lastly the game should give feedback to the user to show how he is performing, especially if the game has a learning component. [1]

Games can be fun and educational due to tapping into the motivational drivers of human behaviour - positive/negative reinforcement and emotions. The principles central to gamification and games in general are the MDE framework - mechanics, dynamics, and emotions. The types of people involved are - players, designers, spectators, and observers. Mechanics are the decisions that designers make to specify the goals and the boundaries of the situation to be gamified. There are three different types of mechanics, setup, rule, and progression mechanics. Which are tremendously important for games, setup mechanics shape the environment of the experience. Rule mechanics shape the concept or the goal of the experience to be pursued. Whereas progression mechanics dictate the reinforcements present in the experience - this increases the likelihood that certain behaviours will be repeated in the future. [2]

### 1.4.1 Tools Considered

The goal is for a Windows and Android release, I am mainly considering tools which allow easy cross-platform development. The small touch screen controls of Android and iOS favours the simpler arcade kind of games [9] whilst windows computer games with keyboard, mouse and possible periphery control gadgets led gamers owning those devices to be more inclined towards more complicated and competitive games.

* LibGdx - A flexible cross-platform 2D video game development framework based on OpenGL 2.0 and the Java programming language with some C and C++ components for performance intensive code It allows a project to target Windows, Linux, Mac, Android and iOS. Supports many 3rd party tools. Has low level OpenGL helper features such as meshes, textures, framebuffer objects etc. Additionally, contains high-level 2D APIs relevant to my project such as Orthographic camera, sprite batching and caching, 2D particle system and a TMX tile map support.[10]
* SFML - Simple and Fast Multimedia Library, it is a simple interface for the development of games and multimedia for multiple operating systems, Windows, Linux and macOS. It is a C++ API but can be utilised from C#, Java and Python among many others. It is an exclusively 2D graphics-based library. It is more of a cross platform layer abstraction API rather than a straight game development framework. [12]
* Pygame - Is a Python game programming library used for the development of 2D games. It is based on SDL. It allows for easy multi-threading, utilises optimized C and assembly code for core functions, which is much faster than Python code. It prides itself on being easy to use due to requiring a small amount of code while giving the user a lot of control over their game. Many indie games have been made with it. It is very modular so it’s possible to utilise different libraries for specific aspects of the game. [13]
* Java- Java has Swing and AWT which can be used for 2D graphic work, this makes the language able to be utilised for making arcade games without any external libraries. It is great for Windows development and doesn’t require any installation porting the game to Android is trivial. Porting to other platforms is also possible due to the JVM but harder.
* rise.global - Allows for the creation of dynamic real-time leader boards from a CSV file (Comma Separated Values) or excel spreadsheets. Useful for adding a player versus player (in terms of score) competitive element or a goal to an arcade game. It has an extensive list of features such as email updates, score card, scheduling of score collection, etc. [11]
* Unity Game Engine – the unity game engine allows for the easy creation of videogames through the use of a graphic user interface. It allows for easy cross-platform development.

# Chapter 2 Project Description

## 2.1 Aims and Objectives

The main goal was to create a 2D side-scrolling game for Windows and Android platforms which gives the user a fun and engaging experience. I had chosen to name the game ‘Run n Jump’. The project statement was to re-create a classic arcade game. After careful research I set out to recreate the Super Mario Bro. I wanted to try my own hand at re-creating the classic experience with my own twist on the formula – while keeping what made the classic experience fun and engaging intact.

I chose the Android platform due to my research showing that the Android market would be most appropriate for this sort of game alongside the iOS market. If given enough time and resources I would port the game to iOS, but time constraints only allow me to focus on two platforms at maximum. I had chosen to make the game for Windows too due to the ease of testing and development on Windows.

In the game the player would play a character like in Super Mario, he would move this character using the Android on-screen touch controls or the keyboard if on PC. The player will be able to set his own player character’s name when starting the game for the first time on the device. This will be an arcade game at its heart it will compete on the Android and Windows markets with other arcade games. Another aim is to have it released on itch.io – an indie/arcade game distribution site for Windows and on the Google play store for Android in the future.

The twist on the formula is the addition of RPG-elements such as quests, NPC dialogue, non-linear level design and more player choice in the way he has a choice when to use a power-up, which the player can store in his inventory and which route to take through the level. The survival game mode also adds unlimited replay-ability to the game which is lacking in Super Mario Bros.

The game upon launching will show the player the main menu screen, this is where the player can pick the game-mode, after choosing the campaign game mode the player will pick the level to play. Initially only the first level will be unlocked. When the player starts the level the gameplay loop begins. The game world is loaded, and the player character is placed with the camera following the player character. The player must avoid any dangerous hazards – spikes, enemies, etc. The main objective of a campaign mode level is to reach the end of the level victory flag, secondary objective is to reach highest score you can. The game will have an aspect of exploration with the idea that the player must find the end flag, each level will have multiple non-linear routes the player can take and optional content. On the way the play can interact with collectibles, use power-ups, talk to NPCs, reach checkpoints to respawn at in case of player character dying. The player can die and respawn at a checkpoint if he has lives left. The player can receive quests or tasks from NPCs he encounters which can give him rewards and things to do. The quests can be something like finding another NPC by interacting with him and returning for a reward, killing certain number of enemies, or finding something.

The game will have a storyline aspect in the campaign game mode, the story will be conveyed to the player through the NPC dialogue system. The NPCs will talk to the player and tell him things. There will be a trading system where the player can purchase power-ups, other items such as keys (for unlocking passageways) or quest items from vendor NPCs.

The second game mode, the survival game mode features an infinite procedurally generated world with a more arcade-like gameplay. The goal in this game mode will be for the player to survive and run as far as he can to the right, gathering as much score as possible in the process. This game mode is called survival because the player is chased by a wall of fire of an erupting volcano, the idea is that the player needs to keep moving to the right of the screen otherwise he will die, any obstacles that delay the player can result in the wall of fire catching up and killing him. As the player runs the level is generated in front of him to provide a potentially endless experience.

### 2.1.1 Summary of Objectives

* Working on most devices of the proposed user-base
  + Android version 18
  + Windows 10
  + Device with at least 1gb of RAM memory
* Two Game-modes
  + ‘story’ based campaign mode of 3 levels which need to be done sequentially with saving of progress
  + ‘survival’ mode which consists of an infinite randomly generated level ‘endless run’, with the goal being to acquire as much score as possible before inevitably losing.
* Power Ups which can be stored in a player inventory and used whenever needed, with indication of how many of each the player has in his possession.
  + Super speed
  + Lower gravity
  + Rock throwing – another way to kill enemies.
  + Invincibility
  + Ghost-walk – ability to pass through certain walls.
* NPCs placed on the campaign levels which can
  + Engage in dialogue, the player can approach an NPC and interact with him.
    - Dialogue window appears and the player can read it in the style of a Visual Novel type game.
  + Give the player scripted quests and rewards for their completion
  + Trade with the player, the player can buy quest items, a key, power up, a life etc.
* Tile-based levels
* Enemy Monsters with differing AI
  + Hedgehog – patrols from one point to another, kills the player on contact
  + Bandit – stays in one place, upon spotting the player he chases, jumping when necessary to reach the player, if the player is in range, he strikes him with his sword. He kills the player only using his sword attack, which has a short cooldown. Can be killed by the player jumping on his head.
  + Rock-thrower – similar AI to the bandit but has an additional ranged attack in the form of throwing a rock at the player if the player is in range. If the player gets close, he uses his sword to kill the player. Can be killed by the player jumping on his head.
* Collectibles
  + Power Ups
  + Score - Coins & Stars
  + Hearts (Lives)
  + Quest items
* High score saving and leader board system
  + The player will be able to see his highest scores.
  + If time constraints allow this will be done through an online database system.
* Animations
  + All the sprites for moving elements will be animated.
  + Enemies will have attack animations which need to play to damage the player.
* Music and SFX
  + The game will have background music and SFX where appropriate.

## 2.2 Scope

### 2.2.1 The Platforms

The product is intended for Windows and Android release; the gameplay will be pretty similar on both platforms, but due to the nature of the two platforms it will differ in some respects. One difference will be that the Windows version will allow a higher resolution and window size. Although despite this I will restrict the player view distance to the same pixel width & height on both platforms for easier gameplay balance. A major difference will be that the control schemes will differ significantly, due to Android being a touch-screen device and Windows being mouse and keyboard based. This necessitates difference in control, I could not reasonably expect people to use a keyboard with their Android device or a touch screen with their Windows PC. Lastly the game likely will be harder to play on Android because of the touch screen and typically smaller screen, due to this I will keep the leader-boards separate for each platform. The game will use basic graphics so the higher power of a gaming desktop PC will not grant the user significant performance improvements or higher graphical fidelity.

The focus will be on Windows 10 as my main platform on which I’m focusing my development efforts on. It is the biggest desktop PC operating system on the market currently I developed and tested on Windows 10 before Android due to the platform allowing for quicker loading and build times during development. In addition, the integrated development environment debugging tools run quicker than on Android. The player will play the game using their keyboard and mouse. The keyboard will be used for moving the player character and interacting with the game such as using power-ups or talking to NPCs. The mouse will be used for navigating the menu screens. Due to my granting of priority to the Windows version it is likely the user experience will be better on this platform due to greater amount of focus given.

Android will by my secondary platform. Android is a mobile platform for mainly smart-phone and tablet devices with a touch screen. The user will control the character primarily using an on-screen touch ‘joystick’ and an interact and jump on screen buttons. There will also be a use power-up button and a button for switching the selected power-up. The API level targeted will be 21 or android platform version 5.0 and above, which will make my game compatible with over 94.1% Android devices which use that version or newer. Although this could change during development if the newer Android API features showed themselves to be worth the sacrifice in market-share. It is also possible the game will work on older Android devices but that will not be tested.

### 2.2.2 Deliverables

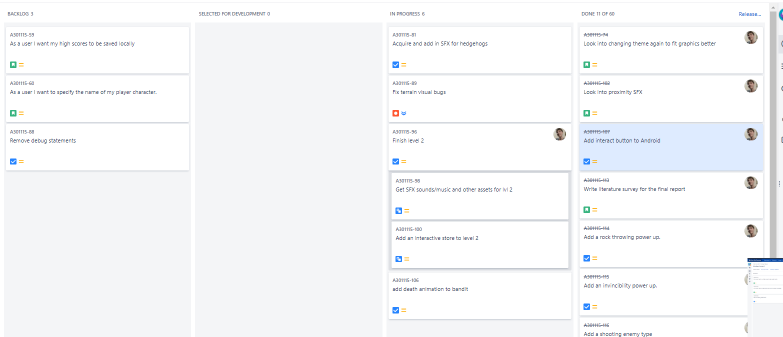
Below are the deliverables for the release of the game in week 30, with the estimated workhours required to implement that feature.

* A working game on both Android and Windows. – 20 hours
* Menu system – 10 hours
* Music and Sound FX – 10 hours
* Camera view and screen systems – 5 hours
* Player character control systems – 30 hours
  + Android on-screen joystick and button controls
  + Windows key-bindings.
* Tile-based map functionality – 10 hours
* Animation system – 10 hours
* Graphics – 20 hours
* Save data functionality – 5 hours
* Campaign mode – 60 hours
* Survival game mode – 40 hours
* Scoring and leader-board system – 20 hours
* At least 3 power up types – 10 hours
* At least 3 type of enemies with differing AI – 30 hours
* Inventory system – 10 hours
* NPC system – 30 hours
  + Dialogue system
  + Trading/Shop system
  + Quests
* Basic storyline – 5 hours

Estimated time required to implement the product altogether – 325 hours. All in all, essentially the goal is to have a complete albeit short videogame by week 30. The intended playtime of the campaign mode should be 1 to 2 hours for a player who plays the game for the first time with no prior knowledge aside from reading a basic guide. The way the game will be built is that a player who already played it will be able to complete it much quicker again due to no need for exploration.

# Chapter 3 Project Methodology

## 3.1 What?

To organise the work, the Agile Software Development methodology with a Kanban board was used. Jira was used for organisation. Jira is a digital tool which allows the user to have a digital project board accessible from anywhere and saves product developers from using a physical whiteboard. It allows to create issues into a backlog on the project board, release software versions and specify what is currently being worked on, this allows to see the current project progress. The agile methodology allows a product development team to focus on continuous delivery of software, it is very flexible in the way it allows to change the project requirements as development occurs when the client or developer acquire new information.

