

# OCR Programming Project

Name: Eusebiu Moldovan

Form: 13GJM

## Contents

<i>Analysis</i> .....	4
<i>Project Outliner</i> .....	4
<i>Stakeholders</i> .....	5
<i>Computational Methods</i> .....	6
<i>Interview</i> .....	8
<i>Research</i> .....	10
<i>Similar Games Analysis</i> .....	12
<i>Features of Solution</i> .....	13
<i>Initial concept of my character</i> .....	14
<i>Game States</i> .....	15
<i>Controls</i> .....	15
<i>Limitations to my solution</i> .....	16
<i>Requirements</i> .....	16
<i>Development Language/IDE</i> .....	16
<i>Stakeholder requirements</i> .....	17
<i>Design</i> .....	18
<i>Basic structure of the solution</i> .....	18
<i>Decomposition of the problem</i> .....	19
<i>Object-oriented design/programming</i> .....	19
<i>A justification of the approach taken</i> .....	19
<i>During the Gameplay U. I. Example</i> .....	20
<i>Default Game Controls</i> .....	21
<i>Structure of the Solution</i> .....	22
<i>Flowchart of solution</i> .....	23
<i>Test Data for Development</i> .....	26
<i>Development</i> .....	28
<i>SDLC Process</i> .....	28
<i>Test data for beta testing</i> .....	29
<i>Class Diagrams</i> .....	31
<i>Key Variables and Data Structures</i> .....	32
<i>Iterative Development</i> .....	35
<i>Coding</i> .....	35
<i>Assets</i> .....	36
<i>Menu Development</i> .....	39
<i>Development Process</i> .....	51

<i>Evaluation .....</i>	<i>84</i>
<i>Testing.....</i>	<i>84</i>
<i>Testing against Features of Solution.....</i>	<i>91</i>
<i>Questionnaire.....</i>	<i>93</i>
<i>Maintenance and Limitations .....</i>	<i>94</i>

## Analysis

### Project Outliner

I will be making a platformer, 2d single-player adventure game in which you will be playing as the villain of the story. This was inspired by games such as

### Problem Identification

At this point in time, a lot of games start to blend in with each other, especially on mobile devices where the gaming market has merged. There is less innovation and thought put into these games. The groups go as followed:

- The new boring arcade/indie style games, company Ketchapp is notorious when it comes to that, bringing titles that are fun for a few weeks and then die off, get filled with ads and such. There are no good updates, or a goal to drive the player, just getting the highest score. There is not even a bit of storytelling.
- Then there are the “og” titles. Games that have been around and will be around for a long time. Titles such as Candy Crush, Subway Surfers, Hay Day, 8 Ball Pool, all games that have developed overtime and kept their popularity with updates and new audiences.

Currently, the Google Store is filled with rip offs and rushed titles. Overcomplicated or too simplistic and it takes away from the gaming experience and entertainment the user should feel, even for a free game.

Most of these games require little processing power but considering it is a phone, some can take up more than expected (PUBG Mobile). Due to the chips having integrated graphics, the lack of a GPU means that some games that require more graphical processing power are unable to run. This is a restrictions game developer have, but easier for me to work around it. A lot of great games need very little processing power, e.g., Minecraft, League of Legends, Terraria, Stardew Valley, etc. but they all make up for it in the gameplay. Keeping it simple is what worked, even if these games started on PC.

There also aren't many platformer games done in pixel art. The great decline in pixel/retro art is partly due to the great ride in HD and Full HD 3D rendered games. Pixel art was born due to device limitations, creating chunky pixels and limited colour pallet gave birth to this genre. A lot of games and series came from it that kept a simplicity but not many still do.

The niche market and lack of investments dried out projects resulting into no new pixel art games being put out, and forcing people to keep replaying older titles. The genre is versatile and hated by no one. With a large community behind it, finding sprites, ideas and animations for a game can be easy, but wanting something specific can prove difficult.

### Problem Decomposition

There are a few issues that come with making a game that also has a story.

1. The story has to be designed and layed out before hand, as well as the characters stories, goals, ambitions, motives etc.

2. Making animations, sound effects and using a program that is capable of using all these things.
3. Making a menu system that also has settings for the usual things such as brightness, sound etc.
4. Making a level system for the player to progress through
5. Making a levelling up system, maybe even different abilities and tools
6. Making the enemies or AI if needed
7. Verification to make sure that the user is fit to play, for example age confirmation, email, etc.

### Stakeholders

The client and demographic for this game would be mobile phone users/pc users. Because the game will be so broad, and aimed at various age groups across multiple platforms, it is important to make the game very intuitive and easy to pick up right away without any issues, without making the game feel boring.

The gaming market is huge, but what I will be aiming at is casual gaming seeing as the game has to be welcoming to newcomers but also fun for more experienced players, and also easier to build.

In order to reach more people the game has to:

- Be fun to play as it is to watch
- Cheap but reasonably priced
- Welcoming and intuitive
- Meant for all ages

This should help the game get popularity among the clients but also viewers who just want to see the game being played by someone else on YouTube.

One problem people might encounter are different errors due to running the game on different computers. This is one of the reasons I have to use a game engine, in order to ensure that the game can be made more efficiently.

## Computational Methods

### Abstraction and visualization

- Key objects: Main character, enemies, backgrounds, orbs/pickups, animations
- Will be needing sprites for all the characters, 'buildings', background etc. but also fonts (most likely to be Amhurst SF) as well as icons for pickups, and power-ups.

### Thinking Abstractly

My game needs essential features to make the gameplay lighter and give enough mobility for the player to navigate the level in their own way. Some of the essential things include:

- Pause feature, allowing the player to take breaks or just be able to stop the action for a bit in order to go in the kitchen or there is something they quickly need to do. From the pause menu they also have access to certain more features such as a settings menu. This pause feature is a must to include.
- Character and Enemy Animations to bring fluidity to the game, which will be created using sprites, making the game look more appealing and smoother.
- Score counter which will determine what award they get at the end of the level. This is not an essential feature but there should be a driving reason why the player would try to replay the level, and this could be, in order to increase their score.
- Health/Power up bars which are going to keep the player aware of the losing possibility/ factor. This is making the game more intense, knowing you are not fully invincible, and even though you have a powerful ability, you cannot use it 24/7.

### Thinking Ahead

- Controls should be intuitive and easy to use. Usually the controls are similar to most games and so they would be AWDS for moving the character, and mouse for any strikes the character might make. Space/Shift for any abilities the character might have and E, Q or F for any world/items the character might interact with. This will make it easy for gamers to get used to as they are very common in most games but also for new player as the keys are easy to reach and learn.
- Settings should also have a few options that the player has access to such as volume, some display settings and so on. Controls could be one of them but it is not very essential.
- Damage system should be balanced towards the player. The player should be way stronger than the enemy but the enemy balances out in the numbers as there will be multiple enemies the player should fight against. Player health is 100 and so is enemy. Player could do 20-80 damage per strike whereas enemy can do around 5-15.
- Scoring system is simple. Every enemy destroyed scores a certain amount of points, and even more points if you don't take much damage or you don't die yourself in the process.

### Thinking Procedurally

Although some of these steps are less important and therefore might be sacrificed depending on the time limit, but that is the target, the aim.

The task is split into multiple tasks:

- UI of the game including the menus, pause menu, settings, info etc. These are all the menus the player can go in, each of those being one scene by themselves in unity.
- Levels of the game which are also going to be structured using sprites. The game will not have many levels but just enough so that the player can explore.
- Characters including the player and enemies. These will use sprites in order to animate movement but also they will have their own characteristics. The NPCs should also have some sort of vision so that they actual attack the player when they see it.
- Scores which will be calculated by how well the player has done during the level, and will award the player one of three awards according to their score on that level.
- Saving the game is also key. The essential part is saving the score on the levels already done so that progress is not lost. These can be stored in a separate file and showed to the player when the game is opened.

These tasks will be coded separately and added to the backbone of the game which the player will not get to see any of. This will allow the different scenes to interact making the experience more fluid as they go from Main Menu to Play to Gameplay to Pause

### Thinking Logically

- The Main Menu gives the user the option to go Play, Options, and Info. This means that the game state will be looped on the Main Menu until a decision is made by the player
- When player is hit or hits someone else, the health should decrease accordingly, and score should increase as the player takes down more enemies, and decrease as the player takes more damage.
- The game level will run until the finish is reached or until all the enemies are taken down. The progress of that level will then be saved for when the player comes and replays the level.
- Player can exit game from pause menu, so pause menu will loop until player makes a decision of Resume, Settings or Exit.

### Thinking Concurrently

- The enemy can detect the player but can also be killed before they do so, as there is a delay allowing the player to take the first enemy without other enemies being alerted. The game is very alive and responds to every move the player makes, as the program runs different processes simultaneously which increases the immersiveness and smoothness of the gaming experience.

## Interview

The point of the interview is to get some primary data and information that can help me develop the game in a way that people will enjoy playing it.

1. What is your favourite game genre/game
2. Would you consider yourself a gamer?
3. Are you into platformer titles such as Mario, Sonic, Celeste etc.?
4. What device do you play games on most of the time?
5. How much time a week do you spend playing games?
6. How do you purchase the games you play?

Henry

1. "It has to be first person shooters such as Ironsight, and if not that, then story telling games"
2. "Yes, I would say so"
3. "I find platformers very repetitive and boring, however I cannot say no to Mario or Celeste"
4. "I play on my PC for most of the time, or phone but rarely"
5. "I spend around 10 hours a week gaming, more or less depending on the week"
6. "I prefer to play free games, but if I want to buy a game, I do it online all the time."

Analysis: We can see Henry as a potential customer. First person shooters can definitely be chaotic and really fast paced sometimes so I can see him playing as a bad guy in my game. We can also see that he plays a considerable amount of games, but we also don't want to waste his time so the game has to feel fast as well when it comes to storytelling, less than 12 hours of gameplay.

He is also a PC user, which is the biggest platform out there, and it is the platform we are aiming for mainly, of course the game can be made for multiple platforms and bring it to them later on.

Danika

1. "I only play games at a friend's house or on my phone in the evenings"
2. "No, not really"
3. "Mostly puzzle games, with levels and such"
4. "On my phone or a console my friends might have"
5. "At maximum around 2-3 hours"



6. "I don't buy games at all, not for me at least. I do gift them sometimes"

Analysis: Now she is less likely to be a potential client because she is not a very big gamer and also because she probably doesn't see PC's as a gaming machine. However, she does play games on her phone and even if she might not be interested at first, she might hear from a friend about a game that she might want to play.

She also buys games for other people as gifts, so if she is still a potential customer, just not a potential user.

Andrea

1. "Loot and shoot type games. That or horror, but I can appreciate all games"
2. "Most definitely"
3. "I was for a long time but they all became repetitive so I stopped playing them"
4. "Everything from PS4 to Mobile Devices but I spend most of my time on PC"
5. "More than 25 hours, including streaming the games online on platforms such as Twitch or YouTube"
6. "I don't mind paying for games if the game is worth the money, but I rarely pre-order a game"

Analysis: In this scenario we can see the potential client takes gaming less like a hobby and more like a full-time job as she streams games and plays a lot. Having people like this play your game is great as it increases the game's popularity. She has access to all the different platforms but PC is the one she prefers. So for her, the game has to be entertaining to play but also to watch for her Streams.

Overall: We see that our audience can and will be very wide, which means the game has to be entertaining to play and to watch so that it attracts more players and also viewers making the game more popular in return. But the focus is being put on originality and uniqueness for this game to be made for the PC platform to start with.

! It is also important to explain the games inner mechanics to the player for people like Danika who have little experience playing games but also allow them to skip it if they want.

## Research

There are a lot of games out there that fit the following criteria:

Casual game with mediocre graphics that also has some basic story telling aspect that drives the game but can also be developed into an arcade, multiplayer etc. It has multiple levels, maybe even bosses.

Mario would be a good example, but more recent title is Fancy Pants.



The stick man character is simple having smooth animations all around, and the story telling aspect is simple but it drives the story, the same way Mario is driven by the kidnapping of Princess Peach, for the last almost 40 years of the series, with small variations in the story, but it keeps players coming for more, as the game mechanics and gameplay are fun, even if the visuals weren't that pleasing in the beginning of the genre.

This game has done well and created a good fan base, made multiple parts but ultimately died off. Out of all the stickman games over the years, this one is one of the best. It was fun, fast paced, and just a casual game with extremely fluid gameplay, even for a platformer.

Although this was a successful series, there have been more, less successful platformer titles.

One of which is Super Neighbour World. The controls of the game are very stiff and it lacks originality as it borrows aspects from different games such as Mario and Hello Neighbour inspiration. It has no real drive for the player to keep going.



But there have also been extremely innovative platformers, one for example being FEZ which brings the 3d aspect to the game in order to solve puzzles. A platformer in order to stand out, must have some unique aspect to it which separates it from any other platformer/casual game. This can be the 3d/isometric aspect or just simple but smooth animations for the characters. Having an aspect like this is a must as people will then associate it with your game. This can be part of the story telling as it is a must to attract players to it.

FEZ sold over 1 million copies by the end of 2013 and inspired games such as Monument Valley, Crossy Road and Secrets of Rætikon.



FEZ – Puzzle solving 2d game with a twist. The levels are based around a tower like structure/city. Although it is in a 2d perspective, you can go around the tower giving it different perspectives of the same tower and requires you to go up.

Due to its cool factor FEZ was praised by the gaming community and was relatively successful. It made it unique and special, it made it stand out from other 2d games.

For my game, this aspect might be, playing as the “bad guy” for once. There have been games who have done this but not enough. Some of these games are considered classic and this aspect of the gaming industry has been lost.

Games such as Destroying All Humans, Prototype, and GTA Series etc.

All these games are fun, partly due to gameplay, but partly due to the freedom the game gives you. It moves you towards being a villain which is something most games don’t do.

Keeping the audience hooked is the main goal, so choosing and developing a character in an interesting way is crucial to the game’s dynamics and story line.

The main idea was having the main character being a version of the Grim Reaper, and have him do missions on earth in which he has to bring people to the underworld in whatever fashion he wants. Now we have some playing room around this idea. We can make the Grim Reaper, a considered bad guy, and slowly develop him into a good character. We can also prove his immense power and let the player have all the fun.

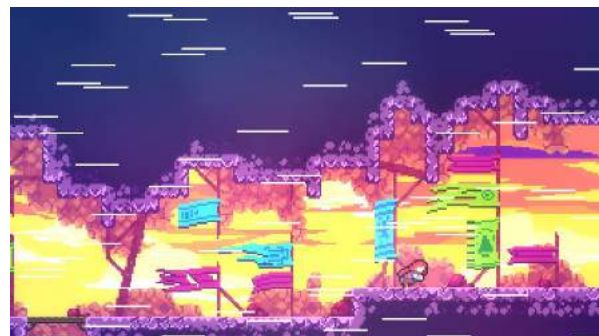
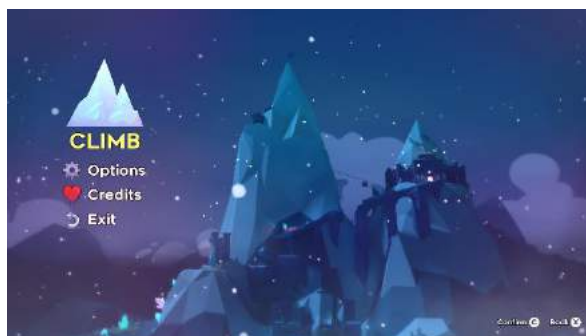
Of course around the same mechanic we can have a caveman for example, that was brought into the future by mistake. This leads onto you destroying things for fun to try and find a way home, although you are stuck 10,000 years into the future. This also gives you as the player, an excuse for destroying bridges, cities, killing humans and so on. It would be fun if you could go back in time each level throughout history until you get back to your family and friends.

Both of these ideas are similar when it comes to game play. But looking at upgradability, being the grim reaper can also bring an arcade mode to the game and also can be way more fun and unique.

### Similar Games Analysis

Celeste (The Platformer): This little pixelated adventure 2d game has been very praised by many players. It is a fun and challenging platformer which is part of the inspiration for the art style I will be using, which is pixel art. This game is simple, yet unique, with some very cool features. Celeste looks amazing and feels great playing it with smooth gameplay and harsh penalties. It is definitely an example for what a platformer should look like and play like. Celeste is also played on almost every platform but it doesn't lose its core attribute of being a simple easy to pick up and play game. Some new comers to gaming in general might be confused as to what the game wants from them but learning as the game progresses is crucial, which is what drives them to keep playing.

The main menu is simple and to the point. It has the necessary options, credits, exit and the main play button or "climb". This simplicity is seen through the rest of the game and is a source of inspiration when it comes to the design of the menu and some parts of the game.



GTA V (The villain narrative): This game is one of the best ever with a story mode that is still talked about. Even earlier in the series, playing as a normal pedestrian was never an option. Being chaotic, unpredictable and a real villain type character was the best choice. Rockstar have done that very well with the character Trevor. A psychotic 40 something man that wants nothing but revenge and some "good ol' fun". He is weird and ridiculous but at the same time, he feels real, which is what made him such a popular character. When it comes to bad guys, Trevor is a perfect example, not just for games but even for movies and TV series. Showcasing his craziness early, and keeps adding to the unnecessary violence and paybacks makes Trevor a fun character. Being able to replicate that with a twist is the aim for my game.

In order to make a believable villain, like Trevor, the character has to prove itself. Maybe put an event at the beginning of the game or a bit after the first chapter that explains us the villain's perspective and why they are the way they are. This is what I will use in the story telling, a progressive character development to build a relationship between the player and the character.



### Features of Solution

Feature	Explanation/Limitations
Pixel art graphics	This is the type of game I am aiming for as I see this as being a gap in the market that I want to explore further, and the type of game that people don't have enough of currently.
Relevant sound effects	The sound of the game should reflect the atmosphere of the game. It should be a somewhat light game but still have edgy/darker colours and so the sound should be able to reflect that.
Clear instructions and help	Some players might be somewhat new to games so the game should help them with information on how to play it. The game should also be very intuitive for more experienced players.
The game is single player	This genre is getting somewhat of an overlook nowadays, and so making the game single player is not only more fit for my skill set, but also makes the game more unique and original compared to other games.
The menus are easy to navigate	The game has a certain feeling to it and so the menus should make the whole experience much more immersive. The menus are not a focus, unlike loot n shoot type games. Players do not want to spend much time in menus.
There is only one possible character to play with	This is because the game focuses on gameplay, although being fairly arcade oriented, there is one base character that I will implement due to avoiding any added complexity.
Relevant background and UI elements	This should make the game feel more polished and relevant, as it fits better with the rest of the game elements.
An end screen after player finishes level or dies	This should tell the player how well or how badly they have played the game. It should have information about their performance and reward them accordingly.



### Initial concept of my character

The concept I will go for is a platformer, pixelated 2d adventure game where you play as the Grim Reaper on Earth. The aim is to find your target human and take them to the underworld. This allows you to play as the bad guy around different levels and do whatever in order to get to your destination.

The character development will just be the grim reaper travelling around the earth, becoming more human than ever before, chasing his 'target'.

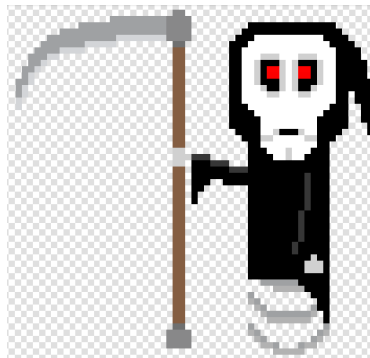
The character would be brought to life using animation and sprites in order to simulate movement during the game play, allowing the player to control this characters movement which is a big part of the development as well as animating the enemies, making sprites for the menu system as well as background and tile sets.

### Art Inspiration

The Grim Reaper Inspiration



Pixel art version 1



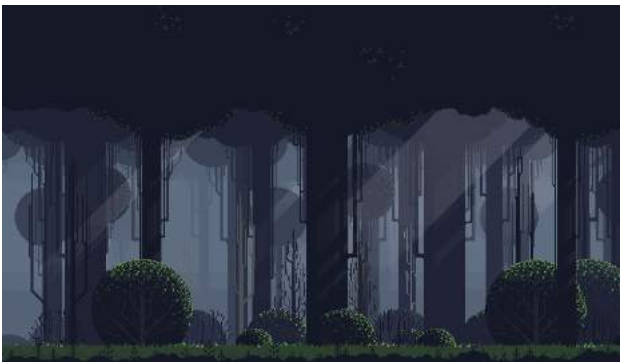
Pixel art version 2



### Backgrounds

Menu background is this dark pixelated image on a Japanese town. It fits the setting of the game having a dark style and yet it is in a more HD side of pixel art having very realistic shapes and figures.

The game will not look like this but this feel like a good way to welcome the player to the game.



This background could change over the course of development depending on my needs. But this would be the main idea, a dark forest or a ruined town type setting for the majority of the gameplay.

## Game States

**Menu:** This is the first part players get to interact with and see. The player has a few options they can choose from. They include "PLAY", which leads them to the actual game, "OPTIONS" for the game itself with some things they can change within the game, "INFO" which gives them some information about controls used in the game and how the game works, and the last one being "QUIT", which exits the game completely.

**Paused:** This is when the player chooses to press "ESC" to pause the game as they will not be at their desk to play it for a bit. The game should freeze completely, as it only unfreezes when the player comes back to the game.

**Playing:** This game state is when the player actually plays the game. They should have some UI at the corners of the screens to give them an idea of what is going such as a health bar, or a powerup bar or even hints/directions.

**Over:** This is the end screen the player sees when they have finished the level or died trying to finish it. It should have some data showing the player how well they did during that level and what they can improve on things such as time taken to finish, damage taken, damage dealt, enemies killed, lives used, overall score or maybe even a medal ranking from bronze to gold/stars to signify how well the player did.

## Controls

Controls for main character:

"A" moves towards the left, "D" moves towards the right

"W" makes the character jump or go up, "S" makes the character crouch or go down

"SHIFT" makes the character use their special ability which is the dash ability.

"SPACE BAR" activates the chargeable power the player has which does more damage to enemies.

"E" is the button used to interact with world objects and orbs.

"Left Mouse Button" is used to strike or slash with the scythe.

"Right Mouse Button" is used to aim your scythe and prepare for a throw

The mouse movement allows the player to slash or throw in any direction they so wish.

"ESC" is what leads them to the pause game state and can be activated anytime during the gameplay.

### Limitations to my solution

The biggest limitation is my own skill. Having little skill when it comes to game development, this will be a challenge for me and a way to learn more about the topic. But this also means that there are things which are harder for me to implement which might lead to a lesser game than the one I had hoped for.

Another limitation is time. Having a time limit might mean I have to sacrifice parts of the game for others, which might make the game feel rushed and have gaps within it.

### Requirements

#### Hardware

- A computer capable of running the game at a good quality and frame rate, that
- Input system, preferably a keyboard and mouse – this can be changed to a controller but that will have to be implemented in the settings section.`

#### Software

- Windows operating system at 64-bit – This supports almost any game and also supports unity
- For the pixel art I will use a program called Sprite
- Any extra things required by unity before it can start your game
- I will be making the game in Unity 2020.1.5

### Development Language/IDE

The language I will have to use is C# as that is the most common language used in Unity for game development. An alternative would've been from the C family, C++ but that would be when using Unreal Engine to develop the game, but Unity is better overall for 2D game development.

Unity doesn't have a code editor but it is closely linked with visual studio, which is the code editor I will use to code and Unity to sharpen the game and make it look like one.



## Stakeholder requirements

### Design

Requirements	Explanation
A simple main menu that is intuitive and easy to use for new comers but also has enough settings for more experienced players	This will allow both low and high skilled players to feel right at home
The design scheme reflects the story and the main character being the Grim Reaper.	This give the game some personality to be associated with, e.g. Minecraft has a 3d square as a logo that resembles dirt and grass.
The whole theme should be dark along with the writing in the menus, backgrounds, sounds, etc.	This goes along nicely with the meaning and feeling of the whole game, which is to be expected.

### Functionality

Requirement	Explanation
The usual game setting that can be tampered with so that every user can personalize their experience to a certain extent. Includes controls, graphics, gameplay etc.	This will allow higher skilled users to personalize their experience the way they want it, but it is also good for people with more special screen and so on to run the game optimally.
The game should also have a pause menu that allows you to change some of the in game settings and maybe even a tip section for those that are lost and don't know what to do	This will allow the users to personalize the game experience but also stop their progress for a little break if they so wish.
The ability to change key binds would be a good thing to add	This can mean that users can customize their experience even more and could allow room for connecting a controller later on.
A save button, although the game will save automatically, having the option to do so manually can prove to be beneficial and some people prefer it.	This will allow users to ensure that their progress is saved after they are done playing for the day.
Having a cheat menu or just cheats to add some more fun to the game.	After the user has finished the game they can replay it using cheats which might just make it more fun and hilarious.

### Hardware and Software

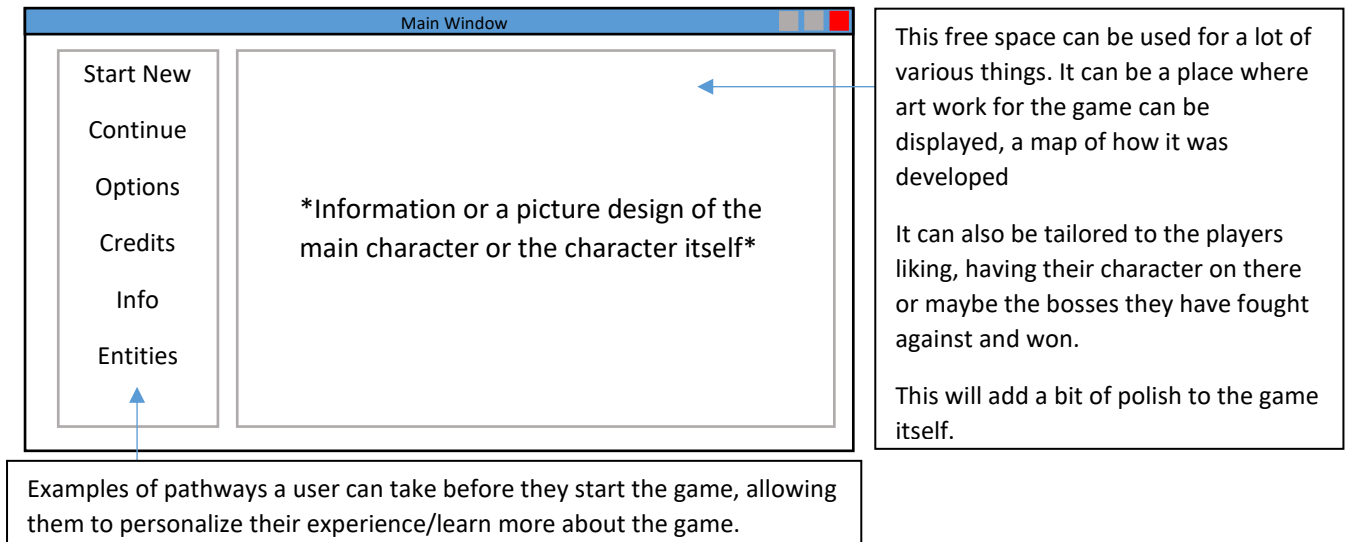
Requirement	Explanation
Standard computer peripherals: Computer with a keyboard mouse and monitor.	The user needs basic peripherals in order to play the game and to progress through it.
Preferably headphones or speakers for the sound effects, although it shouldn't affect the user performance a lot.	This is not a must but it is recommended in order to get the full game experience.
Minimum Computer Specs: (enough to run the software)	The computer needs to be able to run the game. Although not a lot of image processing is needed, some still is.
Windows, Mac or Linux operating systems	In order to run the program and to store the data, you need an operating system.

## Design

### Basic structure of the solution

The game will be comprised of a main menu, from which the player can access the game, settings, information about the game as well as quit the game or access saved files.

### User interface design (Menu)



### User interface design (Game)



This is where the main gameplay will take place, including the combat and character actions. It must not be restricted by the other gameplay U.I. so that the player can see clearly what is going on

There is also an objective at the bottom of the screen so that the player knows where to go and what to do

There is a combo counter for the player which gives him more damage every x5 which can encourage the player to increase it and use it to their advantage.

The health. Power up and special ability bar are all in the same place making it easier for the player to just glance and know everything about their chargeable. On the top of the screen it also shows the new location you are in which just adds to the atmosphere of the game. Enemies will also have their own health bar and the damage taken by each enemy.

### Decomposition of the problem

In order to meet my goals, the game will be developed in two stages. The first stage is building a character, hit box, movement, collision mechanics etc. Also making the enemies, and making the player be able to interact with the enemies. Also building some sort of mediocre A.I. so that the enemies are not just clueless. After all these assets are built, they will be used in different levels.

Of course, these problems can be broken even further, which is why I used object-oriented design to do so.

### Object-oriented design/programming

Assets	Description	Mechanics
Player	This is the character that the player will control, and help to progress through the story.	Hit box, Movement, Inter body interactions Dialogue, Health bar, Power bar
Enemy	These are characters whose job is to stop the player from progressing.	Hit box, Movement, Inter body interactions, Some dialogue/sounds, Health bars for bosses
Level Floor/ Objects	This is the space the player gets to explore in order to progress.	Collision box, design changes
Weapon/ Scythe	The weapon the player will use to damage enemies.	Away from character wmechanics, retractable.
NPC's	There will be characters along the story line that the player can interact with	Dialogue and interactions/tasks
Orbs	Collectibles the player can interact with that give him health, or fill his powers.	Interaction with player.
Backgrounds	Each level has to have a background, but also far objects move from right to left slower than the things that are closer to you.	It reacts to player movement accordingly, meaning foreground moves faster and background moves slower.

### A justification of the approach taken

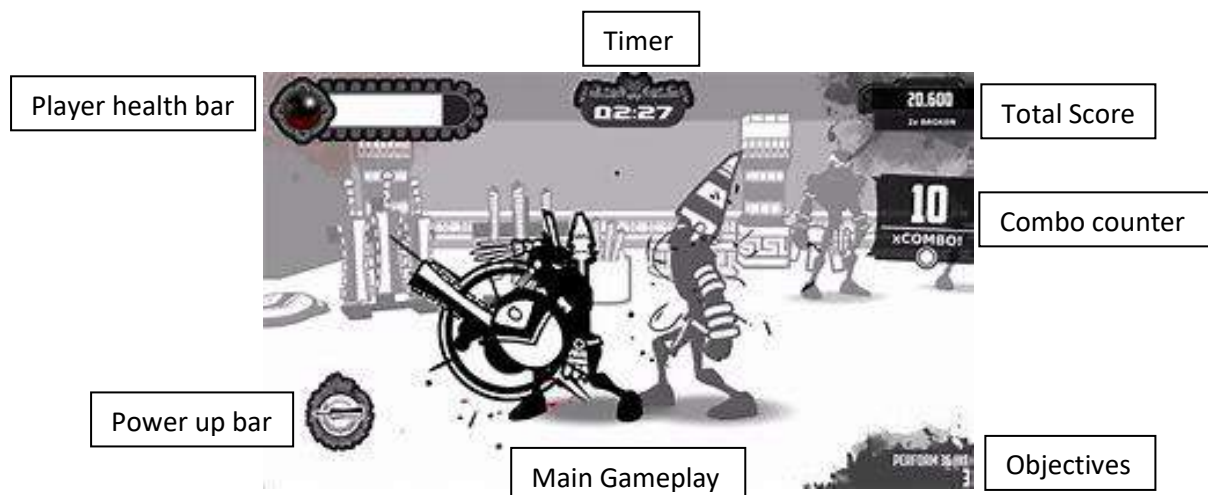
The problem will be approached in this way because it is the easier way to do so using almost any game making and also allows me to reuse all the assets from level to level, or even different game modes.

It will also enable me to reach my goals easier, those being:

- A high and constant frame rate even for a not so equipped computer achieved through efficient processing
- Good graphics, with smooth animations for the characters
- Sounds to create a more immersive experience for the player
- Level design to create a constant progression through the story
- Develop it in an easy way to add new things later on
- Easier to understand and breakdown the game later if needed

The approach is not definitive, as it can change as development takes place, and so iterative development can change the course of the games development.

### During the Gameplay U. I. Example



- As we can see from a gameplay screenshot, the game is very combat focused, so some things such as a combo counter and timer may not be needed, but they are useful to point out.
- Of course, the player has a health bar, which will have to be implemented in my game. This keeps the player on their toes and so the game can punish the player for making mistakes, and drive the player to become better at the game.
- The powerup bar is also something that can and should be implemented as it adds an extra layer to the game, allowing the player to learn the game mechanics and see the powerups for his/her advantage.
- Objectives are something that need to be made clear, either through the story or just pointing them out, as otherwise, it leaves the player clueless, which is sometimes an issue with a lot of game.
- Total score is something I can implement as Score per level. Score will be linked to how much damage you took, and how much damage you dealt. Score can also be affected by the time it took you to finish a level/tasks

The use of those functionality features are the same old basics from game to game, which is a fairly good thing. You can expect players to have some knowledge of what is going on in the game.

It is intuitive for a 2d game, even with no prompts, to go towards the right side of the screen. It is normal for the controls to be AWDS or arrows, or for the jump to be something like space.

This is one of the reasons many games spend very little time on introduction of the mechanics and controls, because they are very universal. Because games use this universal language, games that deviate from it are harder for people to get into or to enjoy playing.

People enjoy seeing new takes on the same design of U.I. which also makes the gameplay more intuitive for most users but also easy to get used to for new gamers.

## Default Game Controls



AWSD (Blue) – Controls are used for moving the character on screen. These buttons are used in most pc games as they are well positioned for the user to use their left hand while the right hand is

free to use the mouse. These controls can move the character left, up, down or right respectively.

E/Left Shift (Red) – These buttons can be used for the special ability/dash ability for the character. They are close to the main movement control key so they are easy to reach and use without much movement in the left hand.

F (Yellow) – This button will be used for any world/item interaction such as orb collecting etc. this is also an easy button to reach and use but due to preferences, it might be the most frequent button to change in the settings by the user due to some confusion games created between E and F.

Esc (Purple) – This button will be used to pause the game or go back to the previous window in the menus. This button is in the corner of the keyboard which is good so that you don't



Left Mouse Button (Blue) – This is the button used to 'strike'. The user would use their right hand to use these two buttons as the mouse controls the combat of the game. The strike would do damage to the enemy and the player can hit repeatedly for more hits. This button is also used as a main button for combat in other games to control things such as 'shoot'.