Typically, agile project teams have a weekly or daily scrum meeting to talk about progress and how to proceed. But since this is a solo project, a scrum meeting with just myself is impossible, instead I had a weekly meeting with my project supervisor which served a similar purpose. For each weekly meeting with my supervisor I had prepared a small rough word document which noted what I had worked on the previous week and what I am planning on doing the next week. I would read that document to my supervisor and receive his feedback.

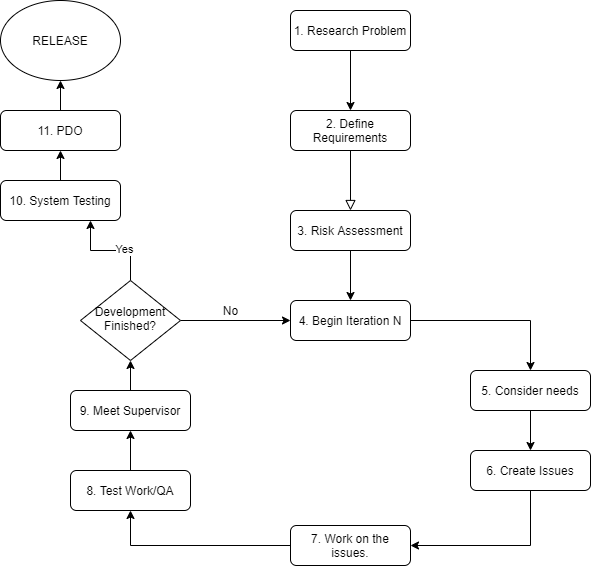


Figure 1 My Project Methodology flow diagram

The above flow diagram in figure 1 visualises the methodology utilised for this project. The project began with research concerning the subject area of arcade gaming. Based on this an idea on what the project should be emerged, so the next step was to define the objectives and requirements. Next, it was crucial to conduct a risk assessment, to mitigate any possible risks with the project. Then the actual iterative development process was to begin. Each iteration would be a weekly sprint which will begin by considering my priorities – what needs to be worked on this week. Issues were created issues for each week – sometimes other issues which come to mind also will be created which will be worked on the following weeks. The next step would be to work on the highest priority issues, afterwards the work done would be tested. Afterwards the week should be done, and I should have my weekly meeting with my supervisor. At these meetings I informed my supervisor what work I had done and what I intend to do next, acquiring his feedback on what I should work on or change in the project. If the development isn’t finished, the process would be repeated while taking the supervisor’s feedback into consideration. If the development is finished, or the final deadline for the project release is close, the next step was to proceed towards system testing. Afterwards the demonstration and presentation of the finished product through a presentation, demonstration, and the oral examination (PDO).

## 3.2 Why?

In the end I chose Agile, but other methods had also been considered such as the Waterfall methodology which is a classic development methodology which is heavily requirements focused - you need to have a really good idea of what the project requirement are before proceeding with the project. I found this difficult, my project heavily evolved over the course of its development, I knew my requirements could change so I did not see Waterfall as appropriate. It is a sequential methodology divided in stages which come one after another. It is pretty high risk and inflexible, but it has its advantages, it’s easy to understand and use, its structure keeps you on track. The focus on requirements leads to you having extensive documentation which can allow joining a project mid-development easier. In the end I had chosen Agile which allowed me more freedom for changes and adaptation. Weekly meetings with my supervisor allowed for my project to evolve in an agile fashion.

# Chapter 4 Project Requirements

## 4.1 User Group

The game will be geared towards people with a mobile Android smartphone device or a Windows 10 computer. It is designed for a single-player experience for people at least 12 years old (due to violence and death occurring in the game). The game contains a small social aspect in the way online high scores grant the user the ability to compare his scores to the scores of his friends.

## 4.2 Functional Requirements

Actors

* Player
  + This actor is the actual expected user of the application, the player of the game.
* AI agent
  + This actor represents an artificial intelligence which will control an enemy or friendly NPC character in the game.
* Server
  + This actor represents the server database which stores the global high score data of all players.
* Name: Start Campaign Mode
* Summary: The player navigates the menu screens chooses a level and starts playing it.
* Actors: Player
* Preconditions: The player has installed the game.
* Sequence:
  + 1. The player starts the game.
  + 2. The player clicks ‘Start’ option
  + 3. The player chooses the level.
* Exceptions:
  + 3. The level the player chose has not been unlocked, a pop up appears telling the player he needs to complete the previous level first.
* Post conditions:
  + The player can now play the level he had chosen.
* Priority: High
* ID: FR1
* Name: See high scores
* Summary: The player navigates the menu and can see his own highest score in Survival mode or a campaign level.
* Actors: Player, Server
* Preconditions: The player is in game at the main menu.
* Sequence:
  + 1. The player starts the game.
  + 2. The player clicks ‘High-Scores’ option
  + 3. The server sends the high scores to the player.
  + 4. The player clicks ‘Survival’ or one of the level high scores.
  + 5. The player clicks on the ‘show my position’ button.
* Exceptions:
  + 3. The server is offline so it cannot send the player the data. It will show the last downloaded high scores. If there are any.
  + 3. The player has no access to the internet; the user will be informed that the high-score data cannot be retrieved from the server until an internet connection is established.
  + 5. The player hasn’t completed a level or played the survival mode so there will not be a highest score for the player, the button will do nothing.
* Post conditions:
  + The player can now see his own highest score achieved on the level/mode played.
* Priority: Low
* ID: FR2
* Name: Start Survival Mode
* Summary: The player clicks the survival button in the main menu.
* Actors: Player
* Preconditions: The player has installed the game.
* Sequence:
  + 1. The player starts the game.
  + 2. The player clicks ‘Survival’ option
* Exceptions:
  + None
* Post conditions:
  + The player can now play survival.
* Priority: Medium
* ID: FR3
* Name: Control Player Character
* Summary: The player can run, jump, and interact with certain game elements such as power-ups and enemies.
* Actors: Player, AI agent
* Preconditions: The game has been started after either choosing a level or survival mode. (FR3 or FR1)
* Sequence:
  + 1. The player uses the joystick(android) or arrow keys to move the player character towards the right of the screen.
  + 2. The player sees an obstacle then he uses the jump key to avoid it.
  + 3. The player sees a power-up. He interacts with it and acquires a power-up.
  + 4. The player encounters an enemy agent; he kills it by jumping on it.
  + 5. The player uses a power-up using its numerical key binding (Windows) or tapping its icon on Android.
  + 6. The player notices an NPC; he interacts with it to start a dialogue.
* Exceptions:
  + 2. The player hits the obstacle and the player loses a life, respawning at a checkpoint, or the player has no lives left and a game over screen appears with an option to play again.
  + 3. The player fails the mini-game and doesn’t acquire the power up.
  + 4. The player fails to jump on it and instead dies from a slash attack used by the enemy causing the player character to die.
* Postconditions: The player can complete the level.
* Priority: High
* ID: FR4
* Name: AI chase player behaviour.
* Summary: The AI agent moves in response to seeing the player.
* Actors: AI Agent, Player
* Preconditions: The game has started and a player came close to the AI agent.
* Sequence:
  + 1. The player character approaches close enough for the AI agent to become active.
  + 2. AI agent determines which type of character it is for.
  + 3. The player approaches the AI agent close enough.
  + 4. The AI agent faces the player and moves towards him.
  + 5. The AI agent uses a slash attack on the player when close enough.
* 3. Exceptions:
  + 1. The player never moves close enough to the AI, so the NPC never becomes active.
  + 3. The player never moves within vision of the AI agent, so the AI never reacts to the player.
  + 4. The AI agent determined is a non-moving or non-chasing type, so the NPC doesn’t move towards the player.
  + 5. The AI agent is a type which doesn’t have an attack, so the agent never attacks the player.
* Post conditions:
  + The AI agent causes the enemy character to move appropriately and give chase to the player.
  + The slash attack either kills the player or fails because the player agent has dodged.
* Priority: High
* ID: FR5
* Name: Receive Event Die
* Summary: The agent has been killed.
* Actors: AI agent, Player
* Preconditions: The player is in the vicinity.
* Sequence:
  + 1. The player or the AI agent dies due to being hit by a rock.
  + 2. The hit agent’s sprite disappears.
  + 3. The player’s score or lives is updated.
* Exceptions:
  + 1. The AI agent doesn’t die – since there is no friendly fire.
  + 1. The player doesn’t die – since he cannot kill himself with his own rocks.
  + 3. The enemy NPC killed was a type which does not give score points.
* Post conditions:
  + The agent is no more, or if it’s a player dying, he respawns or gets a game-over.
* Priority: Medium
* ID: FR6
* Name: Store high scores
* Summary: The score is saved and sent to the server.
* Actors: Player, Server
* Preconditions: The player has completed a level or ended an survival mode run.
* Sequence:
  + 1. The player reached the ‘game over’ state.
  + 2. The player is asked to input his name.
  + 3. The player’s score is saved.
  + 4. The player’s score is uploaded to the server.
  + 5. The server stores the player’s high score.
* Exceptions:
  + 2. The player inputs a name which is too long or an empty string, this causes a pop up telling you to input a correct name.
  + 4. The server being offline, or the device having no internet connection. This results in a pop up informing the player that the high score will be sent to the server as soon as a connection is established.
* Post conditions:
  + The player can now view his high score and compare it to other people’s.
* Priority: Low
* ID: FR7
* Name: Fetch Quest
* Summary: The player engages in dialogue with an NPC and gets a quest.
* Actors: Player, AI agent
* Preconditions: The player has started a campaign level.
* Sequence:
  + 1. The player approaches a friendly NPC.
  + 2. The player presses spacebar (or interact button on Android).
  + 3. The dialogue window appears.
  + 4. The player taps the screen/presses spacebar to progress with the dialogue.
  + 5. The NPC gives the player a quest to find another NPC and return.
  + 6. The dialogue ends.
  + 7. The player goes and finds the NPC which he was told to find meeting the quest condition.
  + 8. The player engages in dialogue with the found NPC.
  + 9. The found NPC disappears.
  + 10. The player returns to the quest giver.
  + 11. The player starts a dialogue with the quest-giver NPC.
  + 12. The player is given a quest reward.
* Exceptions:
  + 1. The NPC approached is not friendly, this can cause the death of the player character due to being attacked.
  + 4. There could be no dialogues left to progress – ending the dialogue sequence.
  + 5. The quest could have already been done – so the quest won’t be given again.
  + 7. The player could have met the NPC without having to have taken the quest. Without the quest the player cannot meet the quest condition.
  + 12. The player could have found the wrong NPC or failed to find the NPC therefore not finishing the quest, so there is no reward given.
* Post conditions:
  + The player received a reward and the quest-giver NPC
* Priority: Low
* ID: FR8
* Name: Doors Opened
* Summary: The player acquires a key and new passages are unlocked.
* Actors: Player
* Preconditions: The game has been started in campaign mode.
* Sequence:
  + 1. The player collects a key of a certain colour.
  + 2. Passages blocked by unlockable doors corresponding to that coloured key are unlocked.
* Exceptions:
  + 2. There could be no blocked passages unlocked by that key.
* Post conditions:
  + The player can walk through formerly blocked passages into a new area.
  + Priority: Medium
* ID: FR9