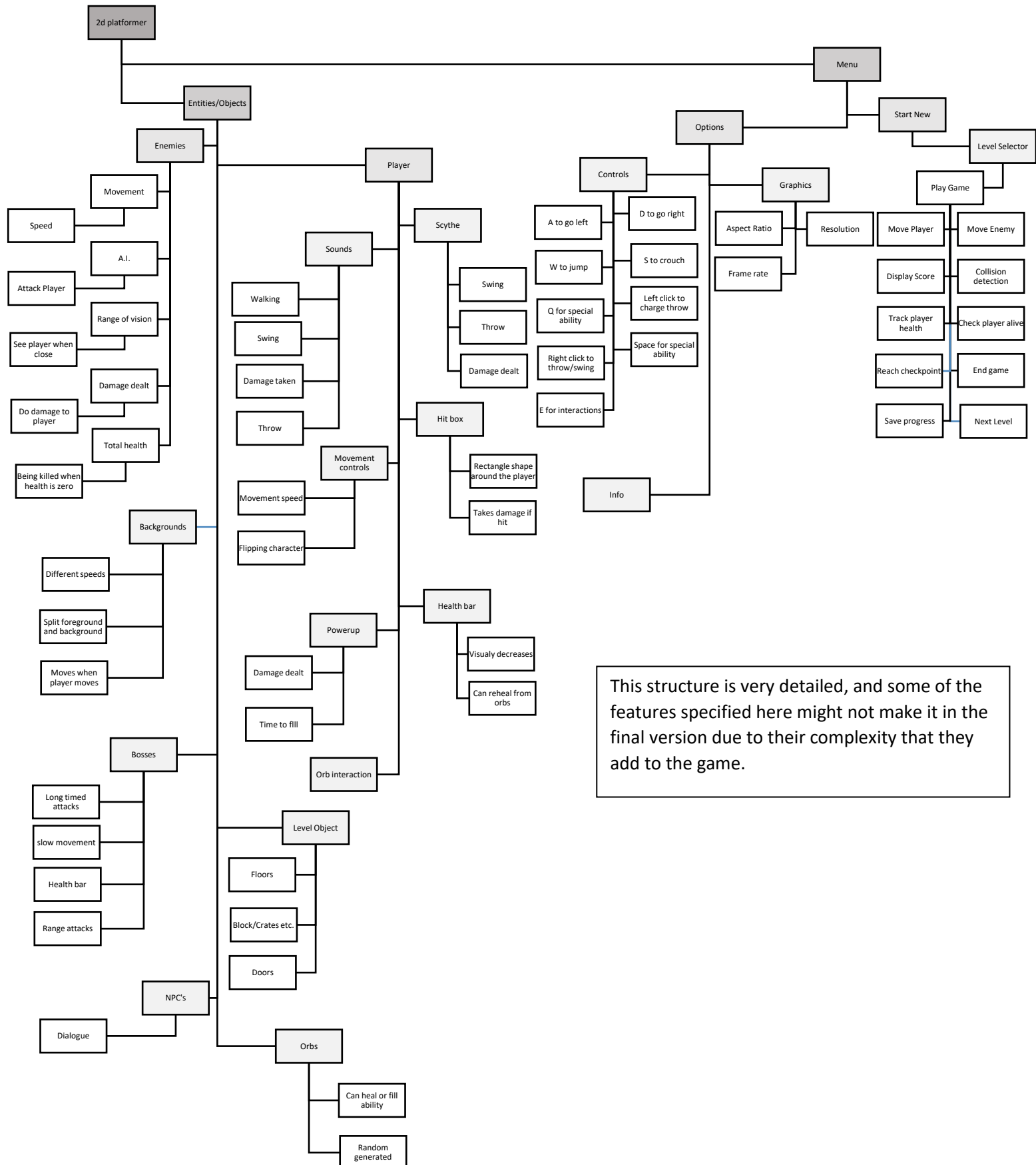
Right Mouse Button (Red) – Holding down this button gives you the option to aim your scythe, as the character gets ready for a throw.

Movement of Mouse (Green) – This allows you to aim at what enemies you want to swing towards while pressing left mouse button. It also allows you to aim the scythe throw when holding down on the right mouse button.

Right Mouse Button (Red) + Left Mouse Button (Blue) – Once aimed and with the right mouse button being held, pressing the left mouse button will throw the axe towards the direction aimed at. If it hits any enemies, it will do damage to them.

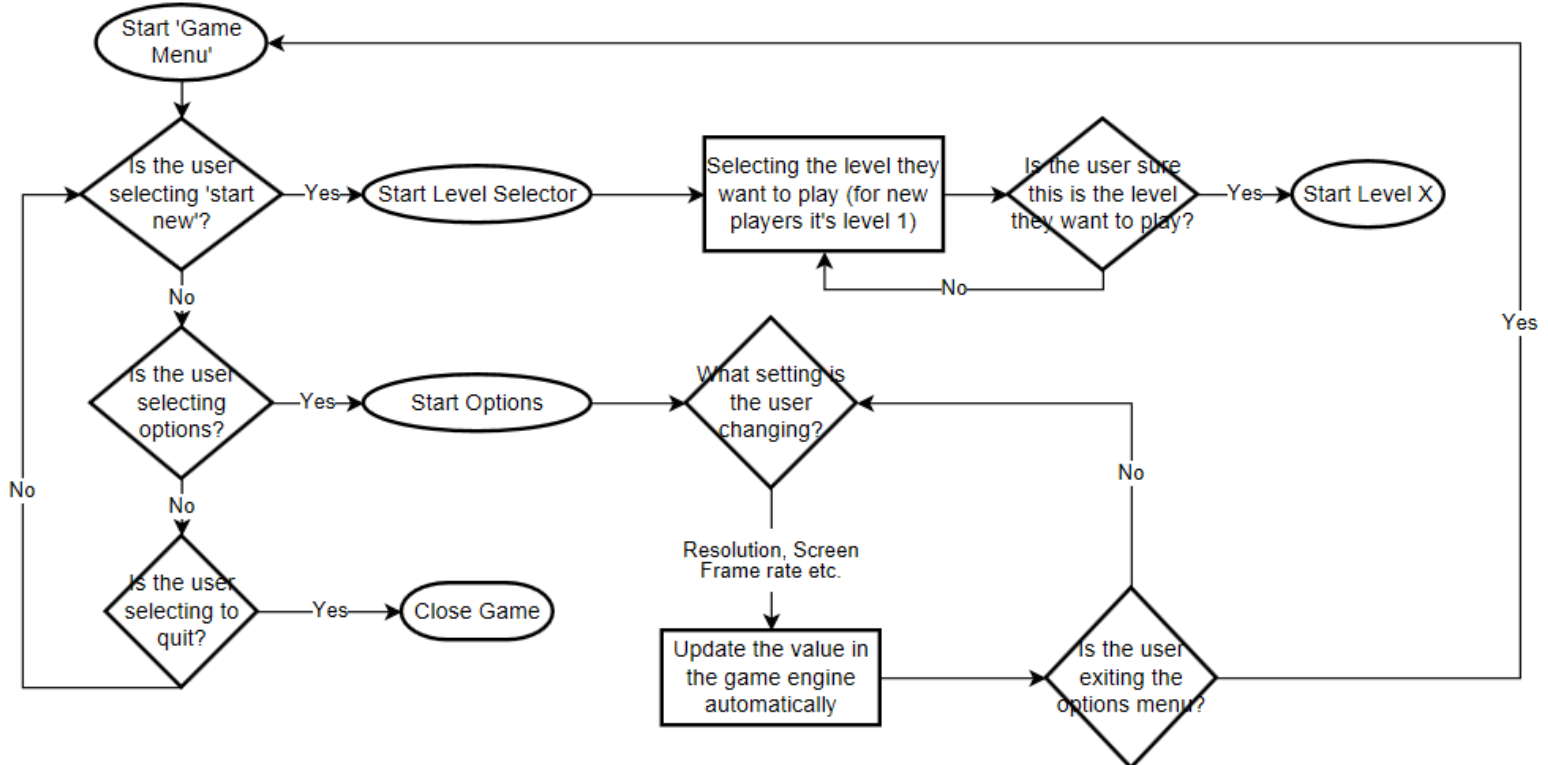
This is a way to include ranged combat in to the game, adding another layer to the combat making the game more fun allowing players to play in their own way.

## Structure of the Solution



## Flowchart of solution

### Menu Flowchart



The menu is fairly simple as I will not focus too much on it, but it should give the user some control over their experience and should also be simple enough for someone new to games to be able to navigate easily with little to no problems.

The menu's look will be determined by a simple background as well as button animations that gives the game a polished look and feel.



### Character movement

When the level starts the character spawns in stationary and will remain stationary until the user takes control of the character.

The controls will change the position of the character accordingly.

A – makes the character move to the left for as long as it is pressed with a certain fixed speed

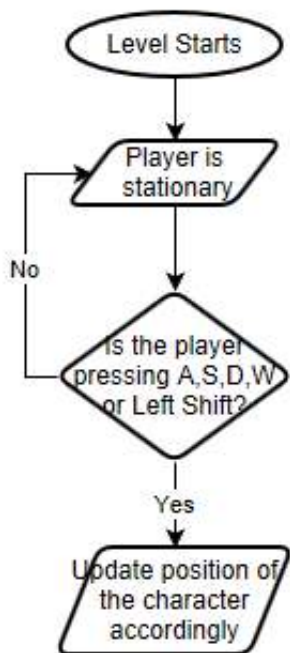
S – makes the character go down or crouch in certain areas

D – moves the character to the right of the screen for as long as it is pressed with a certain fixed speed

Left Shift – Special dash ability that the player can use but has a cool down so keeping it pressed won't do anything.

The position of the character is then updated every single frame, taking into account things such as gravity so that when pressing W the character jumps and doesn't fly.

Flowchart is simplified because unity helps when developing



### Character Combat

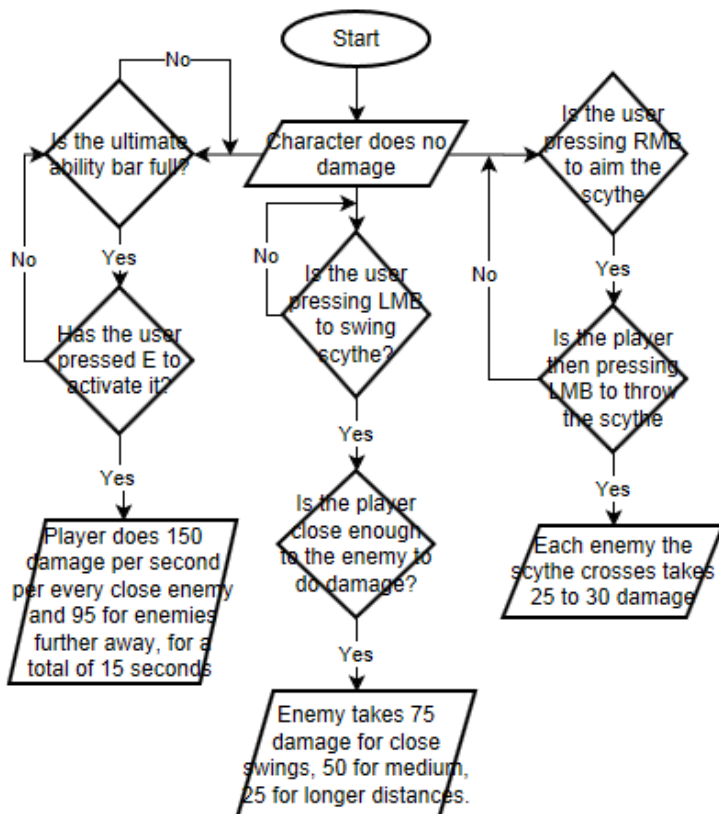
This flowchart takes into consideration the distance between the main character and enemies so when swinging the scythe, it does damage accordingly.

This also gives the user a reason to stay close to the enemy, but it comes at the costs of them doing damage to you.

The only advantage the user has is that they can do way more damage per second as the swings are quick and swift whereas the enemies are slower, especially the bosses.

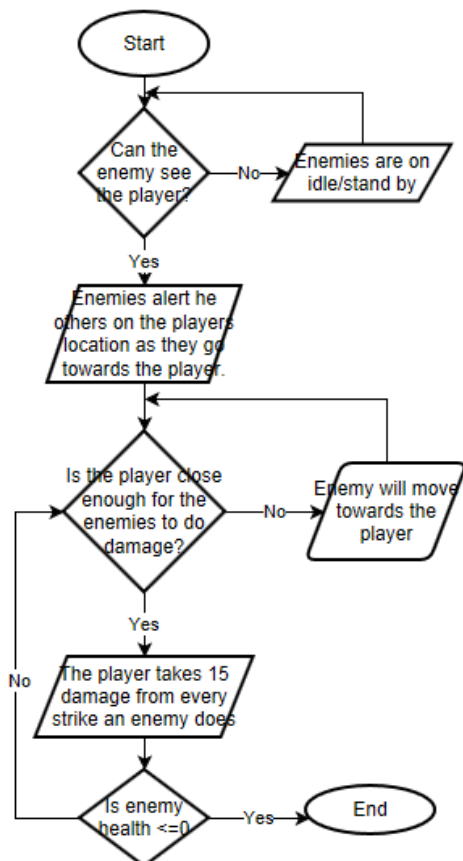
Also, the scythe can do damage when thrown and retracted, 25 every time it passes an enemy.

The ultimate ability takes a long time to recharge so you may use it maybe once a level, but does loads of damage to enemies.





### Enemy A.I. Structure



The A.I. for the enemies is kept simple. When the enemy sees the player, they confirm that it is in fact the player, so they might take about 0.3 seconds to decide and then start attacking the player.

They also alert other enemies to rush towards the player and stop him from progressing through to the next level.

The enemy has a smaller range of vision than the player, which allows the player to see the enemy before the enemy sees the player which allows for stealth kills and planning.

As soon as the player comes within that range of vision, the enemy will then decide on the player and then attack whilst alerting the other enemies.

The enemy also needs to know if the player is close enough to them for them to start striking. If the player is close enough, then the enemy does 15 damage points per second from each enemy.

Enemy will also die if their health is below or equal to zero.

### Test Data for Development

Identification and justification of data to be used during development is tested for functionality before moving onto the next stage. Each of the next few characteristics of the game play an important role in making the game whole and immersive, as well as polished and glitch free for the players experience.

#### Movement

What is being tested	Data Type
Player movement	Valid
Movement Speed	Valid
Dash Ability	Valid
Enemies move when they see you and they don't when you are far.	Valid/Invalid
You can move even when ultimate is activated.	Valid
You cannot move through any world object	Invalid
You should be able to move around or even dash past certain enemies.	Valid
Controls for movement should be as specified	Valid

#### Combat

What is being tested	Data Input
Swing scythe	Valid
Aim to throw	Valid
Throw Scythe	Valid
Retract Scythe	Valid
Enemies damage player	Valid
Special ability	Valid
Camera shake	Valid
Sound effects	Valid
Combo perks	Valid
Damage doesn't happen if there is no contact between the characters.	Invalid
Cannot use power when it's only 99.9%	Borderline
The dash ability cannot be used when the cool down happens	Invalid
Controls for combat should be as specified	Valid

#### Interaction

What is being tested	Data Input
Collecting an orb	Valid
Soul hunger	
Talking to random strangers along the way	Press Q

#### Enemies

What is being tested	Data Input
----------------------	------------

Enemies run towards you when they see you	Valid
Enemies should take 100 damage maximum in order to die	Valid/Borderline
Enemies can alert others when you are near	Valid
Enemies can do damage to you only when you are close	Valid

#### Menu

What is being tested	Data Input
Can you start a new game by clicking play	Valid
Can you continue a game from where you left off	Valid
Accessing the setting menu	Valid
Changing some of the settings and does it actually make a difference	Valid
Does the quit button actually work by quitting the game.	Valid
Can the game be pause mid-gameplay by pressing "esc"?	Valid
Is the game frozen while pause menu is up?	Valid
Can you exit back to main menu from pause menu?	Valid

#### Levels

What is being tested	Data Input
Is the level re-playable?	Valid
Does the level show the score you got when finished?	Valid
Is the level re-spawning you in the right place?	Valid
The level should stay consistent even if changes are made in the settings	Valid

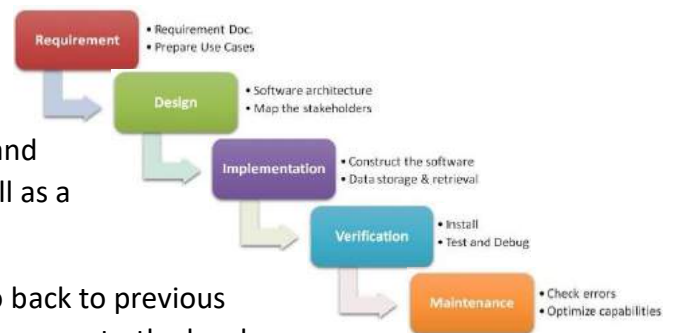
!Any glitches that the game might have during development and testing, e.g. falling through the ground or one hit deaths or even glitches that are an advantage to the player will be noted in the development or testing stage.

## Development

### SDLC Process

#### Waterfall Methodology

The waterfall methodology is very structured, following a sequential design process to the development of the system. It is also a very easy model to manage, and it would work for the small size of my project, as well as a faster delivery of the project.

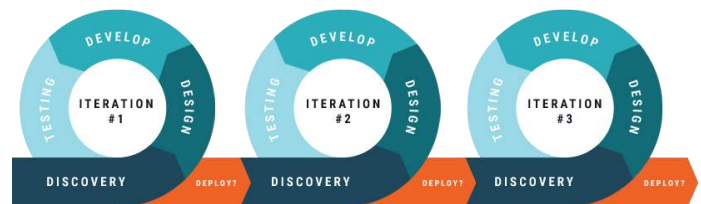


But using this methodology will make it harder to go back to previous stages which I might have to do in order to polish the game to the level that I want it to be.

#### Agile methodology

The agile methodology is a practice that helps continuous iteration of development and testing, as these activities are concurrent, unlike waterfall model. Using this methodology should lead to a more polished end product all around.

The disadvantage is that the methodology is harder to use for smaller projects as it doesn't use all of its advantages, and the project could also go off the track.



#### Rapid Application Development (RAD)

In this case the model would help with its flexibility and adaptability to changes. This would help a lot with my project as it would allow making changes to be easier. Due to prototyping in nature, there is a possibility of lesser defect and glitches.

It is harder to use it for smaller projects. And it is not suitable when technical risk is high as it requires very skilled people when using it, which is why I will not use it as my skills are not as high as the skills required to use RAD efficiently.

#### Conclusion

I will be using a combination between the two methodologies, waterfall and agile. Waterfall is used as a basic structure for the project but I will have to make my game incrementally and iterate it over time in order to get the end product that I want. These iterations are too small within the game in order to call the methodology agile but it does take aspects from both methodologies, but the main one used is waterfall.

### Test data for beta testing

#### Movement

What is being tested	Data input	Expected output
Player movement	AWSD	The character moves in the expected direction as soon as a key is pressed along with showing the specific animations attached to that "movement"
Movement Speed	N/A	The speed at which the player moves is acceptable, fast but not unfair. It still makes the player think strategically.
Dash Ability	Press Left-Shift	The character dashes forwards for about 0.2 seconds in which time it cannot take any damage and covers a reasonable distance. Enemies lose sight of you for a split second.
Enemies move when they see you	"Get close to enemy"	Enemy runs towards you looking for an attack
You can move even when ultimate is activated.	Press Space	The player can move slowly when ultimate is activated so that they can go towards the enemy.

#### Combat

What is being tested	Data Input	Expected Output
Swing scythe	Left Mouse Button	Does 50-100 damage to enemy, throws them back
Aim to throw	Right Mouse Button	Aims the scythe and gets ready to throw. Animation
Throw Scythe	Left Mouse Button (with Left Mouse Button being held)	Throws the scythe doing 25-75 ranged damage.
Retract Scythe	Click Right Mouse Button	Retracts Scythe to hands
Enemies damage player	N/A	Depending on the enemy, they can do 20-50 dps on player. Health bar decreases.
Special ability	Press Space	Does continuous damage for 15s at 150 dps for close enemies and 95dps for further enemies. With animation and decrease ability bar and increase health bar.
Camera shake	The action and amount of combat	Camera shakes during combat with every hit given and taken.
Sound effects	N/A	There are sound effects with the swings, damage and special ability that all play when the player presses the certain keys.
Combo perks	Left Mouse Button	After a certain combo multiplier, you can do a finishing move and do more damage to enemies

### Interaction

What is being tested	Data Input	Expected Output
Collecting an orb	Press and hold Q	The orb takes 1.5 seconds to absorb and fills either health or power up bar. Also makes an animation and sound effect.
Soul hunger	Press and hold Q	The character can collect souls from the creatures he has fought, but only the ones with souls. Also has an animation and sound effect.
Talking to random strangers along the way	Press Q	A dialogue between the character and the stranger will tell some of the story and a goal for the future.

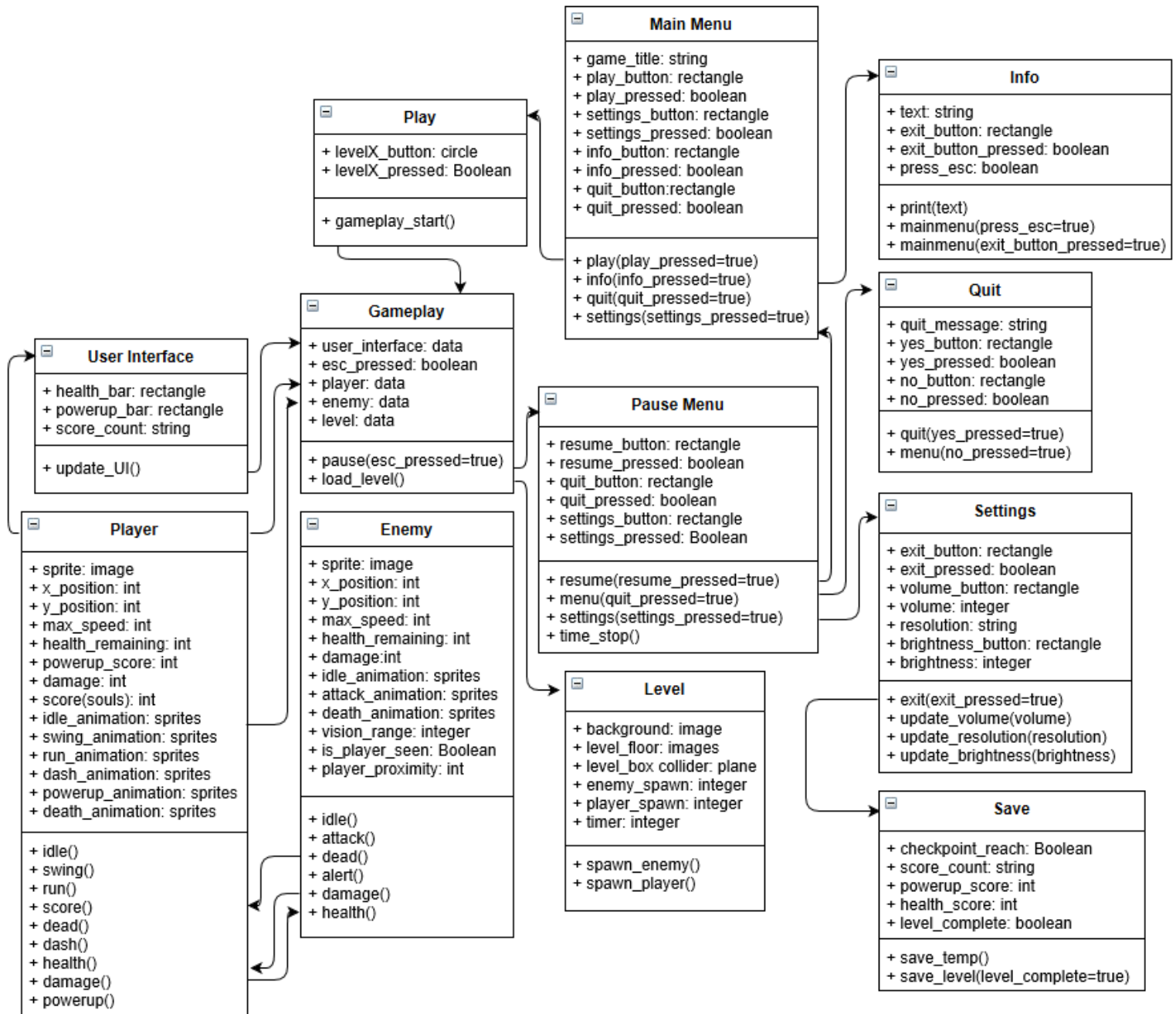
### Enemies

What is being tested	Data Input	Expected Output
Enemies run towards you	Player is in their vision range	Enemies alert each other and run towards you engaging combat.
Enemies should take 100 damage	Do damage to enemy	After 100 damage, the enemy dies so that the player can then progress.
Enemies can alert others when you are near	Get near an enemy without sneaking	Other enemies will come to attack you.
Enemies can do damage to you	Get close to enemy	The enemy should do 15 damage per close strike, which they can do every 1.5 seconds. Some enemies can do ranged attacks but not all. They have to be close to the player to do damage.

### Menu

What is being tested	Data Input	Expected Output
Can you start a new game	Left Mouse Button	The game would occupy a new profile where it would save progression in the future and the game starts
Can you continue a game	N/A	The game should continue from a checkpoint close to where you left off or at the beginning of the level. The game should also save automatically when you finish a level.
Accessing the setting menu	N/A	The game will open the setting menu to the user and allow the user to make whatever changes they want to.
Changing some of the settings	N/A	The game will then allow the user to change some of the settings in the game

## Class Diagrams



## Key Variables and Data Structures

### Main Menu

Method	Stored label	Data Type	Actual meaning	Justification
Game State	Main Menu	NA	The game has opened and the player is in the main menu	The game state will allow the player to select where they want to go from there.
Variable	Decision	Boolean	The player decides what button they want to click and the program reacts accordingly.	It allows the player to access the actual game, settings, etc.
Game State	Playing	NA	The game can then go to a number of different game states	By doing this the player can have more freedom within the menus as well as the actual game.

### Gameplay

Method	Stored label	Data Type	Actual meaning	Justification
Variable	Enemy Vision	Integer	This determines the vision of the enemy and if they can see you or not	This allows the player to sneak and it also doesn't give the enemy too much of an ability as they could see you across the level.
Variable	Enemy Speed	Integer	This is the maximum speed the enemy can travel at	This is lower than the players so that the enemy won't be too hard to fight.
Variable	Player Speed	Integer	This is the maximum speed the player can travel at	This will restrict the player from speeding across the map
Variable	Dash Speed, Duration and Cool down	Integer	The duration and speed of the dash ability along with it's cool down	This will keep the player from abusing the ability to their advantage but also help them when they need it. IT will also make sure the ability cools down before it is used again.
Variable	Player Health	Integer	This keeps track of the health of the player	This will change over time and will be compared constantly until it is bellow or equal to 0.
Variable	Enemy Health	Integer	This keeps track of the health of every enemy	This will change over time and will be compared constantly until it is bellow or equal to 0.
Variable	Damage dealt	Integer	Keeps track of the damage dealt by the player to opposing enemies	This will then be part of the score at the end of the level.
Variable	Enemies Killed	Integer	Counts how many enemies the player took out	This will again be part of the score at the end of the level.



Variable	Wave Number	Integer	This keeps count of how many waves of enemies the player took out.	This will increase with every 2-3 enemies, and will count towards the high score at the end.
Procedure	Swing	NA	This will do damage to any close enemies depending on how far they are from the player	This is the basic combat move the player will have against the waves of enemies.
Procedure	Total Score	Integer	The total score will be calculated as the player progresses through the level	This will make it easier as the score can be constantly displayed to the player while they play allowing them to compare with previous runs.
Procedure	Throw	NA	Allows the player to throw their Scythe by determining where the player is aiming the object.	This will then check what the scythe is hitting. If it is an enemy then it should do damage to them accordingly.
Variable	Power-up	Integer	According to their score, their power up might've been charged and so it holds that as a percentage.	This will be used to check if the power up is ready to be used in which case the power up is at 100% or above.
Procedure	Power up	NA	This actually activated the power up. Recharges the players dash ability and health to 100%, as well as doing damage to nearby enemies.	This ability will be activated when the player presses "space bar".
Variable	Paused	Boolean	This pauses the game when true.	Allows the player to pause the game taking them to a different game state.

Game States

Method	Stored label	Data Type	Actual meaning	Justification
Game State	Paused	NA	The gameplay is paused at this stage	The game hold the code for the game while paused until game is resumed.
Procedure	Resumed	NA	The game is then resumed by the player	This will put the player back to where they left off.
Game State	Settings	NA	The game goes to settings in this stage	Allows the player to customize some of the game play and experience.
Procedure	Back	NA	Allows the player to go back to the main menu	This allows the player to make a different decision such as starting the actual game.

### Iterative Development

The development process of my game follows an iterative process where after the prototype is made, the client will give comments and advice on where the direction the game should be taken in, and will then follow these directions to meet certain requirements with the next prototype, and so on until the client is happy with the final product.

In reality this process would repeat until the budget limit is met or the client is happy enough with the product but in this case, the project is too small to make use of any real budget, and unlikely to reach a publishable standard.

However, the game will go through those iterations throughout the development cycle, as the game will become more and more polished as it gets closer to the final product, and the reason why it might deviate from the original idea and plan of what the game was supposed to be. This can be due to a personal creative decision or it could make the whole project go along much easier and faster if that is what the timeline and deadline will dictate, without changing the original idea.

### Coding

As I will be using unity to make my game, a lot of the things that I must do exclude coding. These are things such as level, creation or just layering the right things on screen for the player.

Unity makes use of a lot of assets and objects to make games, a lot of which don't have to be coded in, unlike the functionality of all those buttons, controls, etc.

The more specific game components will have to be made up of scripts which is where the coding aspect comes into play. This allows me to link different game objects to make them interact, for example the interactions between the player and the enemy's sword.

## Assets

### Game Level

In order to create the game level itself, I will be using this village tile set I found on the unity store. It has a lot of objects made in pixel art, and it looks very detailed, but also fits with the rest of the game.



These sprites were taken from the unity store, and will be used to create my level/levels. The sprites will also take up some memory space and render time for the user as they are quite detailed, but this is taken into account.

The level that is to be loaded is relatively small in scale, it is not a 2D open world. Also, not all the sprites will be used within the level as some of them are useless to me, and therefore will be taken out of the sprite package. This shrinks the game file size but also makes the rendering a bit quicker when starting up the level.

### Menu Assets

- Background: The background should be simple and still represent the game as well as fit the pixel art style throughout. I have decided on this background to be in the game. It has a good resolution, and works as a good canvas for the buttons and the rest of the menus to be laid on top of.

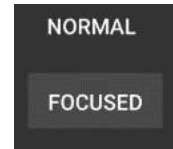
Because it has mostly a dark pallet of dominant colours, the font and buttons should be easy to spot and see for players, meaning it would work well as a main menu background.



- Font Style: The font used throughout the game should definitely be pixelated. This would go with the whole art style and idea of the game. Although the font is not definitive yet, it should look very arcade like, and should fit the game nicely along with the background as well as button designs.



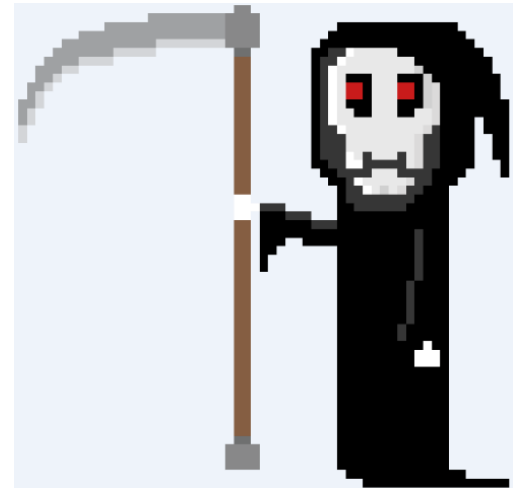
- Button Designs: The buttons should be transparent, and should darken when hovered over them. This gives a much more polished look to the whole game, and contrasts the pixel style for the font, giving it some contrast making it stand out.



### Characters

There are two main character types within the game at the moment. These being the Main Character controlled by the player and the Enemy which is an NPC, non-playable.

- Main Character: -The player, which has its own functionality and set of abilities, as well as its own sprite sheet with animations and such. The character is mean to represent the grim reaper but this art related direction might change due to the time needed to animate the character fully in pixel art before putting it into the game and coding its functionality and interactions.



The actual sprite for this character might change if animating the character will take too long and it will be too difficult to do in the time frame, or even implement in unity.

In order to animate the character, I might have to use bone structures in order to make it easier to animate. This will automatically change the look of my sprite, and maybe even spoil the 2D look of the character, as the squares themselves will not align perfectly anymore.

- Enemy (NPC): - This is the character the player cannot control or have any access to. This character is constantly trying to attack the player. The fact that there are multiple of the same enemy can make it challenging. This sprite might also get change in the future depending on the difficulty of animating this character. Coding the functionality for this character might be easier, but it is the character that is more prone to glitches or mistakes, as there are often more of the same enemy.



This is just one of the enemy types. The original idea was to include more than one, which can only be possible by using other free sprites and assets I might find online, rather than animating more than 5 enemy types by myself which could prove time consuming.

In the real world, both of these characters, along with the object package, tile set and main menu assets would be made specifically for the client and to the client's needs and wants, but because it can be very time consuming and difficult for one person, I might

have to use other sprites and assets I find online, even if they don't exactly match my style or the original idea 100%.

### Gameplay Background



This background is just a canvas on which the tile map/world will be laid over.

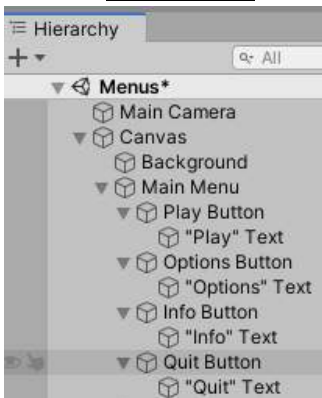
This is not a definitive decision as the resolution of the image itself may be a problem when imported to unity.

It might result in a blurry background as the image would be zoomed noticeably. There isn't a real way to fix this issue and so a different image may be used.

! A lot of the decisions are not decisive when it comes to the sprites. Using already made sprites from online stores might make more sense in the future depending on how long making the sprite sheets might take and on what the client's requirements are.

## Menu Development

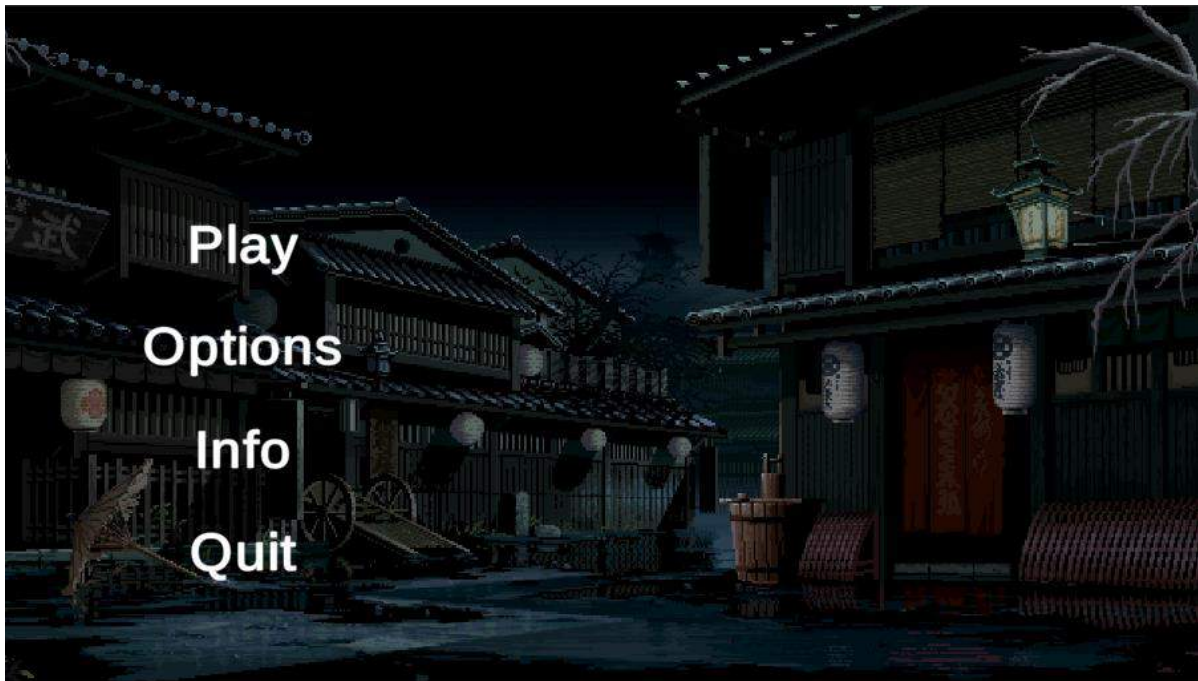
### Main Menu



Here we use a canvas along with the main camera to display the main menu to the user. This is composed of the canvas which contains the background along with the play, options, info and quit buttons.