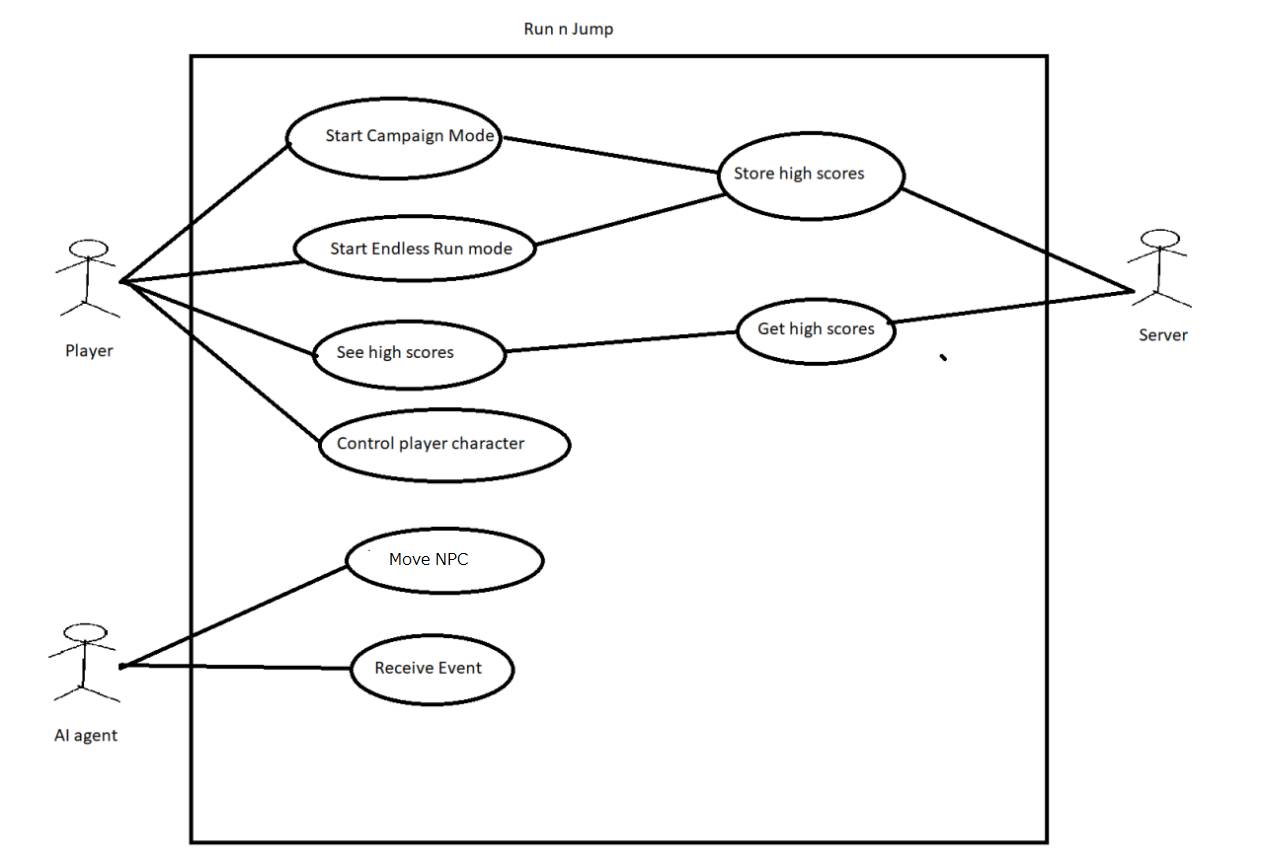
## 4.3 Other Requirements

The system will be scalable, it will be robust enough that new features can be added without much difficulty and without adverse effect on the game’s performance.

It will feel like a finished product – unfinished/unpolished features which could negatively impact the game’s experience will be discarded until they can be finalised.

The game will run on Android and Windows 10 devices with at least 1gb of RAM memory at a normal minimum framerate of 60 per second.

## 4.4 Use Case Diagram



# Chapter 5: Design and Implementation

## 5.1. Implementation Plan

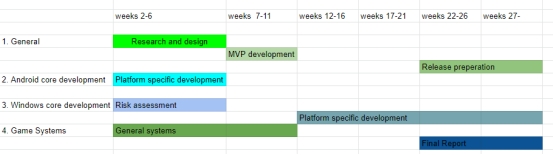


Figure 2 Early Implementation Plan

The implementation plan envisioned focusing on Android development up to the 6th week and later focusing on Windows. This was under the assumption that the Android version would be to a high standard by weeks 12-16. Although focus was actually given to the Windows version due to its quicker debugging and ease of development. Android version only caught up with the Windows version towards the last weeks of the project during release preparation stage. The last few weeks of the project were the weeks with the highest amount of work given, crunch time, alongside work on the final report there was release preparation which consisted of implementing and fixing systems which were not up to the standard envisioned. This is when certain design level decisions were made such as when the decision to only have a local high-score system rather than an online leader board was made due to time constraints.

## 5.2 Package Diagram

For the sake of readability in this package diagram only the most important dependencies are shown on figure 3.

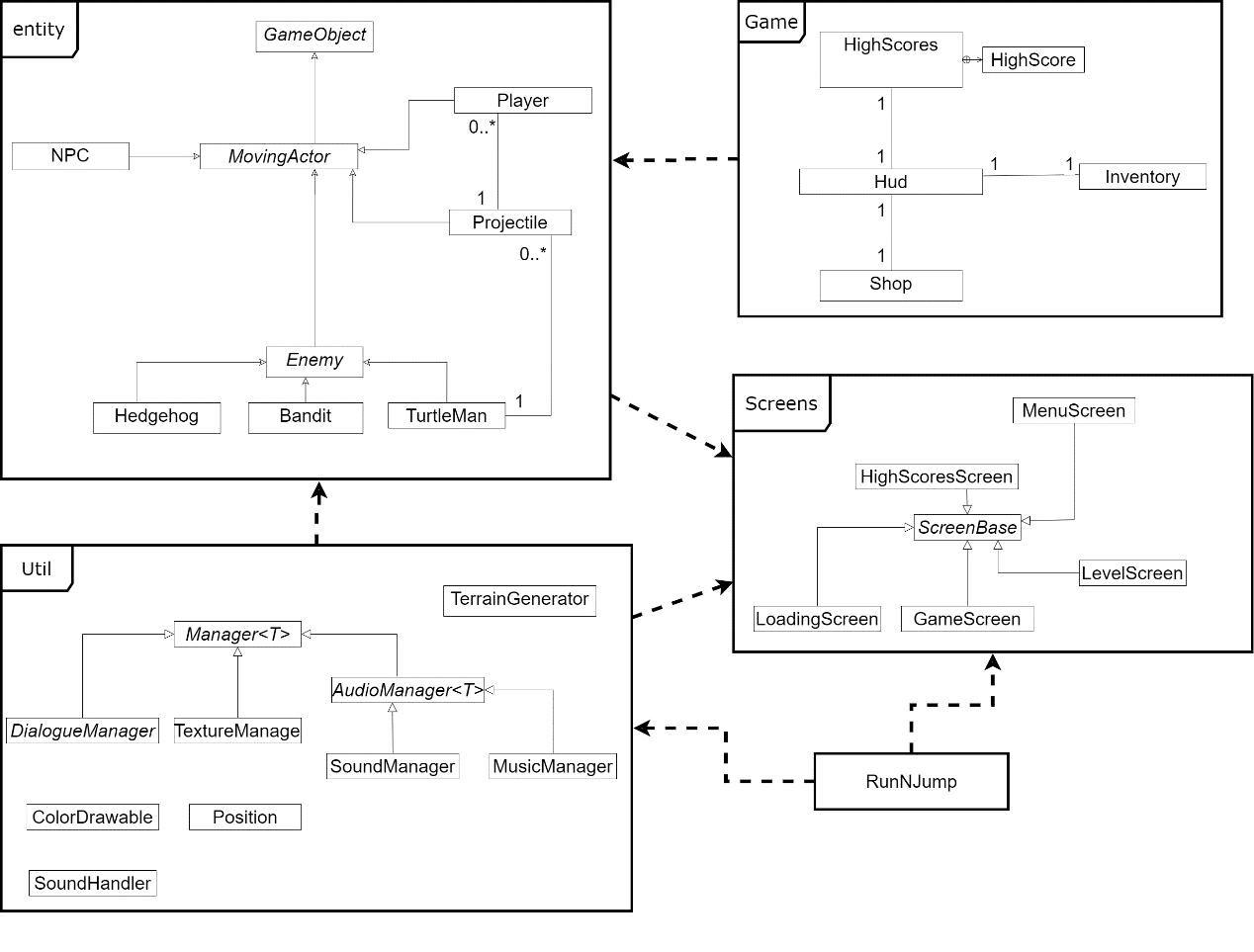


Figure 3 System Package Diagram

The project implementation consists of four key packages, entity, util, game and screens with the RunNJump class serving as the program entry point with the main method.

The Screens package consists of all the screens present in the program; the entire game is a series of screens. Each screen represents a ‘window’ in the game. Gameplay happens on the GameScreen, loading of assets happens on the LoadingScreen etc. Each screen is a child of the ScreenBase class. Each screen renders its own user interface. The entity package consists of all the dynamic game objects present in the game, these are rendered and updated in the GameScreen. The choice to employ screens in such a way ensured it kept different sections of the game separate from each other and ensured that it would be easy to see where user interface for each part of the game was located. I had considered using another semi-transparent screen as a heads-up display, but instead I had gone with its own unrelated implementation due to the complexity and difficulty of placing a semi-transparent screen on top of another screen.

The game package consists of game elements which don’t fit into other packages, such as high scores, shop, and the heads-up display. This package mainly has a dependency with the Player entity class which has an Inventory and a Hud which is updated when inventory changes occur, as seen in figure 4.

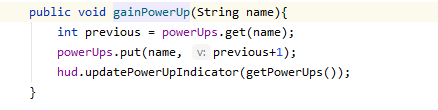


Figure 4 HUD being updated upon inventory creation.

The Util package consists of certain utility classes such as ColorDrawable, Position, etc but mainly it consists of asset manager classes which inherit from the abstract Manager class. Each asset manager is used for preloading and storing the specified asset type for easy retrieval in the entity classes.

The LoadingScreen preloads the assets using the managers by loading all the assets on a new loading thread. The thread sets an AtomicBoolean to true while its loading, setting it to false when it’s done so the program knows it has finished loading and can proceed from the loading screen to the next screen. The choice to use a new thread for loading was easy, since if the loading happened on the same thread as rendering, there would be no way to render anything while the game is loading so the user would see only a black screen. This way the user sees a background image while the game is loading. Figures 5 & 6 demonstrate this in code and in practice respectively.

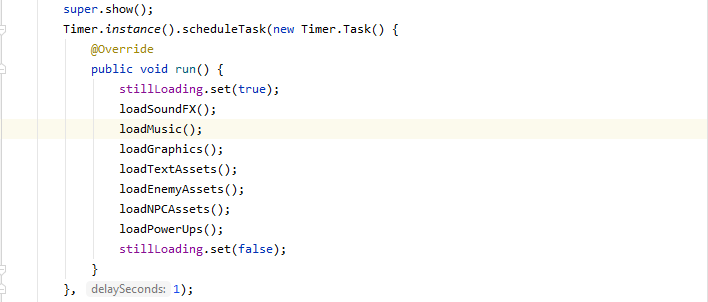


Figure 5 Asset pre-loading.



Figure 6 Loading Screen

## 5.3 Class Diagrams

The following class diagrams only show the within-package dependencies, and only the most important ones. Attributes of the classes are omitted for the sake of greater readability.