In order to save rendering inefficiently, I will build the options menu within the main menu. This will make the game run much more efficiently at start up.

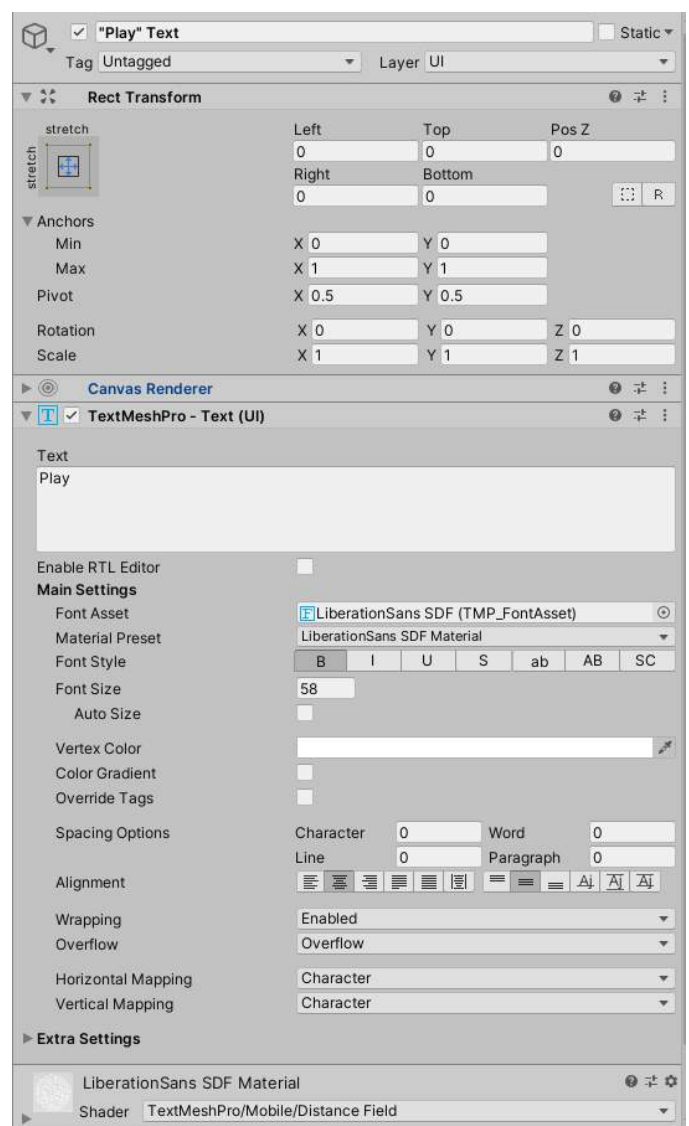
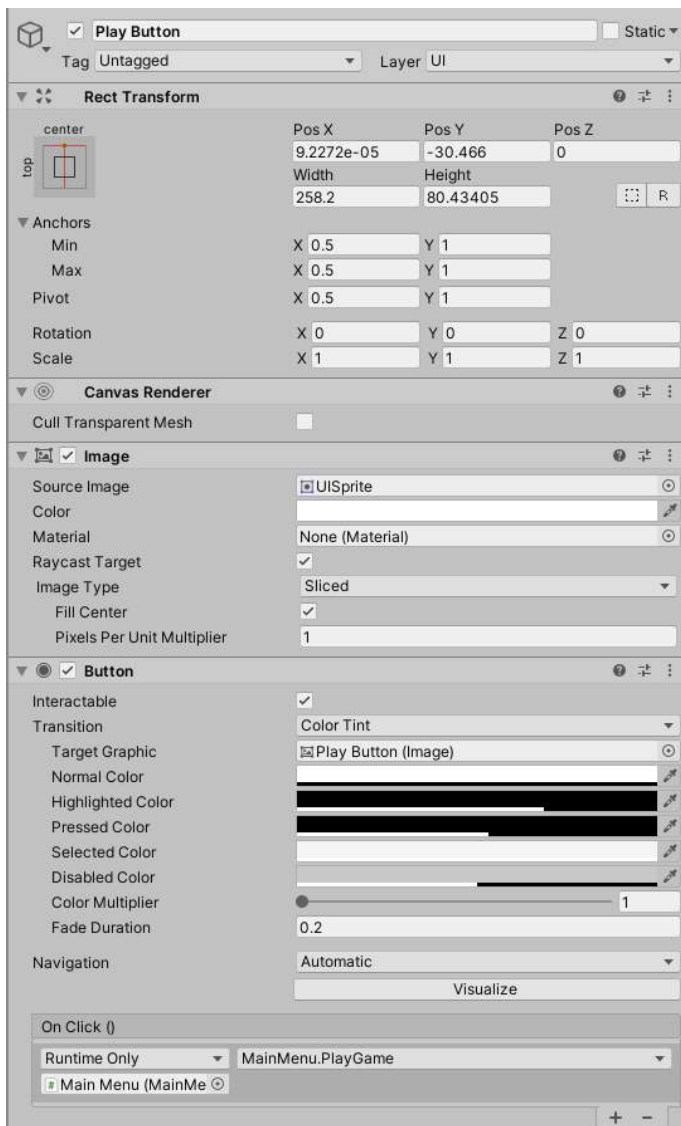
### Style



The four buttons have certain components each, along with the text in each button. Each buttons functionality is done by the C# script Main Menu as well as aesthetic and colour selection as well as onClick() events.



## Play Button and Text



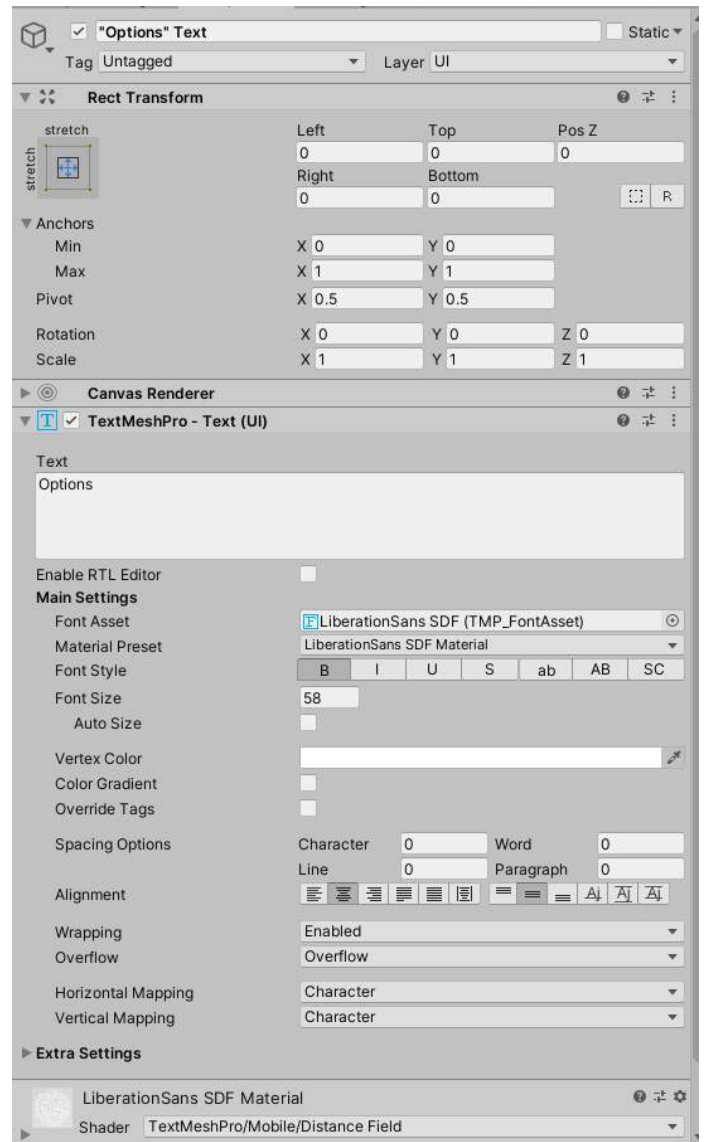
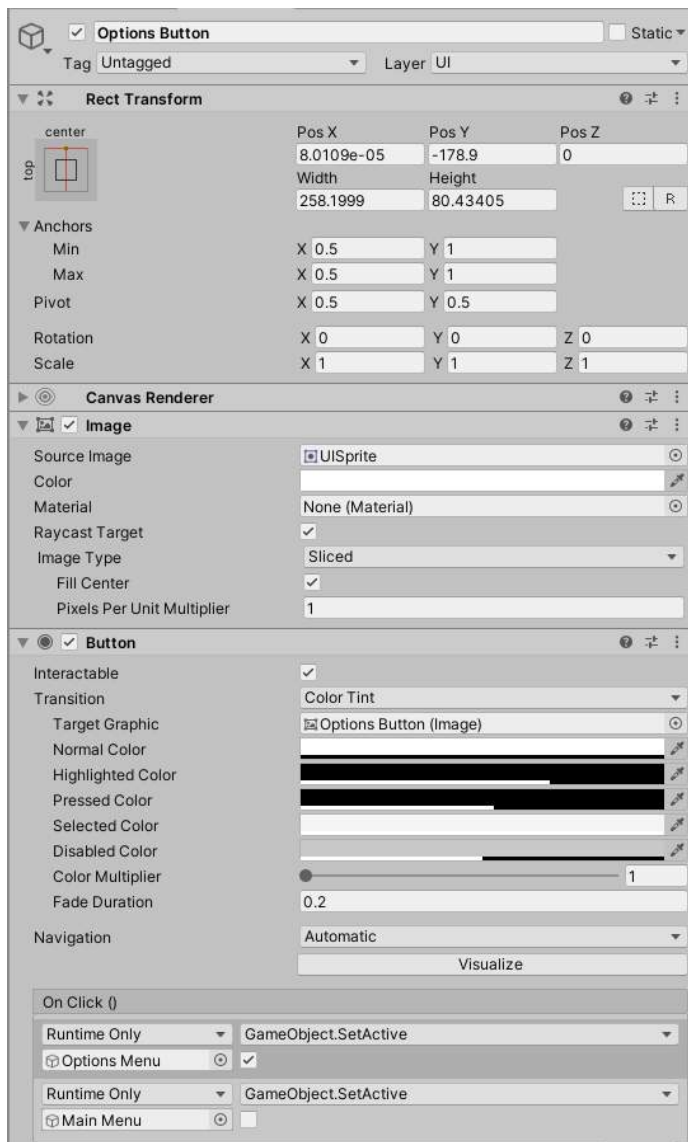
Here we have the components of the two pieces that make the Play button, the button itself and the text within it. The rest of the buttons use a similar layout in order to make them look similar.

The use of a font to replace the normal text proved to be tricky as using it gives an unknown error. Until that will get fixed, I will use the default TextMeshPro font.

The Play button will then load the next scene called Level 1 where the game is played and the rest of the assets are used and where the game will actually run. The scene is separate from this scene. The only menu accessible from that scene will be the pause menu.



## Options Button and Text

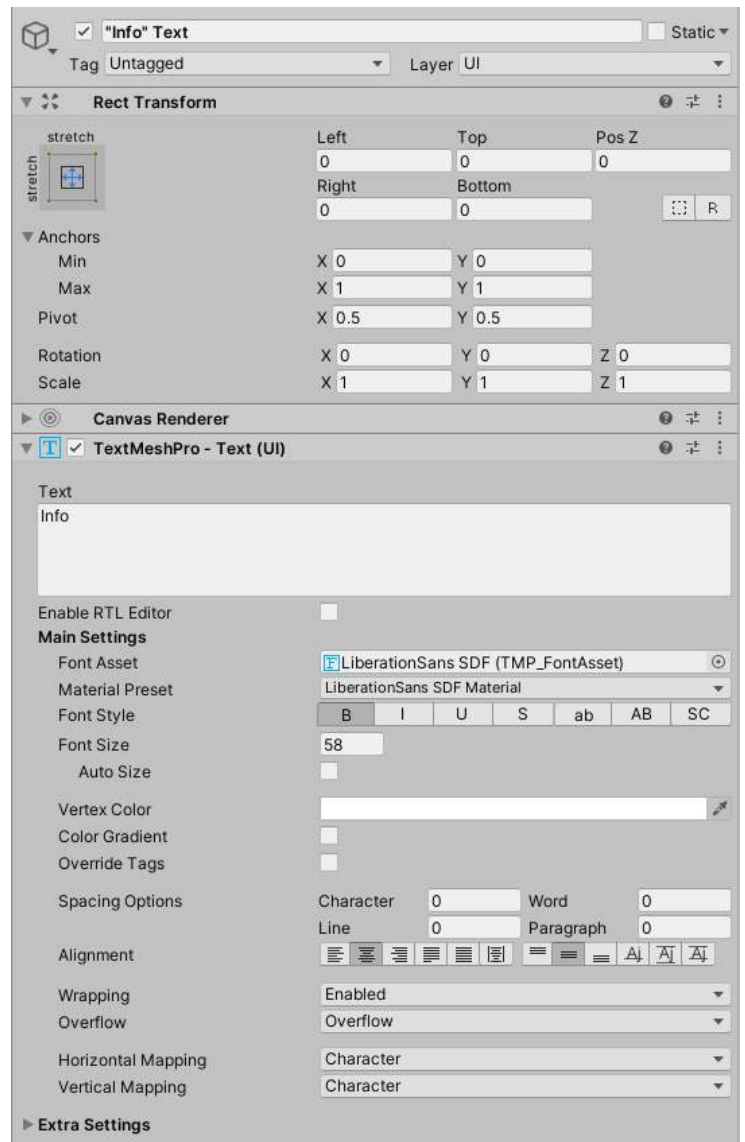
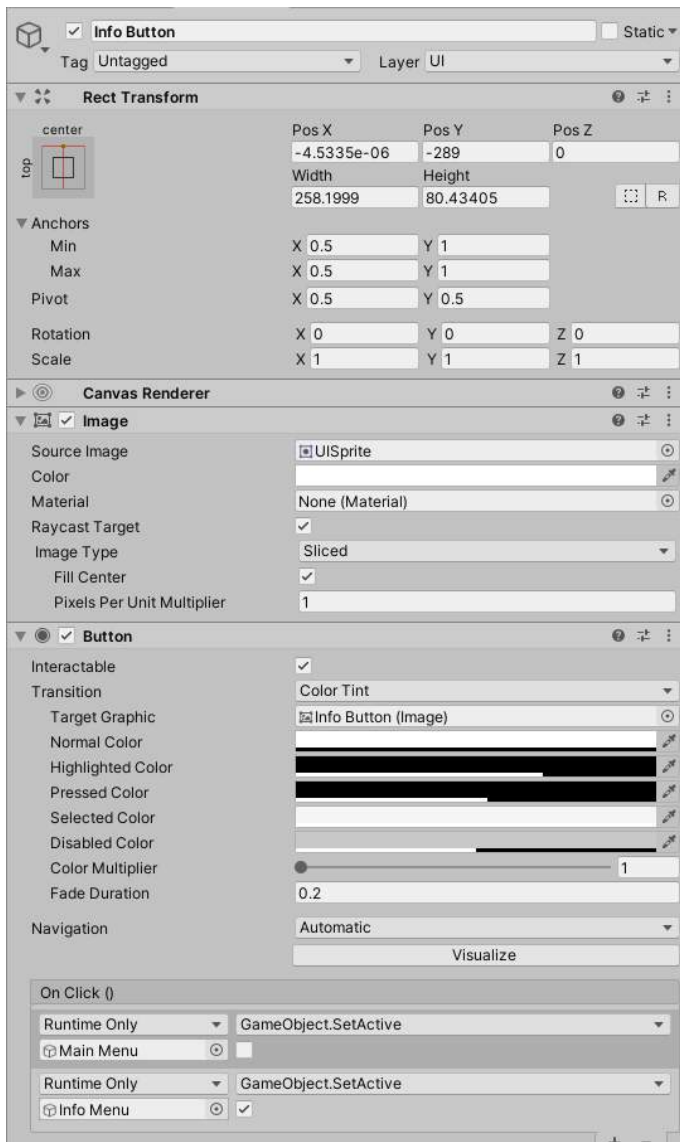


The options button again has its functionality set in the Main Menu C# script but also clicking it leads to a new menu for the settings. To make it more efficient, there is no new scene to be loaded and a scene to be unloaded as the menus can be tagged or untagged within the same scene.

The options menu will then have its functionality in a separate C# script that is just for the System settings that the player can change and have some control over.

To get back to the main menu from this menu, you will utilize the back button.