### 5.3.1 Entity Package

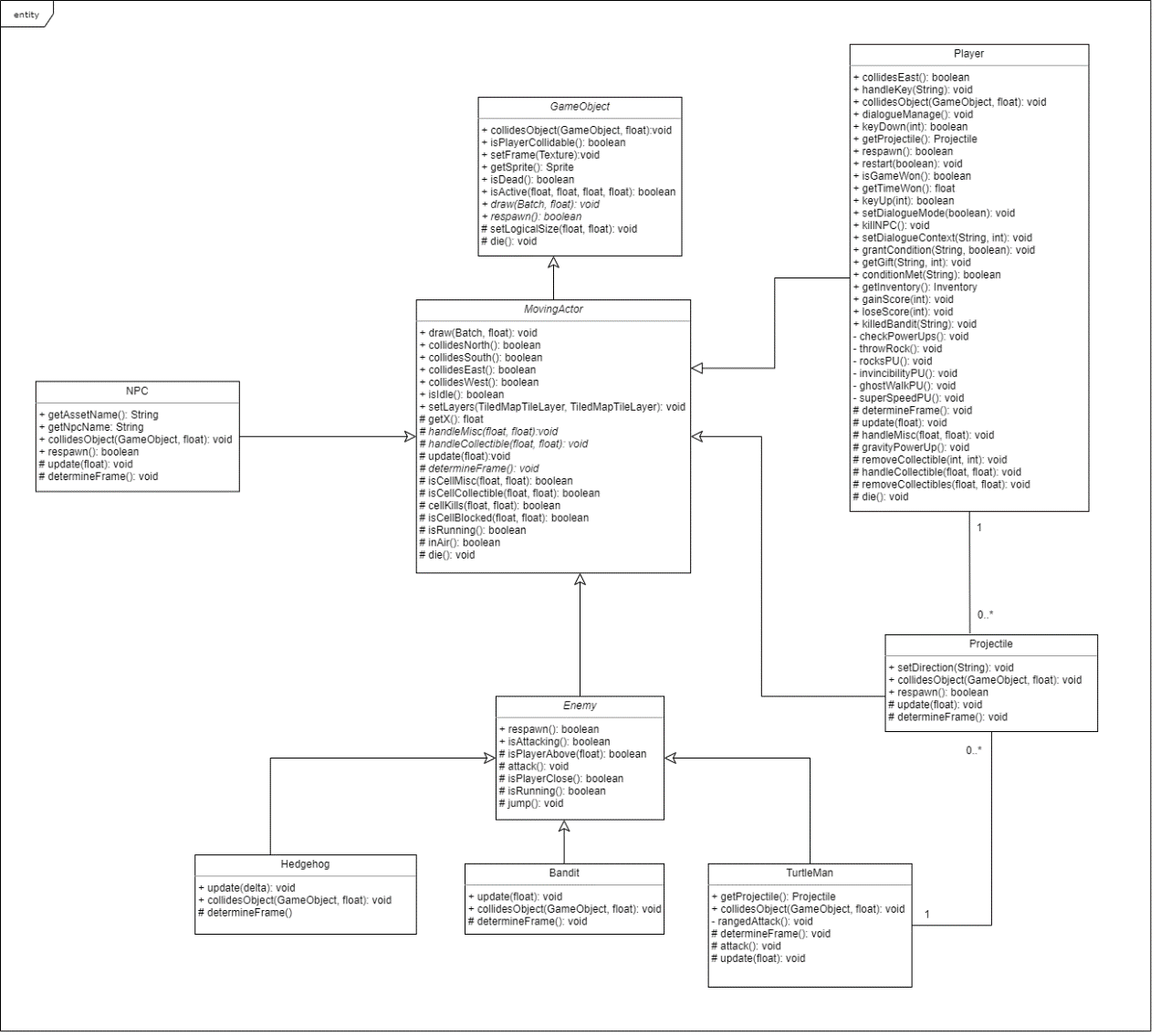


Figure 7 entity package class diagram.

GameObject is the parent of all entity classes, as seen on figure 7, this is to allow easy updating and collision handling using method overloading and polymorphism

### 5.3.2 Game package

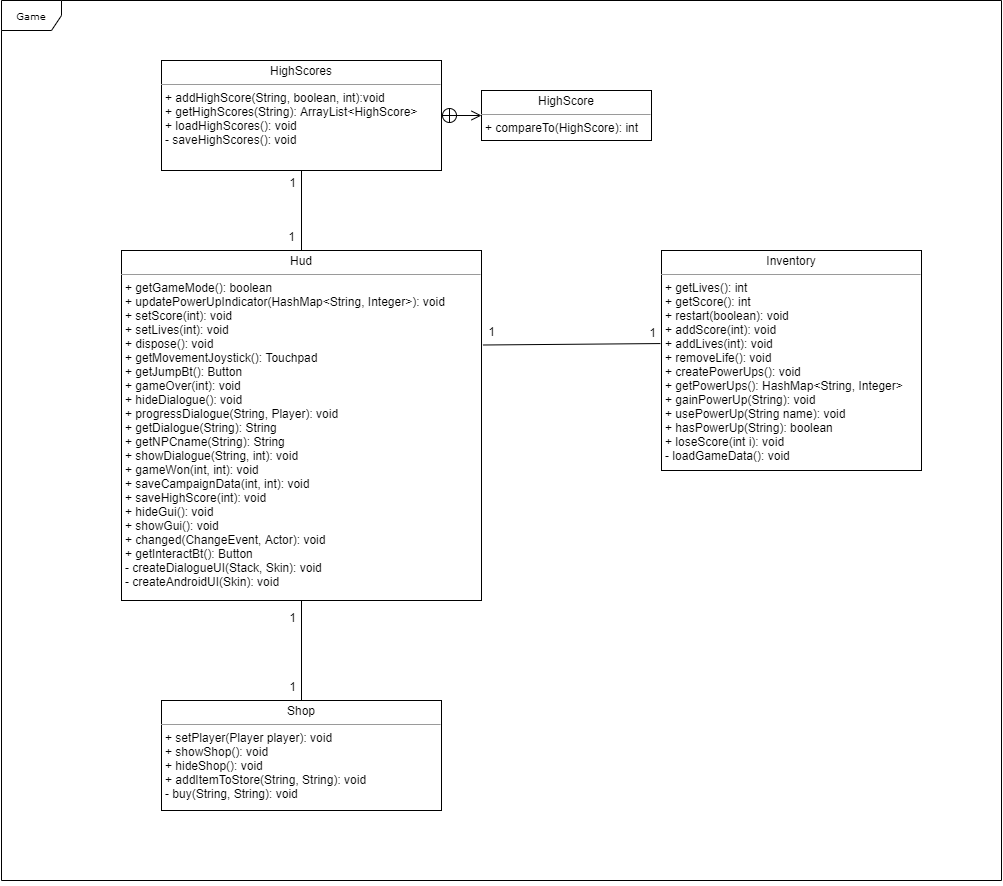


Figure 8 Game package class diagram.

As can be seen on figure 8, the game package consists of the key Hud class which has a link to Inventory, Shop and HighScores, HighScores is the class used for managing the high-score system, it has an inner class which represents an individual high score.

### 5.3.3 Screen package

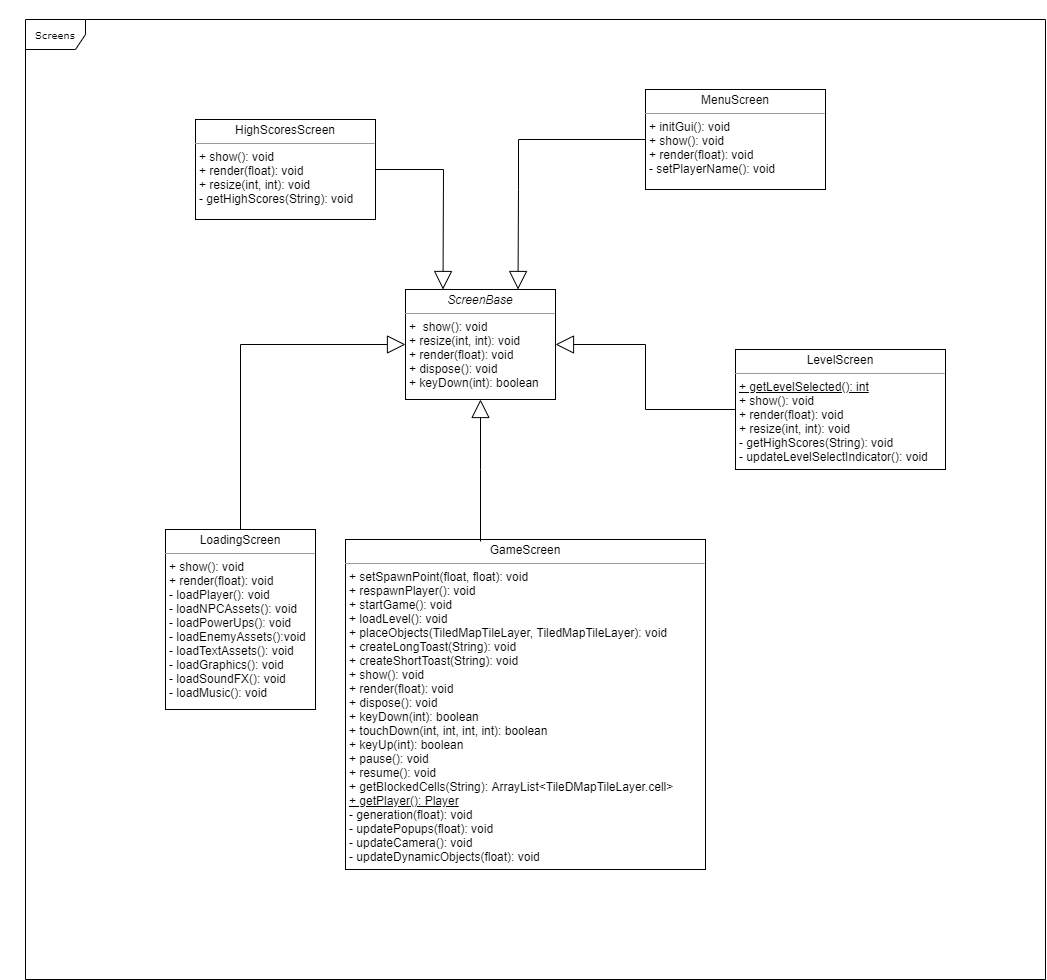


Figure 9 Screen package class diagram

Figure 9 shows the class diagram for the Screens package, all screens inherit from the ScreenBase superclass.

### 5.3.4 Util package

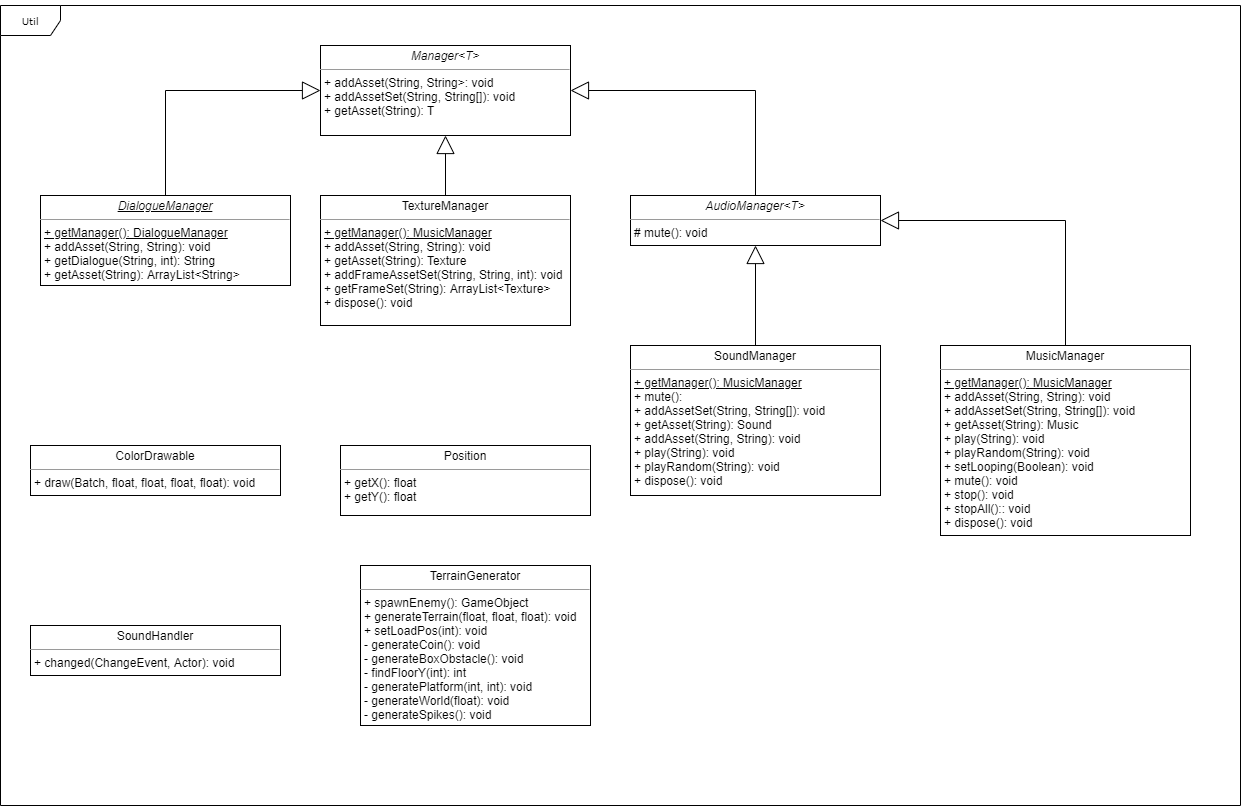


Figure 10 Util package class diagram

Figure 10 shows the class diagram for the Util package, as can be seen on the diagram, all asset managers inherit from the Manager abstract generic class. AudioManager is another abstract manager class which is the superclass for sound and music managers. These managers are used for storing a certain type of asset. The other classes are unconnected on the diagram due to them being dependencies of classes outside the util package. They are helper classes used by other packages.