## Info Button and Text

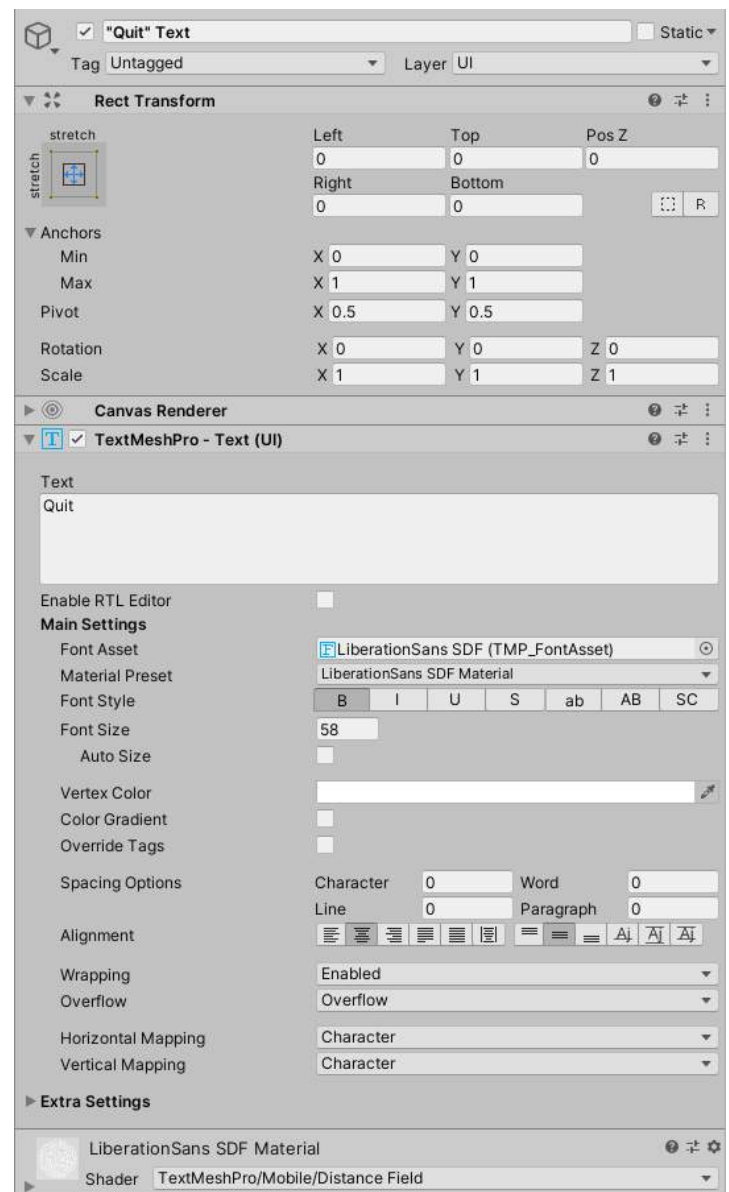
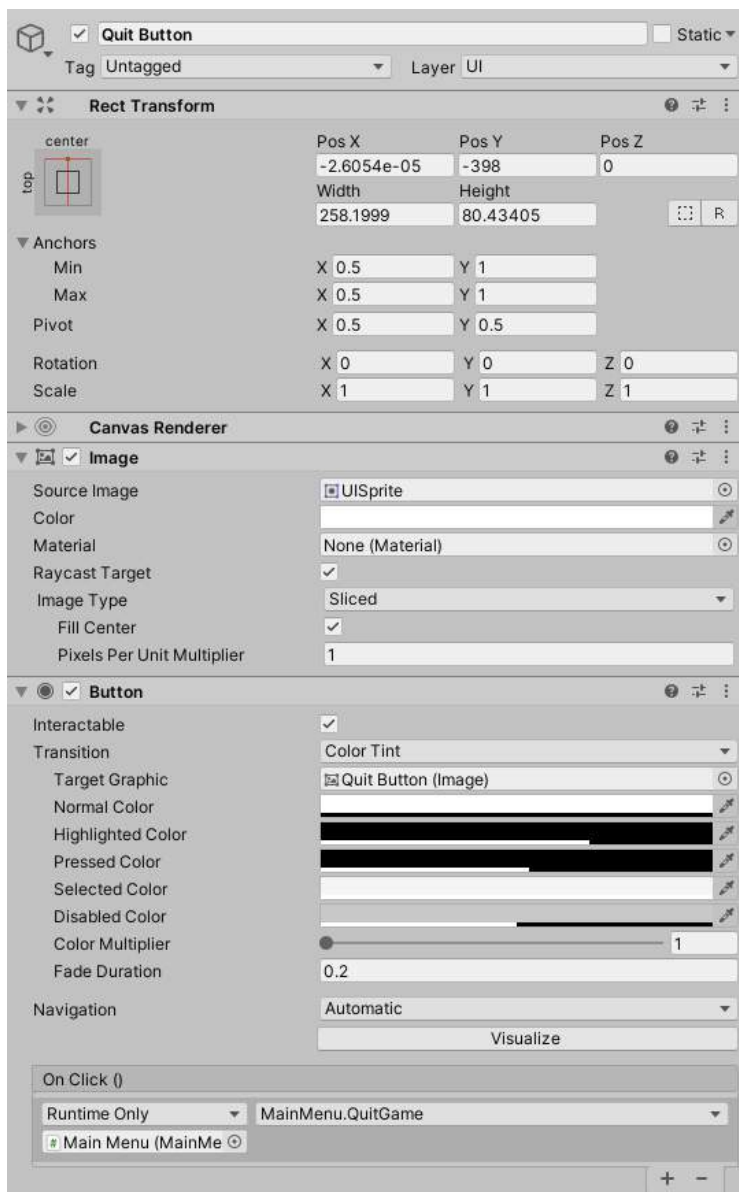


The info button will lead to another menu where information about the game will be displayed to the user in order to help them with playing the game, and giving them some hints on the game and how to play it.

The only way to get back to the main menu is to utilise the back button which then leads back to the main menu.

The On Click events activate or deactivate the menus in order to not have to load multiple scenes and use the same scene in order to make the game more efficient.

## Quit Button and Text



Options button and menu make the interactable screen UI element that when clicked leads to the options menu. This is done as the main menu is deactivated and the options menu is activated.

To go back to the menu, you need to click the back button. The functionality of these buttons will be given from the MainMenu C# script.

The interactable part of this button is given by its components, given by the TextMeshPro button settings.

### MainMenu C# Script

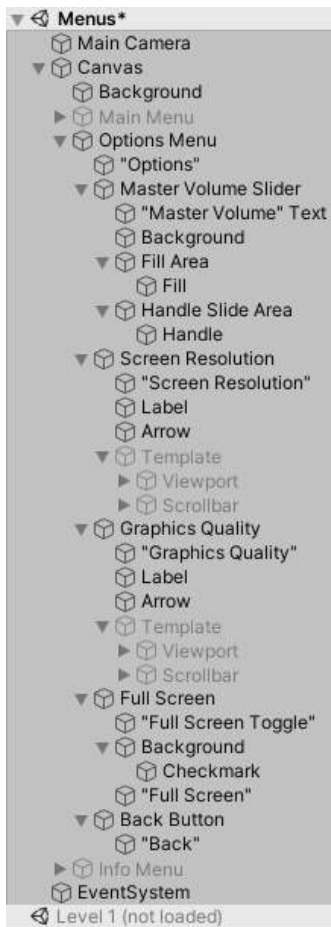
```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour {

    public void PlayGame(){
        SceneManager.LoadScene("Level 1");
    }

    public void QuitGame(){
        Debug.Log("QUIT!");
        Application.Quit();
    }
}
```

## Options Menu



The options menu is more complicated. It has a multitude of UI elements. This includes drop down, sliders and toggles.

In order to make all of those functional, a separate C# Script is made that can change the System settings themselves, without having the player to access the game files.

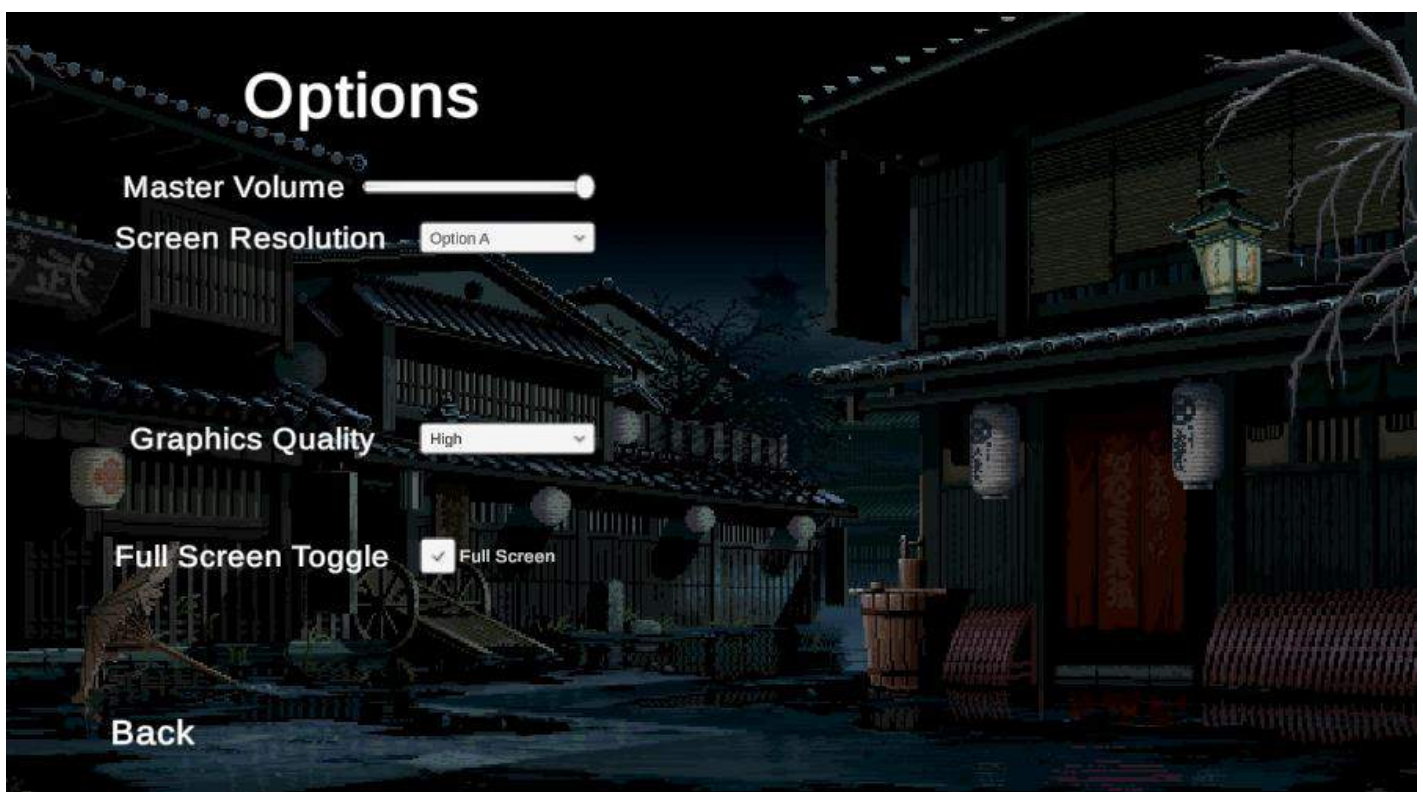
The options menu is accessed from the main menu. It is in the same scene as the main menu so in order for them to not overlap, the main menu is untagged and therefore inactive while the options menu is tagged and therefore active, allowing the player to make changes to the menu without affecting the main menu or the info menu.

The options menu also allows the player pick the best settings for their system, allowing the game to be expanded to different platforms much easier in the future, as the foundation to make sure the users get the best experience is there.

The game objects within this hierarchy are all the UI elements that together make the options menu.

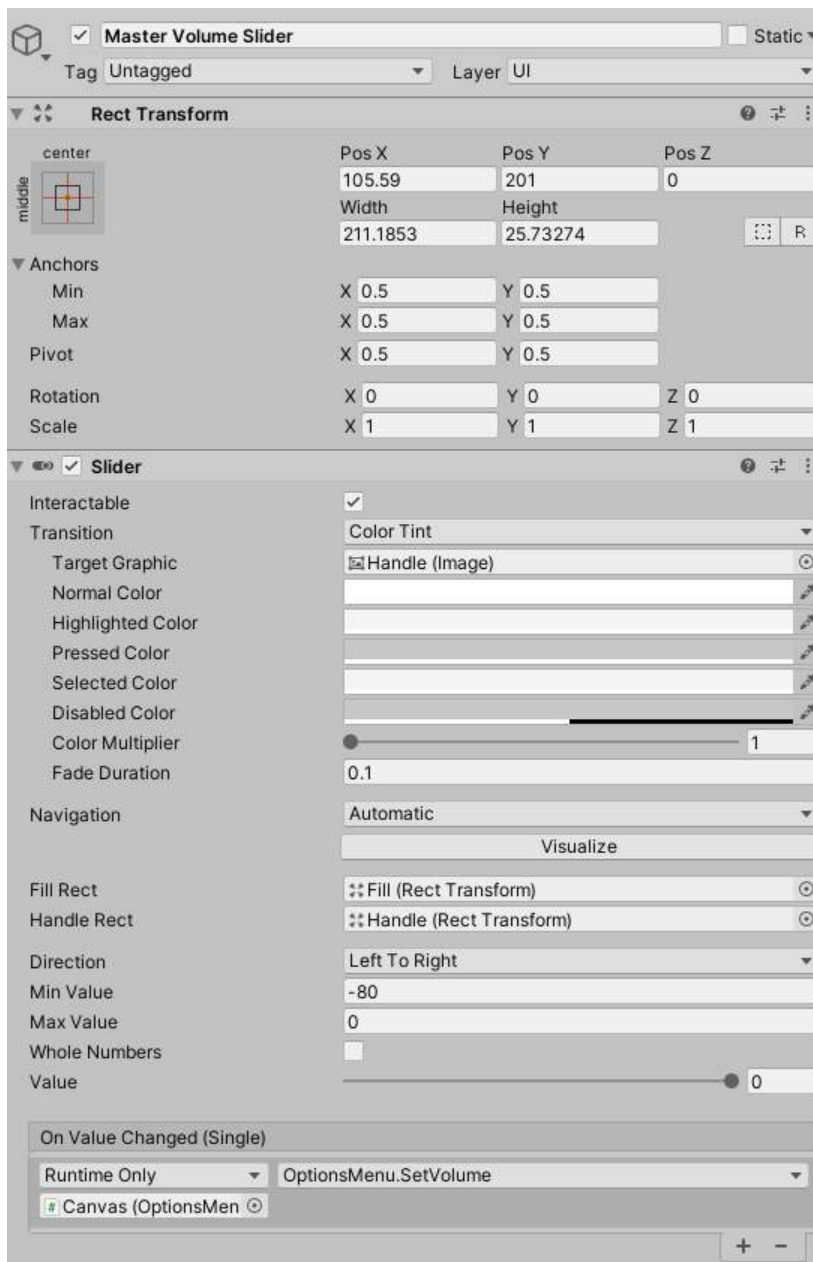
While having so many elements can be confusing and inefficient, they all go into bigger parent object which can then be tagged or untagged keeping the whole hierarchy cleaner and easier to look at.

The functionality behind all the UI elements is referenced from the canvas.





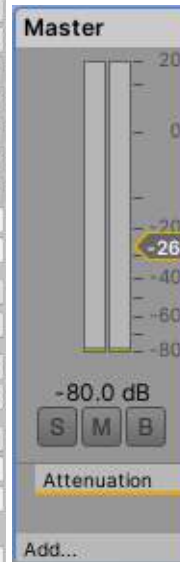
## Master Volume Slider



The master volume slider controls the volume of all the sound in the game as the name of it might suggest.

The slider holds a value that can be between -80 and 0, from left to right. Moving the slider reduces the volume of the in-game sounds.

This is done by reducing the volume of the System MainMixer which is also where all the sound queues will be played from within the game.



This then automatically controls the sound the player receives.

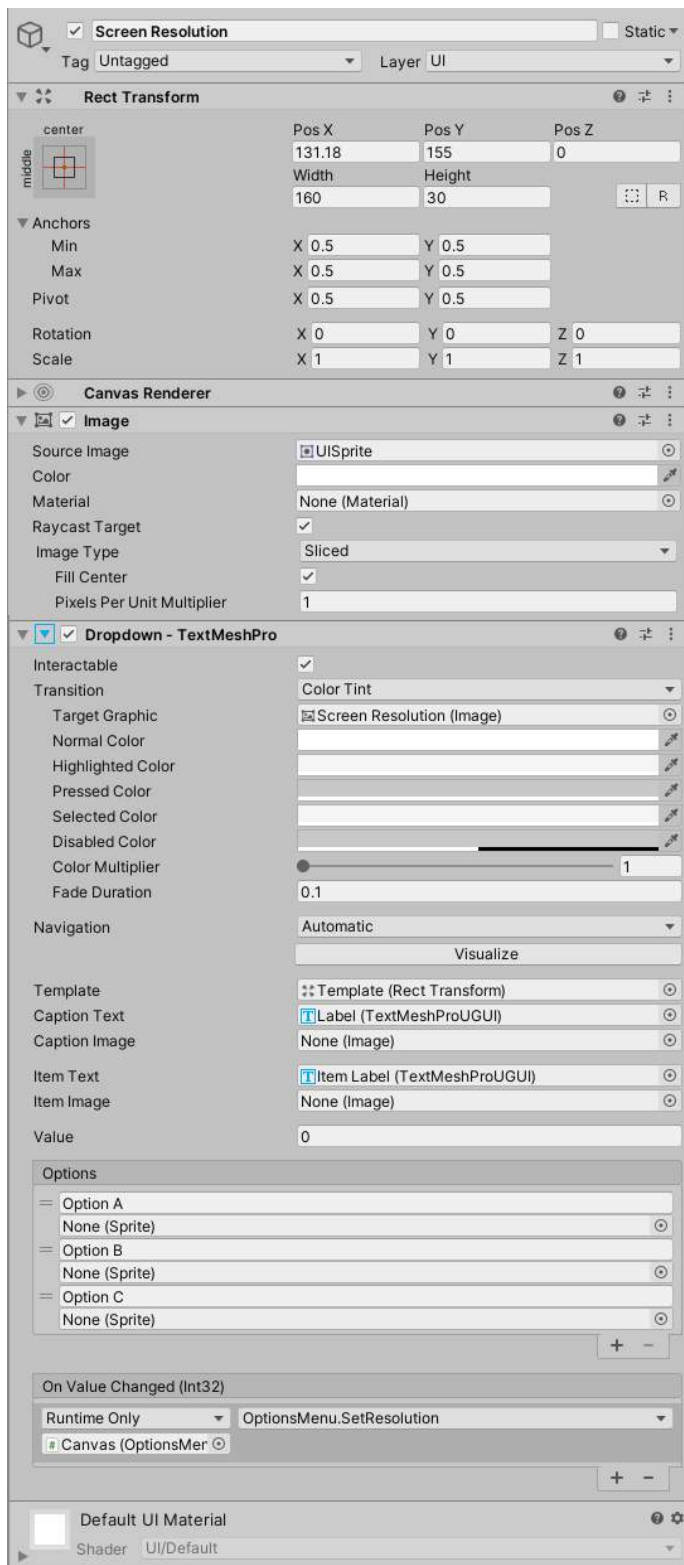
Linking this and the volume slider is done through the OptionMenu C# Script that is referenced from the Canvas.