## 5.4 Tools Utilised

The game is built on LibGdx cross-platform Java game development framework which is based on OpenGL. This framework allows keeping most of the code-base working on Android and Windows, with only requiring minimum platform-specific code. It is free and is easy to use while being powerful for 2D games. Primarily it was used to render things onto the screen. It was employed alongside Android Studio which has a built in Android emulator.

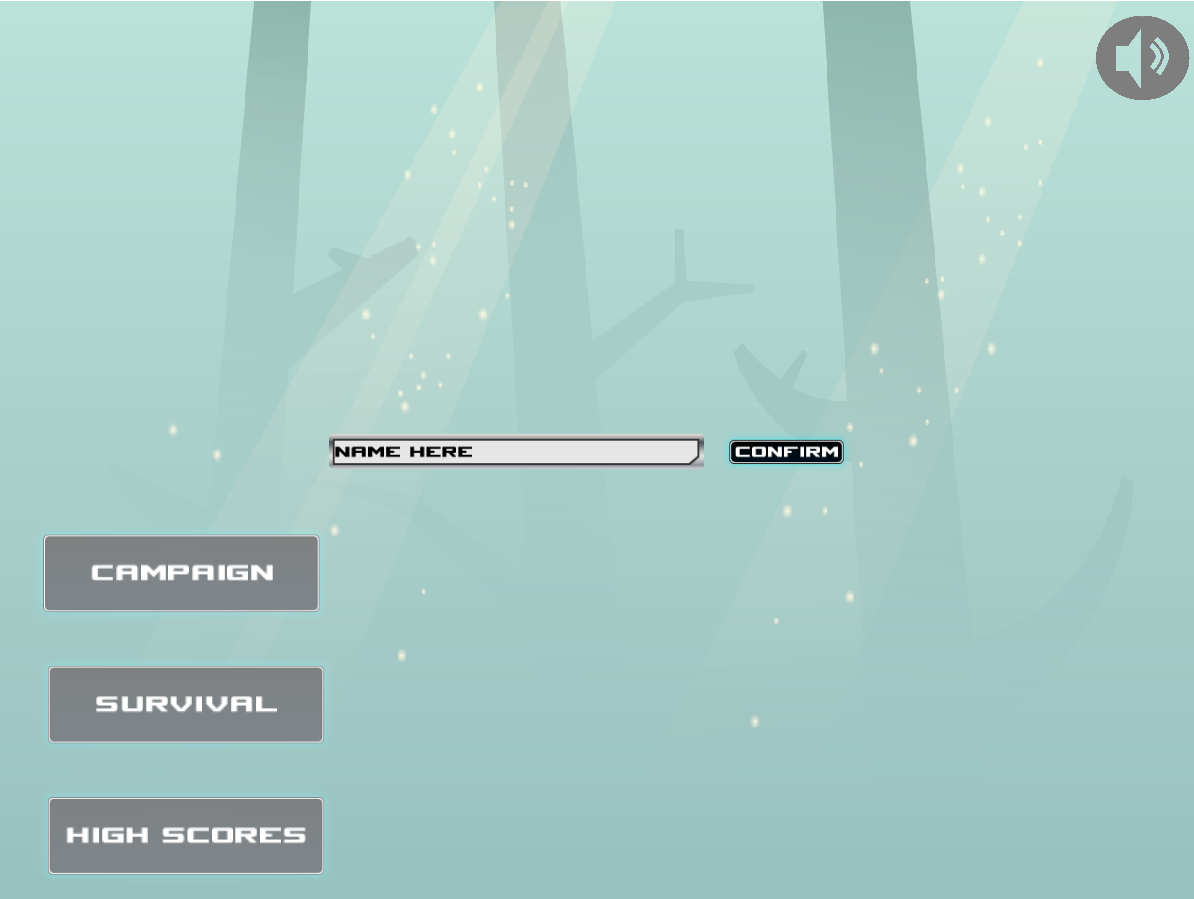
To create the game world and levels the Tiled TMX level editor was used, it is a free and open source easy to use level editor for tile-based levels. This was used to create a TMX file representing a level using tile-sets assembled from sprites using a TexturePacker, which can be used assemble sprites into a sprite-sheet for more efficient sprite access in videogames. For the campaign game mode, the level would be created entirely within Tiled, with object markers being placed to specify the spawn locations of entities. To play the created level the GameScreen class would load it into a class and then parse the file placing all my entity objects in their locations specified by the tile map.

For multi-platform toast functionality, the Toast library/class was imported which was placed in the Libs package. This class allows for similar Android toast notification functionality on Windows. It is a message which appears for a certain amount of time on the bottom of the screen. This was used to give notifications and feedback to the player.

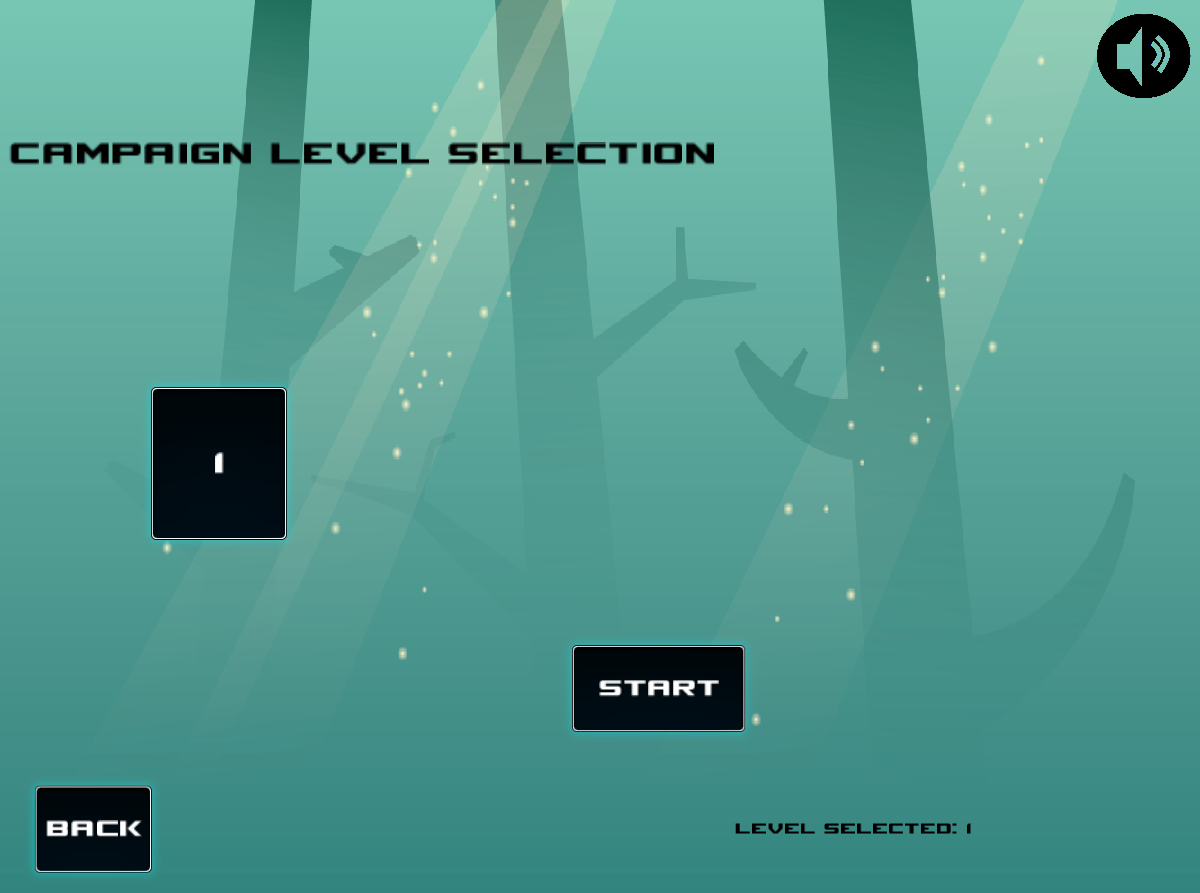
Most of the assets used in the game are assets which I have purchased previously and have rights to use commercially, a few small assets I had drawn myself.

## 5.5 Game Description

### 5.5.1 Game Walkthrough



Upon launching the game for the first time the player will be asked to input their name, which will be saved and on subsequent launches of the game this prompt will not appear. This can only be changed by clearing save game data. The player can mute sound with the sound button in the corner, see high scores and choose the game mode in this screen.



Upon selecting the campaign mode, the user will see the level selection screen, this is where the unlocked levels appear and can be selected. On the screenshot above only the first level is unlocked – due to the player not having completed it and not unlocked the next level.



Upon starting the first level on Windows, this is what the player will see. In the top left corner there is the score, lives left and inventory indicators. The old man is a friendly NPC the player can talk to.



The old man grants the player a quest to find his daughter. The objective is to find the female NPC and come back to Jack for a reward. The yellow coins and stars are collectibles which grant score points.



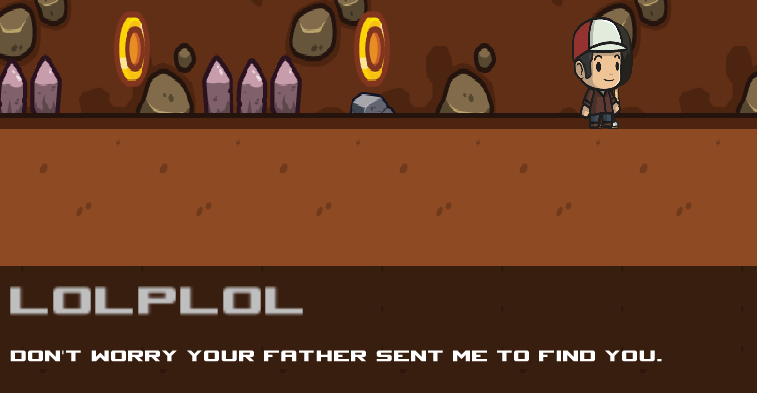
The player can die by touching the spikes on the ground.



Upon collecting a power-up the player will see his inventory display changing.



When something notable happens, the player gets notified through an Android-style toast message.



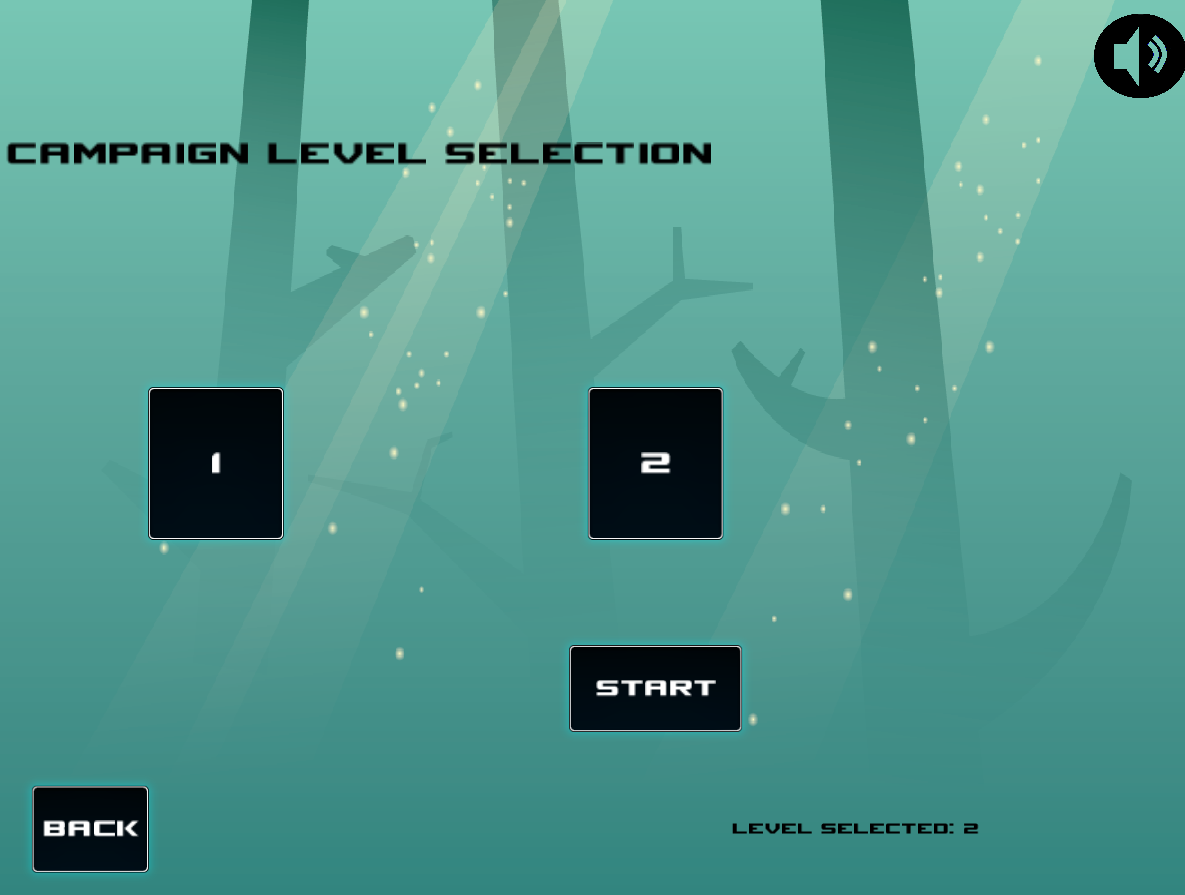
Upon rescuing the daughter for the quest, she will disappear, and you will need to return to the quest giver for the reward.



This quest reward grants the player 15 score points.



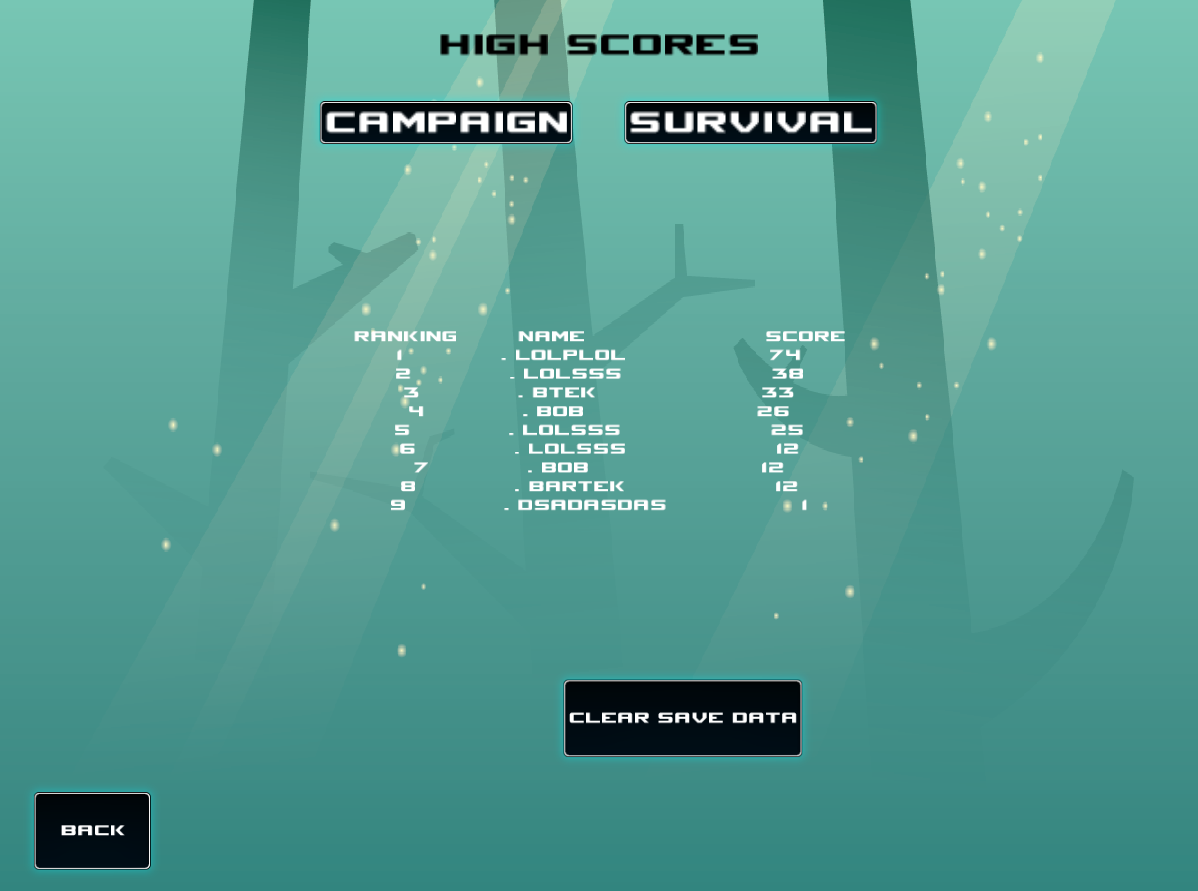
A red flag indicates the end of level flag, upon touching it you win the level. The goal of the game is to find it after acquiring as many points as possible. Upon winning a level your current inventory, score points and lives left are saved. You keep everything when you start the next level.



As you can see above, after completing the first level, the second level was unlocked.



On the second level there is a vendor NPC which can sell you power ups and a key for score points.



On the high scores screen you can view your leader board rankings for the two game modes. This is also where you can clear your campaign saved data and name.

### 5.5.2 Gameplay Elements

Powerups



The mushroom grants the player lower gravity for 9 seconds.



The green egg grants the player invincibility from any hazards for 8 seconds.



The white egg grants the player the ghost-walk ability, this allows the player to pass through walls horizontally. It is the most powerful and therefore the rarest ability. It lasts for 4 seconds. If the player is inside a wall, the ability stays active to prevent the player getting stuck.



The piece of meat grants the player super-speed. The player runs twice as fast for 9 seconds.



This rock grants the player the ability to throw rocks, simply tap the android interact button or press the rocks hotkey on Windows to throw a rock. The player can throw as many rocks as you want for 8 seconds. The rocks kill enemy characters.

Collectibles



The keys unlock locked passages, the grey key unlocks stone doors, while the golden key unlocks purple gates.



The coins grant one score point, the stars grant ten score points. Hearts grant an extra life.

NPCs

These two are friendly NPCs which grant quests or need to be rescued depending on the level. They can talk to you or sell you items.

Enemy Agents



This is the hedgehog, he is the simplest enemy type, he only patrols from one point to another. He can only be killed by a thrown rock.



This is a bandit swordsman which can kill the player with his sword slash animation but can be killed by jumping on top of its head or a thrown rock, he chases the player once he sees him come close enough, jumping to reach the player if necessary.



This is the rock throwing bandit or turtle man. He is the hardest enemy to defeat. He throws rocks at the player with good aim, they kill the player. When the player gets close enough the turtle man takes out his sword and slashes at him which can kill the player. He can be killed by jumping on his head or throwing a rock at him.

The white flag indicates a checkpoint in the level – after touching it the flag is set as the respawn location of the player when he dies while still having lives left.

The red flag indicates the end of the level flag, the goal of the game is to reach and touch this flag.

### 5.5.3 Control Scheme

Windows key bindings

* Movement
  + W – Jump key
  + A – Walk left key
  + D – Walk right key
* Utility
  + Spacebar – Interact key, can be used for talking to NPCs.
  + Left mouse click, identical purpose to spacebar.
  + Number keys – can be used to activate a certain power up.
    - 1 – Lower gravity power up
    - 2 – Super speed power up
    - 3 – Ghost-walk power up
    - 4 – Invincibility power up
    - 5 – Rock throwing power up.
    - In addition to activating rock throwing, number 5 key is used to throw the rocks when the power up is active.

.

Android due to its nature as a touch screen device uses on-screen touch controls.



Figure Android UI

* The joystick on the left is used for left-right movement.
* The little rock above the next button is the button you tap to activate a power up. The button image corresponds to the currently selected power up.
* To switch to another powerup you tap the next button
* The interact button is used to talk to friendly NPCs and to throw rocks when the rock throwing power up is active.
* The jump button is used to make the player character jump.

# Chapter 6: Testing

## 6.1 Debugging

While the development of the product was ongoing every new feature was extensively tested. Throughout the codebase there are numerous debug statements.

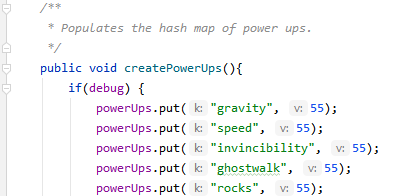


Figure Inventory debug mode.

As seen on figure 12, the Inventory and power-up systems were tested by running the game with a debug flag turned on. This flag gave the tester an almost endless supply of power-ups to use and check their functionality. This allowed the tester to test each power-up multiple times. While checking if the power up was removed from the inventory with each use. Furthermore, completing a level with unlimited power-ups is trivially easy, so it was easy to test other aspects of the level design, collision systems and difficulty while being able to get to any spot on any level quickly. Completing a campaign level with this flag lets you quickly test the save-game functionality – to check if the saved debug inventory stays until the next level, even when you disable the debug flag.

While testing the survival game mode I found out that on a weaker Android device, the survival game mode can affect the performance adversely. This was due to the terrain generation algorithm progressing infinitely. This was an easy optimization by ensuring only 200 blocks

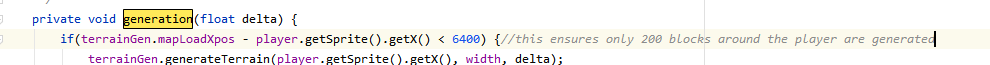


Figure Optimization of generation.

Figure 13 shows my method not generating terrain if the player is 200 tiles or more away from the already generated section of terrain.

The in-built Android and Windows Android studio debuggers were used to walk through the code and find issues. Collisions were tested for weeks to ensure there would be no weird boundary issues.

## 6.2 Devices Tested On

Android

* An emulated Galaxy Nexus API level 28, 720\*1280 resolution with 1024 MB of RAM.
* A physical device Huawei p8 1920x1080 resolution with 3gb of RAM.

Windows 10

* Windows 10 Desktop PC with 16 GB of ram i7-4790 CPU, GTX 750 TI graphics card.

## 6.3 Known Bugs

Currently there is several minor known bugs, but due to time-constraints, their low priority and low seriousness they were left in the code. Every software product has some unintended glitches or errors when the user uses the product in an unexpected way. Similarly, my game is no different.

* Ghost-walk
  + Allows the player to walk off and fall off the map which shouldn’t be possible. This is a quick fix, but very low priority.
  + Sometimes prevents the player from being able to collect a collectible. Technically could be considered a feature rather than a bug – you are a ghost so you shouldn’t be able to grab things.
* Survival game mode
  + The coins don’t spawn when they should. There is an embedded tile-set loading issue which likely requires a re-do of the survival game mode tile-sets.
  + Sometimes the score isn’t saved to the leader board.
* Terrain graphical glitches.
  + Sometimes tiny gaps appear in-between the tiles on the level. These gaps quickly disappear when the player moves. It is hard to reproduce this intentionally.
* Android Toast text being too small to read on some devices.

## 6.4 Testing Strategy

The testing has been done against the functional requirements outlined in 4.2. The usage of a unit testing tool such as Junit or similar was under consideration but it is difficult to test a game environment with automated unit tests, so in the end it was decided to do systems testing of the whole system by actually running it and testing each feature, employing a debugger when deemed useful. Occasionally when testing instead of a debugger I used a classic java System.out.println() statement to print the variable values at certain stages.

# Chapter 7: Summary of Technical Documentation

The complete technical documentation is on GitLab here - <https://cseegit.essex.ac.uk/ce301_2020/ce301_markiewicz_bartosz_n/-/blob/6072fad18c9b934441cb0c87cc9fdbb1d3343996/README.md>

The instructions on how to run the project are written in the readme above.