As the slider has a value of -26, the MainMixer reduces the volume of the sound by 26 dB.

The player can also control the sound through their system volume on their computer but this is a good addition as it gives more flexibility.

The sprite of the volume slider is the default look of the TextMeshPro Package of assets. Looking for the right set of UI sprites can be time consuming so I decided not to give it too much attention.

## Screen Resolution



The screen resolution is a drop-down UI element that also has a default sprite that comes from the TextMeshPro package.

The options in the drop-down menu are dictated by the C# script OptionMenu. The options are unique to the system itself as they are not made by me, but by the script. The different options are put in a list and showed to the player. The player then selects an index which can then change the resolution of the game.

This bit of the code was copied as coding this is a bit above my skill level. It takes a lot of knowledge about the unity system itself and how it can be used to make such a setting or even compare different resolutions to each other.

The drop-down selection can have loads of options depending on the system the user is playing the game on.

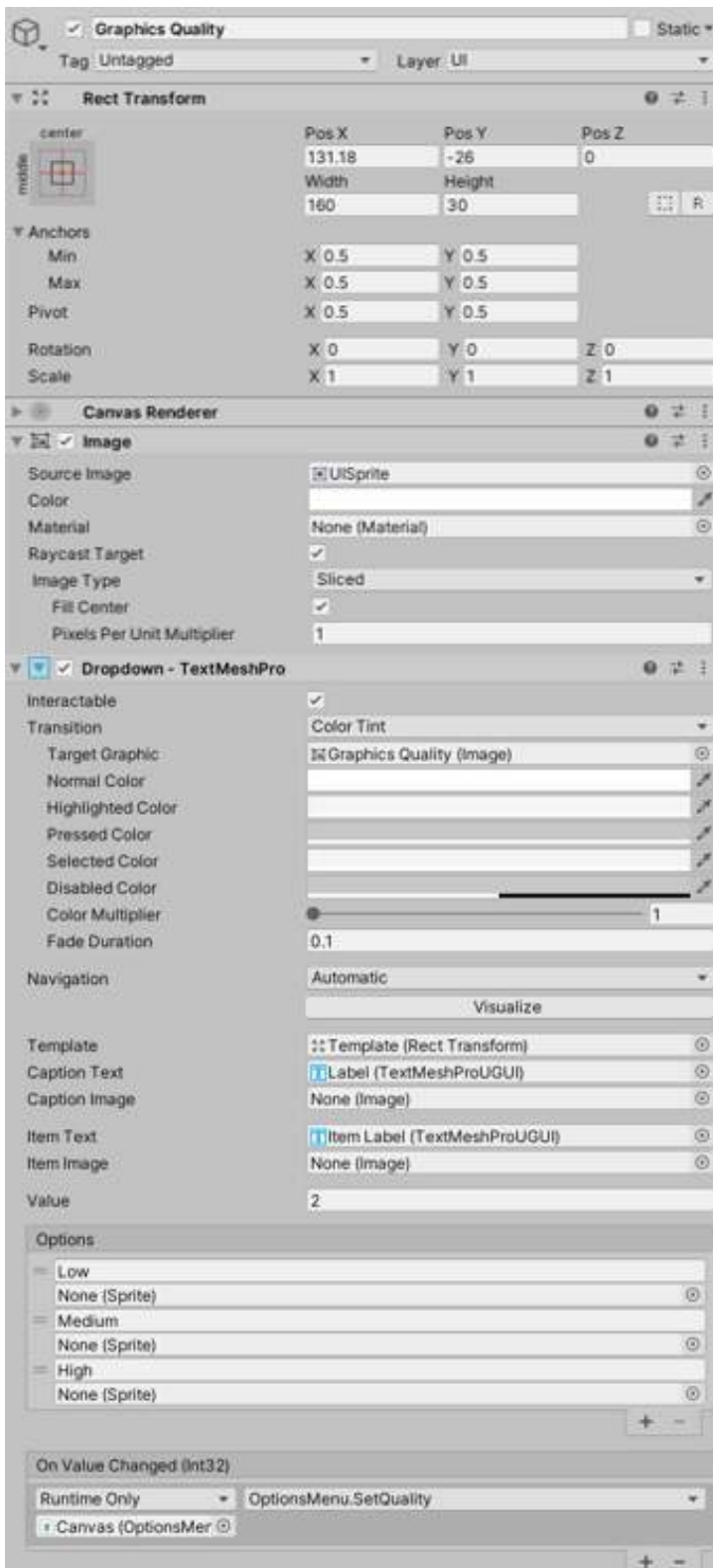


This selection is made up of a background, slider, label, arrow and a template that only activates when the arrow is clicked.

This way of sorting the settings menu is much more efficient and easier to access. It is also very intuitive as it is a very commonly used feature in websites and other applications as well as game menus and settings.

One of the places it is commonly used is picking your date of birth when making a new account for certain websites

## Graphics Quality



\*The graphic used for the drop-down selection option.

To change the graphics quality, I use the same drop-down UI element as the Screen Resolution but the functionality is different coding wise.

To change the actual rendering quality of the game in unity, you can do that by going to the project settings > quality and change from the given options which can be from Very Low to Very High.

For the sake of simplicity, I just kept three options, those being Low, Medium and High.

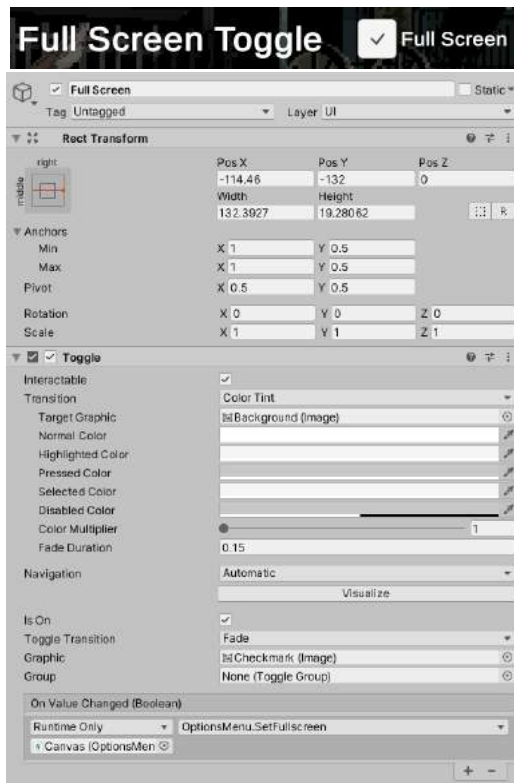
This works by taking the selected index in the drop-down menu, and then goes on to select the same index of the quality in the System Settings.

The effect it has on the game is somewhat insignificant but it does change some of the settings just by switching these pre-sets.





## Full Screen Toggle



The full screen is there as an extra way the user can customise their experience. IT allows the player to toggle full screen on making the game more immersive but also allows them to toggle this off in order to still have quick access to the rest of their PC. The toggle stores a Boolean value as it can either be on or off.

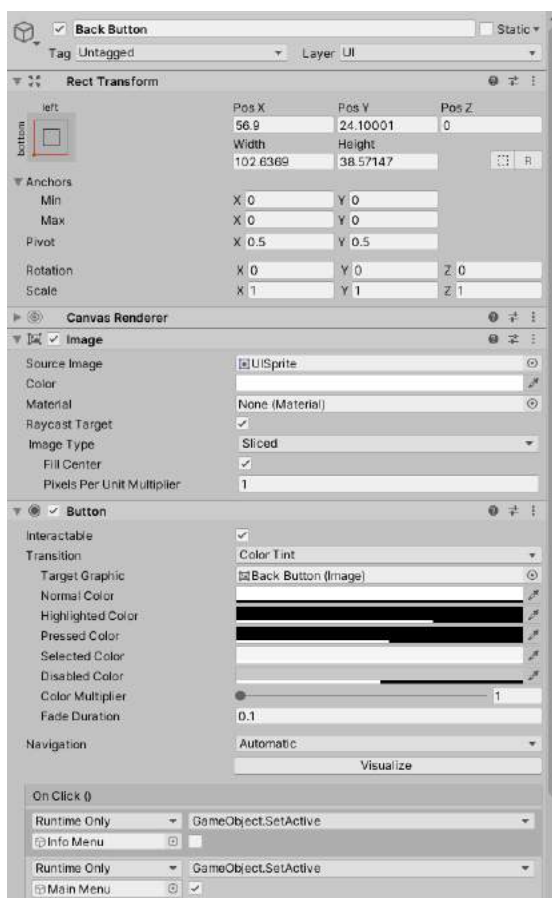
When switched on or off, it accesses the Screen settings and changes the Full screen to on or off. This is done like the previous settings, by using built in Unity functionality.

Using toggles can be much more efficient in terms of the space taken by the UI, but it gets inefficient when there are many possible toggles. Doing this for the resolution would occupy a lot of space on screen

Other settings may be added later but these are some very basic settings each user expects to have access to.

Providing these to the user is a must, as was stated in the Analysis section.

## Back Button



The back button is the same button used in the Info and uses the same functionality of main menu buttons. This uses the onClick event system, where by when this button is clicked, the option menu tag is off, and instead, the main menu tag gets turned on.



This is done by referencing the objects in the onclick events and tagging the object that should be displayed, so main menu, and leaving the other one, the info menu blank so that when the back button is clicked, this menu goes away.

! The colour of the button and it's highlighted colour as well as the selected colour remain constant across both all the back buttons and the buttons themselves as it makes the menu look more polished.

### Character Redo

I had to change the sprite for the main character in order to have animations within my game and make the character more lifelike. To do that, I used an already made sprite sheet that has more than enough animation options including running, roll, jump, fall, slide, block, death, damage, attack that has 3 variants etc.

The sprite is used is called hero Knight and it will be used to represent the main character controlled by the user.



Because the sprite sheet is already made along with the animations, all that is left is implementing these into the game.

But until then, I have to make the world first, or at least a platform on which the character can sit on.

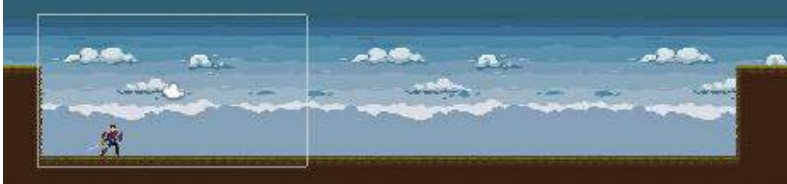
This would most likely also change the look of the enemy character so that the main character and all the enemies are within the same style of pixel art. But this would happen further in the development process due to the decision being dependent on what is available to use for free.

## Development Process

### World Tiles

To improve my work flow by using the original sprite intended to make the game with, I also decided to use a tile pallet and just “paint” the world on the grid.

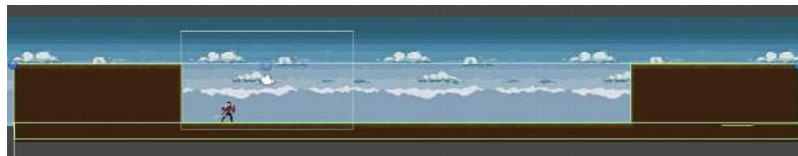
This allowed me to make more with less as it were. I made a small map that has constraints at each end of the map. This makes sure the player cannot go past it or fall out of the map.



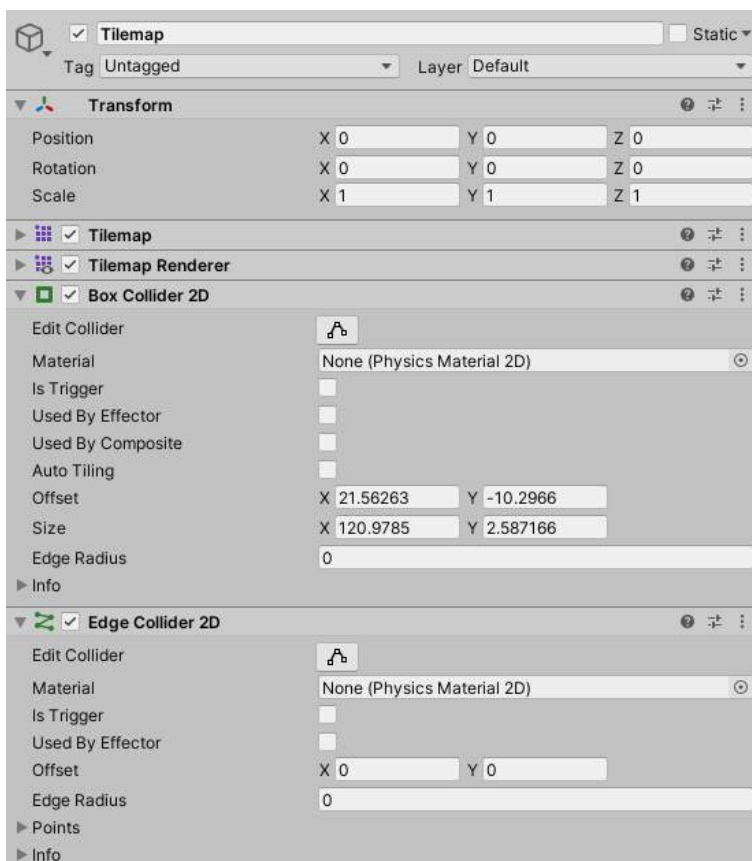
There are no world objects in the map but that will come later.

The map also uses the first background and repeats it.

In order to make sure the player can actually make contact with the tile map; I added a component called Box collider or Edge collider that stops objects affected by gravity such as the player from falling through the tile set map.



These are represented by the green lines as a structure that represents the actual tiles on the screen as a collective and not just very square by itself.



The tile map box collider has no script interacting with it as it is just as a simple shape that ensure that the rest of the physics within the other game objects such as the player can function without any problems.

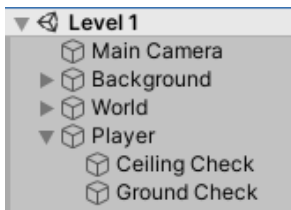
It works as a foundation for the rest of the game world.

This world will be improved later on but this is the bare minimal requirement in order to advance with the rest of the game development

Things such as grass, other objects such as trees, wood logs, and others will be added but later in the development stage as they do not have a big role to play other than for the aesthetic of the game.

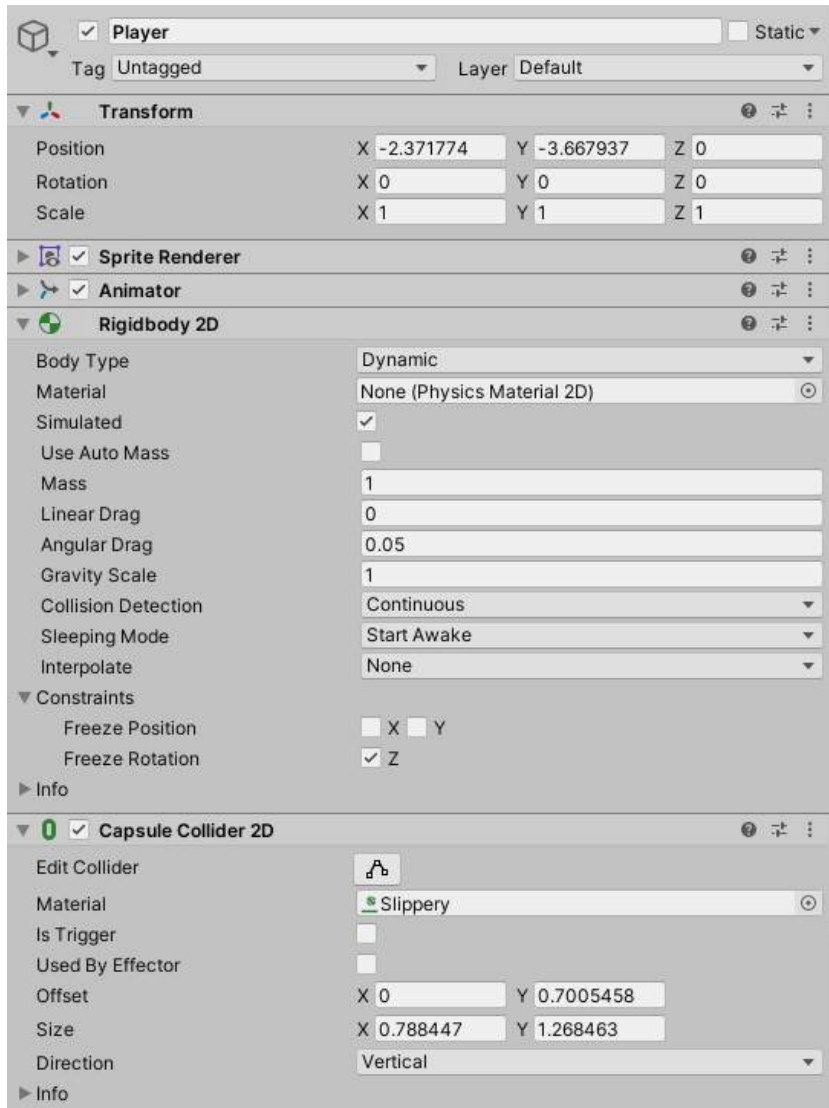
The Colliders are made of two. The box collider for the floor and the edge collider for the map limits as well as the higher platforms on the side.