## 7.1 Outline of the Working System

The system works on top of LibGdx. The entry point of the application is the Runnjump class which extends the LibGdx Game class. The game It is built as a series of Screens, each interface/window you see is a Screen, with its own class which extends the base ScreenBase class. Main menu is the MainMenu screen, the game with the gameplay itself is the GameScreen etc. Each time a new screen is accessed its set using the Runnjump changeScreen method.

Initially when the application is launched, the first screen which appears is the loading screen, this is where the loading of the game assets occurs. They are loaded and stored in their individual asset manager classes. The loading occurs on an alternative thread – to allow the display of a loading screen while loading occurs. An AtomicBoolean flag is set when the loading is finished, and the screen is changed to the menu screen.

The player can then progress through the different menus or start the game by clicking Survival. When the actual game is started, the GameScreen is the screen in focus. GameScreen loads the selected level/game-mode based on its TMX tile-map and parses it for its object’s information. If it is campaign game mode it uses the tile map objects to place the Java objects from their classes in the appropriate positions on the tile map.

Each entity represents a dynamic object on the game world. They are all children of the GameObject abstract class, this is to make the update code more readable. Each game object is in a dynamicGameObjects arraylist which is looped through each iteration of the render method, to ensure every object is rendered and updated. The player object is the most important object, it implements an InputProcessor to ensure the player character can be controlled, the player character has a Hud and inventory objects to update the heads-up display and have an inventory. Each dynamic object class has an update method where its own AI logic is implemented and a determineFrame() method which determine the sprite on that particular frame for the entity, this method ensures the sprite is animated.

If the survival game mode was selected, the loaded map for it is essentially only a template on which the random terrain generation and object spawning occurs. TerrainGenerator class is used to place the terrain elements on the survival map.

# Chapter 8: Project Management

## 8.1 Work Done

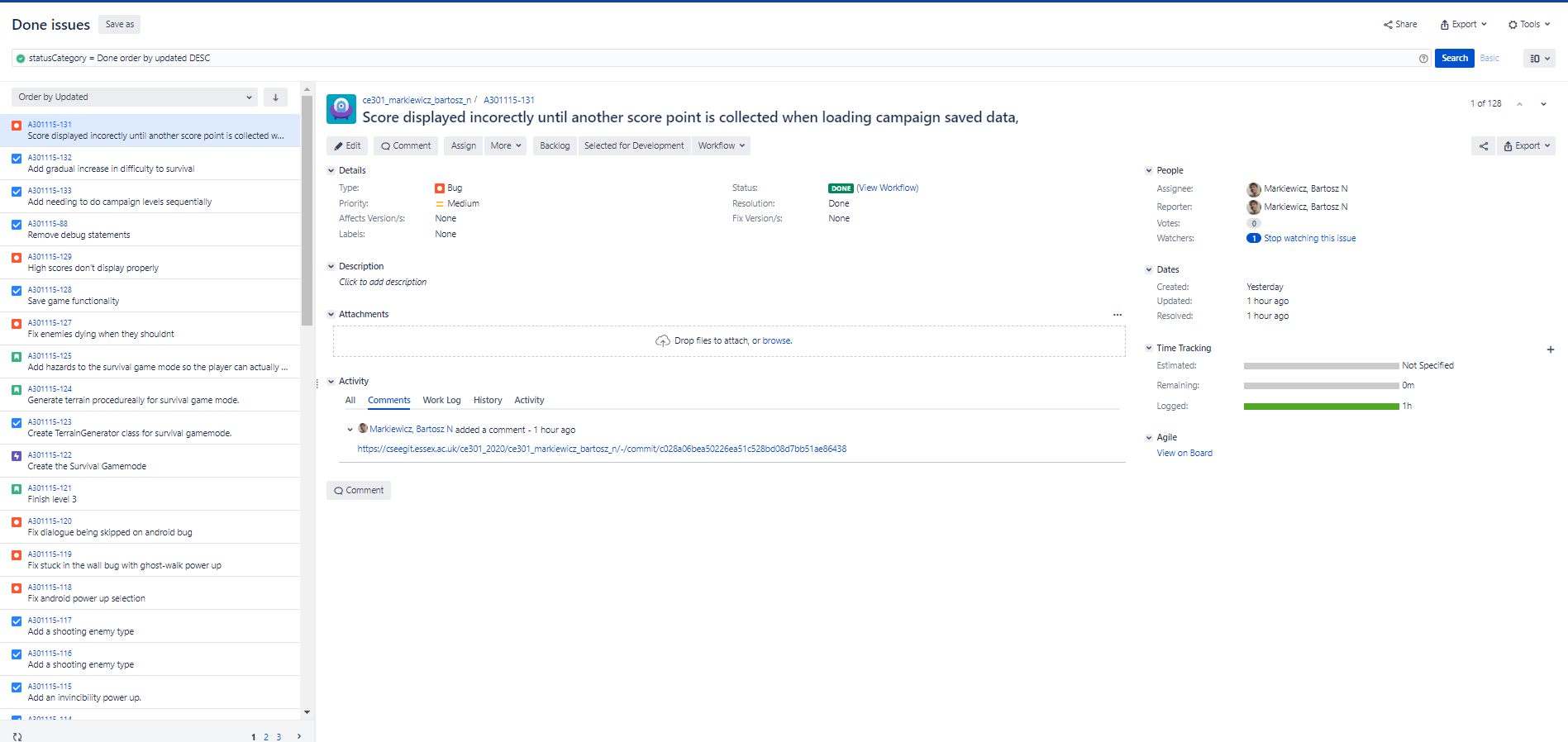


Figure Jira Issue View

Throughout the development of this product I had done 128 issues on Jira, that is on average 4 a week, as evidenced by figure 14 and my Jira project.

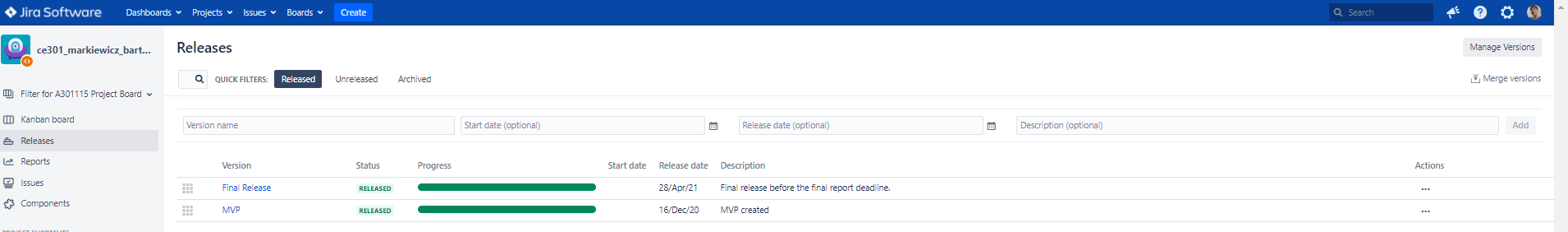


Figure Releases

Two versions of the product were released, the minimum viable product in week 11, and the final version on the 28th of April. Though final minor bug fixes and small issues continued to be worked on up until the 30th.

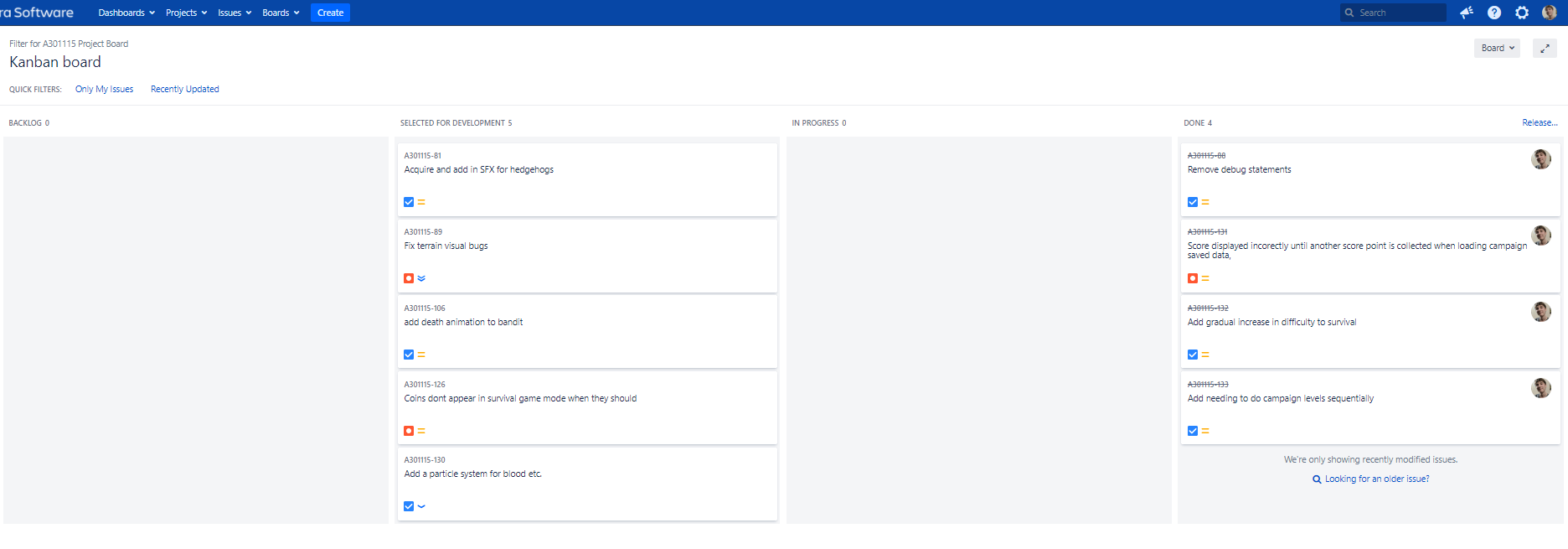


Figure Project Board as of 30/4/2021

Figure 16 shows the project board as of the deadline, following the actual release of the final version on Jira. Some bugs and issues are still left to do, this is due to time constraints and the deadline being close.

## 8.2 Cumulative Flow Diagram

Figure Jira issue Cumulative Flow Diagram

Figure 17 indicates a cumulative flow diagram which displays the progress of my development in terms of issues completed. It clearly shows a steady trend of progress in terms of issues in the backlog and finished ones. Towards the end it shows the completion of most features.

## 8.3 Weekly Meetings

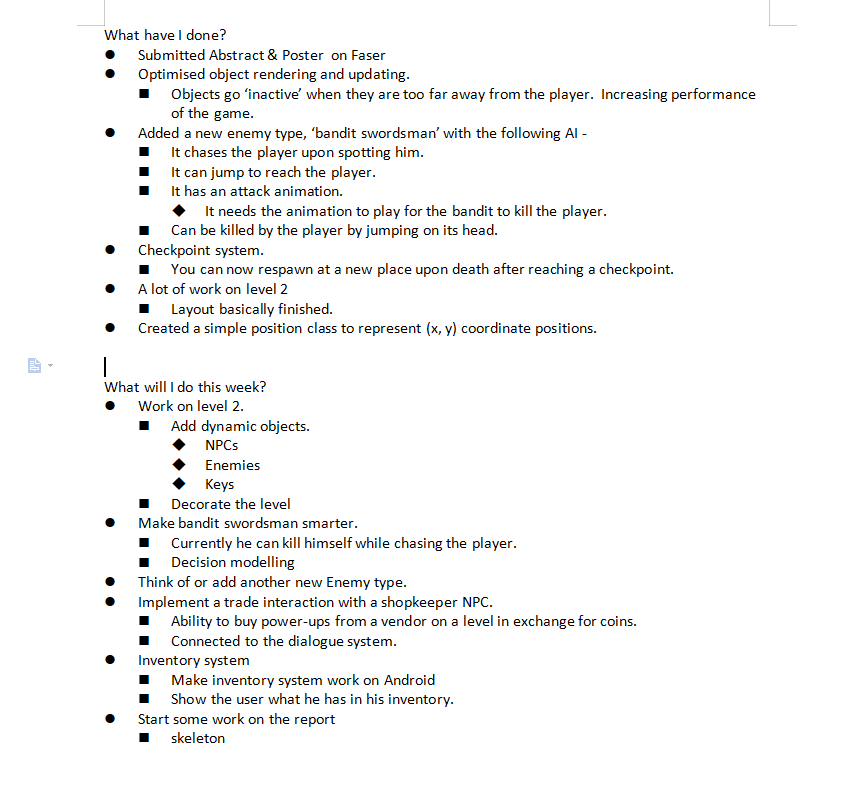


Figure Weekly progress report to project supervisor.

At each weekly meeting with the project supervisor, the product development was kept on track. Everything discussed and planned each week was kept note of in a series of weekly word documents. While showing the supervisor the work done each week and the work to be done next week, the supervisor assisted me with feedback and answering questions, answers to which I usually noted down in a text file or word document on hand. Figure 18 shows an example document I showed my supervisor, the example is from 21/03/2021.

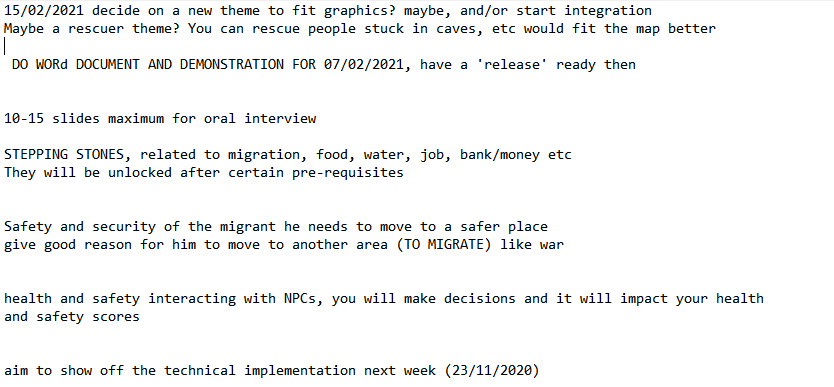


Figure My personal rough notes of a supervisor meeting

During some meetings I had utilised a notepad file to note down the important points of a supervisor meeting to make sure I didn’t forget, figure 19 shows my rough notes based on supervisor meetings.

## 8.4 Legal Issues

Most of the assets utilised are from [www.gamedevmarket.net](http://www.gamedevmarket.net) which is a website where you can purchase assets for videogames for commercial use. The assets were legally purchased and therefore come under their ‘pro license’. The license grants the purchaser the right to use the asset in an unlimited number of monetized and non-monetized media products, if the asset is not sub-licensed or used as a logo or trademark. Since the assets are used for a currently unmonetized single media product, it falls within my legal rights to use the assets employed in the project. [14] In addition a Toast library was used, created by Tomas Chalupnik, imported from GitHub, this library is distributed under the GNU public license which allows me to use it for my commercial product as long as I keep the license with the library and give credit to the author. [15] LibGdx the main library used is an open-source library distributed under the Apache 2.0 license. Allowing for use in all commercial applications without royalties. This clearly allows me to create a product using it with no concerns for legal troubles.

## 8.5 Risk Management

Every software development project has risk involved. Software development projects can fail or be delayed due to unforeseen factors such as unexpected difficulties in implementation, badly defined system requirements, wrong estimates of project resource requirements etc. As the project gets bigger more time will need to be spent towards maintenance and code clean up to avoid failure.

To mitigate these risks when planning the deliverables, the focus was on under-promising and over-delivering. Up to week 11 I had focused on the Android version, but after getting the minimum viable product out I had realised that focusing on Windows would be better due to the Windows version being better at debugging and testing. The Android emulator or device simply doesn’t run as quickly as the Windows debugger. The change of focus was necessary in speeding up the testing and development process. New features would be added and tested initially on the Windows version and only later integrated and tested on Android. Jira was utilised to plan the weekly work. Focus on tasks was ensured by meeting weekly with my project supervisor and deciding with him what aspect of the project should be focused on each week or if anything should be de-prioritised.

Many project managers overestimate their team’s capability and downplay the possible risks that is why the plan is realistic in its estimation of what I am capable of in 30 weeks of the project. It is better to be conservative in planning and be pleasantly surprised rather than be over-optimistic and then be disappointed in the result.

Lastly to manage these risks, Jira with agile project management methodology was utilised. Weekly meetings with my supervisor ensured I had constant stream of feedback and someone to keep me accountable to my promises.

# Conclusions

## Summary

The project evolved throughout development to a large degree, initially the plan was to have a serious platformer with educational value on the issue of human trafficking although an issue with the serious tone of the topic emerged. Since the graphics were largely things, I already had the rights to, and lacking artistic talent I had to adapt the theme of the game to the assets rather than other way round as is typical. The topic of migration in general was considered, but it proved too convoluted, aside from the assets not matching the tone, designing an arcade platforming game to be educational proved difficult. If the game had educational value, it would lose too much on enjoyment. There was an initial design idea of having educational mini games throughout the campaign game levels, but after acquiring feedback and realizing something more light-hearted would be more appropriate, this idea was abandoned. Another issue was that it would ruin the player’s immersion and take him out of the experience. So, in the end the project evolved into a casual arcade game with soft role-playing game elements.

After playing Super Mario Bros the experience was analysed and the result was that there was something missing, the experience was very basic and lacking in variety or intellectual stimulation. The conclusion was drawn that there could be something to innovate on the classic formula. I chose to focus on things such as narrative and player choice instead of the Mario focus on strict platforming. The game was to include multiple ways to complete a level and grant the player the choice when to use a power-up. In the classic Mario game, the player can’t even go backwards on the level. In my game the player can go in all directions and return to areas he already explored - it’s even encouraged using the quest system. For example, on the first level the player can acquire an optional quest to rescue a lost woman. The player can ignore it or find the woman, after finding the woman the player needs to return to the quest-giver to acquire the reward of the quest. This going-backwards is not necessary, but the rewards can be score points or power-ups.

The power ups can be used when the player wants to use them rather than when they appear in the level - giving the player more choice on how to proceed. For example when a player is at a difficult part of a level, he can use the power-ups he had stockpiled through doing quests and finding them while he found the game easy, this allows the player to complete the ‘hard’ parts of a level easier if he planned ahead. This rewards player’s smart thinking.

Score in Mario games is typically just something used for leader board rankings, while this is useful and can be fun there is a subset of casual users who do not care for competition and therefore don’t care about their scores. This product takes this into account and allows the player to spend his acquired score throughout the previous levels to purchase something which would be useful such as a power-up or a key for a secret area. Essentially the player can sacrifice his score in exchange for making the game easier. This is done through the NPC store on level 2 - the vendor NPC first needs to be found, in this way it also rewards exploration of the level, if the user went directly to the end of the level without doing any quests or exploring the level he would end up playing level 3 without many power-ups and lives. The third level is intentionally created to be the hardest to complete.

## Critical Evaluation

The product has met most of the objectives it had set out to achieve. The game is fully playable on two platforms, it has as many power-ups, enemy types and collectibles as promised or even more. Furthermore, almost all the promised systems are developed to a standard promised or even beyond. All the moving characters in the game are animated, sound FX and music are employed. The campaign game mode was finished with three levels, save and load game progress functionality, each level has a unique feel to it with multiple routes to take to completing it, meeting the promise of non-linear gameplay. The dialogue system feels like a visual novel and works well, the quest system gives a small narrative to what the player is doing. The NPC store system works and gives a good amount of player choice. The survival game mode generates a practically infinite game world to survive in with gradually increasing difficulty. The game is surprisingly enjoyable, engaging in combat and trying to complete the game with as much score as possible is good.

However, despite the game being an improvement on the classic Mario experience, it is not without flaws. Due to time constraints the game feels unfinished at times with certain things missing, furthermore the user interface design leaves much to be desired. Additionally, certain animations which should be there are missing – no death animations or death SFX for enemies. At the very least I intended to add a particle system for deaths and projectile impacts, but time constraints did not permit. The music and sound overall lack much variety. The inventory and other in-game user interface elements are just a wall of text which detract from the experience, if there was more development time it would be made visual rather than textual. The survival game mode while being there and generating a map, isn’t made that well. Also it doesn’t actually generate the world infinitely, it is only practically infinite, due to difficulty in implementation of an actual infinite world the map was set up as large as possible, so large so that the player would never see it ending, but practically there is a theoretical end of the level which can be reached. The world generation algorithms are very basic and result in very ugly unnatural looking levels. The game mode is at a prototype stage. Furthermore due to working on the game mode at a crutch close to the deadline, its unbalanced and too easy to survive early on(it gradually increases in difficulty), also the ‘threat’ you are surviving from is just spikes which appear to the left of the player – this was a quick hack job rather than an actual implementation of a threat. The threat to survive was meant to be another smarter enemy type who would chase the player to the end throughout the survival game mode or at the very least a wall of fire/lava approaching but time constraints required a quick implementation. On Android the on-screen touch controls don’t feel optimal, the interface layout just isn’t effective, it requires a re-design.

Additionally, the high-score system is not as advanced as envisioned, it only has local storage of high scores on the device and no connectivity to an internet high-score server as envisioned early on in design. Lastly the game does not have as much content as intended, the first and third levels are too short. Lastly, the game has numerous known bugs, and likely just as many unknown ones. The biggest bug detected that collectibles do not appear in the survival game mode when they should.

Given more time I would fix the issues outlined above. Additionally, I would add more content such as extra campaign maps and a proper online high scoring system. After polishing the game, I would release the game on the Google store and itch.io PC indie game marketplace.

If I was starting this project today rather than ending it, I would have prioritized different things, instead of prioritizing content creation – levels and all the different systems I would have focused on the most important key systems rather than spreading my focus around. I would have a polished well-made survival game mode and only then I would proceed to work on the campaign mode or the opposite rather than doing a little of everything. Because in the end I ended up with a lot of content but much of that not being a polished experience.

# References

[1] S. A. Arnab. (2012) “The Design Principles for Flow Experience in Educational”. *Procedia Computer Science* 15 ( 2012 ) pg78 – 91. Available:

<https://www.sciencedirect.com/science/article/pii/S1877050912008228?via%3Dihub>

[2] K.R, Robson, “Is it all a game? Understanding the principles of gamification”. *Business Horizons.* Vol 58, issue 4, pg 411-420, Available:

<https://www.sciencedirect.com/science/article/pii/S000768131500035X>

1. K.B, Becker, “Teaching with games: the minesweeper and asteroids experience”. *Journal of Computing Sciences in Colleges.* Vol. 17, issue 2, pg 23-33, Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.5072&rep=rep1&type=pdf>
2. C.J.C, Cai “Adapting Arcade Games for Learning”. *Extended Abstracts on Human Factors in Computing Systems*. Pg 2665-2670, Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.294.5335&rep=rep1&type=pdf>

[5] L.S, Laila Shoukry, “Learning Analytics and Serious Games: Trends and Considerations” *Proceedings of the 2014 ACM International Workshop on Serious Games.* Pg 21-26 Available: <https://dl.acm.org/doi/pdf/10.1145/2656719.2656729>

[6] USgamer (2016, Jan. 1) by Jaz Rignall, Available: <https://www.usgamer.net/articles/top-10-biggest-grossing-arcade-games-of-all-time>

[7] GAMERANT (2020, Sep. 11) by John Rinyu, Available:

<https://gamerant.com/best-90s-arcade-games/>

[8] Nintendo Available: <https://www.nintendo.co.jp/ir/en/finance/software/ds.html>

[9] H.B-L.D “Playing Different Games on Different Phones: An Empirical Study on Mobile Gaming“. *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services.* Pg 391-394 Available: <https://dl.acm.org/doi/pdf/10.1145/1409240.1409296>

[10] BadLogicGames Available: <https://libgdx.badlogicgames.com/features.html>

[11] Rise.Global Available: <https://www.rise.global/>

[12] SFML frequently asked questions Available: <https://www.sfml-dev.org/faq.php>

[13] Pygame Available: <https://www.pygame.org/wiki/about>

[14] GameDevMarket license Available: <https://www.gamedevmarket.net/terms-conditions/#pro-licence>

[15] Toast library license Available: <https://github.com/wentsa/Toast-LibGDX/blob/master/LICENSE>