## Character Development



Making the objects Ceiling Check and Ground Check will be useful later.

For now, we have to give Player a sprite to represent the player on the screen as well as giving that sprite actual physics and colliders so that the character can interact with the world around it.



The settings in the components are simple so far.

Rigid body 2D is what gives the character the physics within the game.

Collision detection is continuous which is important in order to ensure Rigid body always updates itself.

Also, rotation is frozen in the Z axis. It is a 2d game so we wouldn't want the character to rotate in that axis.

Other than that, the settings are fairly default.

The capsule collider is the shape around the sprite, just like the tile map box collider, that ensures that the player can interact with the floor, restrictions and other world objects.

The shape is a capsule, similar to an oval. Using this shape instead of a square gives more flexibility. If the player goes up a ramp. Having a curved shape would help rather than having hard edges.

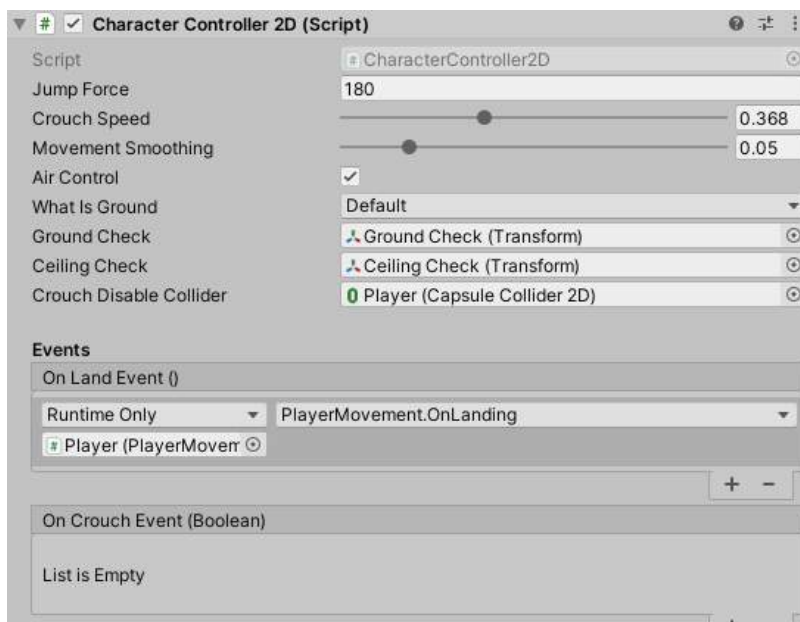
The slippery material property is also important.

### Slippery

It is just a material asset that has 0 friction and 0 bounciness. This has affect in the vertical direction. This means that when jumping on a wall, the player wouldn't just be stuck up there until they stop going towards that wall. The player should be sliding back down which is what this 2D material ensures that happens.

Otherwise, jumping on a wall and keep driving towards that wall would result in the player hanging in the air levitating. In order to prevent that, I made every all "slippery" for the player.

## Character Movement



In order to make the character move, we need two scripts. One that takes care of the state that the character game object is in. Things such as flipping the sprite depending on which direction the character is facing but as well as giving us control over things such as Jump force, crouching, air control and more of the physics behind the character.

These are unique to 2D character. This piece of code has been recycled from an already existing product and implemented into my game as it is a more complicated script to develop. Then, making the parameters to look realistic for example the jumping force and so on.

### CharacterController2D.cs (C# Script)

```
using UnityEngine;
```

```
using UnityEngine.Events;
```

```
public class CharacterController2D : MonoBehaviour{
```

```
    [SerializeField] private float m_JumpForce = 400f;           // Amount of force added when the
    player jumps.
```

```
    [Range(0, 1)] [SerializeField] private float m_CrouchSpeed = .36f;    // Amount of maxSpeed
    applied to crouching movement. 1 = 100%
```

```
    [Range(0, .3f)] [SerializeField] private float m_MovementSmoothing = .05f; // How much to smooth
    out the movement
```

```
    [SerializeField] private bool m_AirControl = false;         // Whether or not a player can steer
    while jumping;
```

```
    [SerializeField] private LayerMask m_WhatIsGround;          // A mask determining what is
    ground to the character
```

```
    [SerializeField] private Transform m_GroundCheck;           // A position marking where to
    check if the player is grounded.
```

```
    [SerializeField] private Transform m_CeilingCheck;          // A position marking where to check
    for ceilings
```

```
    [SerializeField] private Collider2D m_CrouchDisableCollider; // A collider that will be disabled
    when crouching
```

```

const float k_GroundedRadius = .2f; // Radius of the overlap circle to determine if grounded

private bool m_Grounded;          // Whether or not the player is grounded.

const float k_CeilingRadius = .2f; // Radius of the overlap circle to determine if the player can stand up

private Rigidbody2D m_Rigidbody2D;

private bool m_FacingRight = true; // For determining which way the player is currently facing.

private Vector3 m_Velocity = Vector3.zero;

[Header("Events")]

[Space]

public UnityEvent OnLandEvent;

[System.Serializable]

public class BoolEvent : UnityEvent<bool> { }

public BoolEvent OnCrouchEvent;

private bool m_wasCrouching = false;


private void Awake(){
    m_Rigidbody2D = GetComponent<Rigidbody2D>();
    if (OnLandEvent == null)
        OnLandEvent = new UnityEvent();
    if (OnCrouchEvent == null)
        OnCrouchEvent = new BoolEvent();
}


private void FixedUpdate(){
    bool wasGrounded = m_Grounded;

    m_Grounded = false;

    Collider2D[] colliders = Physics2D.OverlapCircleAll(m_GroundCheck.position,
k_GroundedRadius, m_WhatIsGround);

    for (int i = 0; i < colliders.Length; i++){
        if (colliders[i].gameObject != gameObject){
            m_Grounded = true;
            if (!wasGrounded)
                OnLandEvent.Invoke();
        }
    }
}

```

```

    }
}

public void Move(float move, bool crouch, bool jump){
    if (!crouch){
        if (Physics2D.OverlapCircle(m_CeilingCheck.position, k_CeilingRadius,
m_WhatIsGround)){
            crouch = true;
        }
    }
    if (m_Grounded || m_AirControl){
        if (crouch){
            if (!m_wasCrouching){
                m_wasCrouching = true;
                OnCrouchEvent.Invoke(true);
            }
            move *= m_CrouchSpeed;
            if (m_CrouchDisableCollider != null)
                m_CrouchDisableCollider.enabled = false;
        }
        else{
            if (m_CrouchDisableCollider != null)
                m_CrouchDisableCollider.enabled = true;
            if (m_wasCrouching){
                m_wasCrouching = false;
                OnCrouchEvent.Invoke(false);
            }
        }
        Vector3 targetVelocity = new Vector2(move * 10f, m_Rigidbody2D.velocity.y);
        m_Rigidbody2D.velocity = Vector3.SmoothDamp(m_Rigidbody2D.velocity,
targetVelocity, ref m_Velocity, m_MovementSmoothing);
        if (move > 0 && !m_FacingRight){
            // ... flip the player.
            Flip();
        }
    }
}

```

```

        }

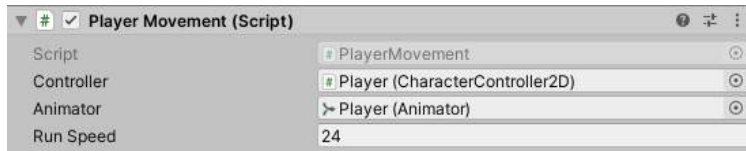
        else if (move < 0 && m_FacingRight){
            Flip();
        }
    }

    if (m_Grounded && jump){
        m_Grounded = false;
        m_Rigidbody2D.AddForce(new Vector2(0f, m_JumpForce));
    }
}

private void Flip(){
    m_FacingRight = !m_FacingRight;
    Vector3 theScale = transform.localScale;
    theScale.x *= -1;
    transform.localScale = theScale;
    //this is the flip function
}
}

```





The other script deals with the actual player movement and input management as well as the running speed of the player and references the animator of the character and the controller.

This script is much simpler as all it does is move the player horizontally or vertically depending on what button was pressed, and checks these inputs for every single frame

This script not only physically moves the sprite but also makes use of the animator, and changes the values of “Speed” as well as “Jump”. These values will then be used as conditions to animate the character.

Input manager was also changed in order to meet the controls that were determined in the design phase. By default, Unity gave left and right arrow keys as well as “a” and “d” for horizontal movement and space bar for jump.

Space bar was then replaced by the button “w” to meet these control requirements. The controls will also be included in the Info menu later in the project.

### *PlayerMovement.cs (C# Script)*

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
public class PlayerMovement : MonoBehaviour{
```

```
    public CharacterController2D controller;
```

```
    public Animator animator;
```

```
    public float runSpeed = 40f;
```

```
    float horizontalMove = 0f;
```

```
    bool jump = false;
```

```
    void Start(){
```

```
    }
```

```
    // Update is called once per frame
```

```
    void Update(){
```

```

horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed;

animator.SetFloat("Speed", Mathf.Abs(horizontalMove));

if (Input.GetButtonDown("Jump")){
    jump = true;
    animator.SetBool("isJumping", true);
}
}

public void OnLanding(){
    animator.SetBool("isJumping", false);
}

void FixedUpdate(){
    //Moving Character

    controller.Move(horizontalMove * Time.fixedDeltaTime, false, jump);
    //Moves character the same amount regardless of the amount of times the function is ran.

    jump = false;
}
}

```

### Controls Breakdown

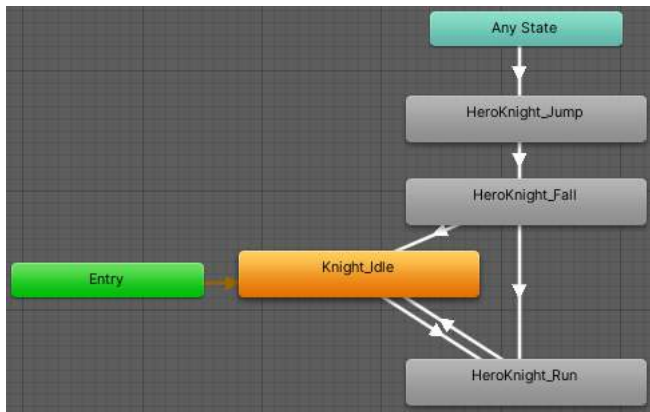
The controls used within the game can be changed from the unity Project Settings. I had to change them to meet the specified.

Horizontal		Vertical	
Name	Horizontal	Name	Vertical
Descriptive Name		Descriptive Name	
Descriptive Negative Name		Descriptive Negative Name	
Negative Button	a	Negative Button	down
Positive Button	d	Positive Button	up
Alt Negative Button		Alt Negative Button	s
Alt Positive Button		Alt Positive Button	w
Gravity	3	Gravity	3
Dead	0.001	Dead	0.001
Sensitivity	3	Sensitivity	3
Snap	<input checked="" type="checkbox"/>	Snap	<input checked="" type="checkbox"/>
Invert	<input type="checkbox"/>	Invert	<input type="checkbox"/>
Type	Key or Mouse Button	Type	Key or Mouse Button
Axis	X axis	Axis	X axis
Joy Num	Get Motion from all Joysticks	Joy Num	Get Motion from all Joysticks

The controls now meet the desired requirement that was stated in the design. SO far the movement, happens only with “a”, “d”, “w” keys for left right and jump respectively.

The rest of the keys for things such as the attack types as well as special abilities will be added in later.

## Animation of Character



The basic animation involves the Idle sprite set of around 8 frames. The conditions that allow the animations to switch from one to another.

The parameters are Speed that is an integer, and isJumping that is Boolean. If isJumping is true then the animation switches to the jump animation, and then to the fall animation after the exit time is over.

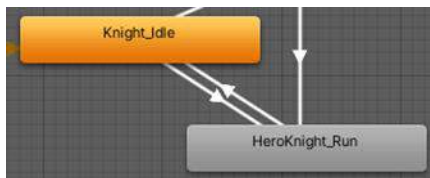
The parameter is changed by the PlayerMovement Script when it detects the specific input.

### Entry > Knight Idle



This transition is default. It will happen all the time as long as the other animations are not playing. This replaces the static sprite/image the character started off as.

### Knight Idle <> HeroKnight Run



This transition can happen both ways and takes into account the parameter of speed. If speed is above 0.01 then the animation plays until speed will be below the given value.

### Any State > HeroKnight Jump



Conditions	
= isJumping	true

### HeroKnight Jump > HeroKnight Fall



This transition has no condition, it just has a time exit as the animation will play after the jump animation happens once.

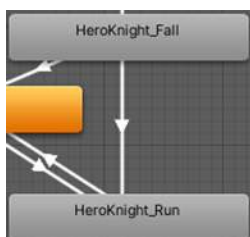
This transition plays as long as the player is falling, without playing the jumping animation again.

### HeroKnight Fall > Knight Idle

This transition will happen if speed is below the given value, meaning that the player is static. And there for the player should idle on the ground.



Conditions	
= isJumping	false
= Speed	Less 0.01



### HeroKnight Fall > HeroKnight Idle

This condition happens when speed is more than 0.01, and therefore the player should be running and so that is the animation that plays.

### Log in System

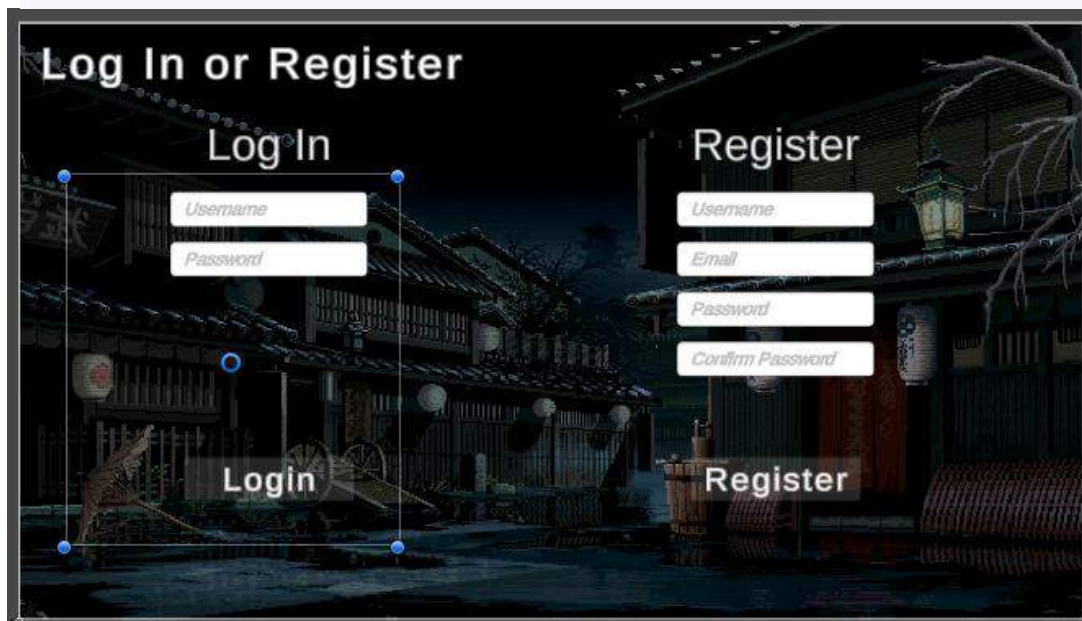
This log in system would allow the user to register using a valid email and a password they have created. The details would then be saved using the Firebase system, allowing the user to connect and register as long as they have an internet connection. This will then allow them to log in the next time they play the game as their details have been saved online.

It also allows me to track and view app usage and log in performance using the Firebase console which can also be accessed by me using my Google account and internet connection.

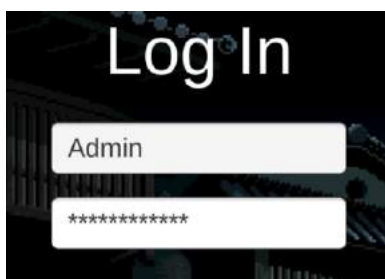
Not keeping the log in files on the person's computer can be a positive, as they can log in to their account and play the game from any device if they have an internet connection.



Identifier	Providers	Created	Signed in	User UID
test@email.com		29 Mar 2021		PfuMXUXWRygwZ5EL6iGZl62lq0l2



Here we can see the log in/register menu, which is the first menu that the player is greeted with. We can also see that it has the same style and background as the other menus.



The first important feature that was added to is replacing the characters in the password fields with asterisks in order to keep the password hidden from people that may be looking at the screen. An extremely common practice among almost all apps and webpages.

Although the data sent to the Firebase are actual characters and the actual input the person has entered.

### Using Firebase

Firebase is a platform developed and owned by Google and is widely used for developing web applications as well as supporting games. Unity has now implemented new features such as the new package manager to encourage people to use firebase which is imported as a package.

It allows me specifically, to add more functionality to my game's log in system, as well as making the game easier to scale up in the future. It keeps the users data safe, and also gives me some monitoring options when it comes to the game.

Seeing as firebase is new, not only for me but also refreshed by unity, it adds a lot of complex scripts to the game. This package seems to be the best option for when making a smaller sized game in which case the idea of maintaining a server that keeps the users data is not a viable solution.

Trying to bring the package into unity to try and use it in my game showed some difficulties. Firstly, in order to do so, it is best to use a newer version of unity, later than 2020.1.17f, whereas I was using a slightly older version, 2020.1.16f.

After installing the new version with almost no issues, adding the Firebase packages was much easier than before as using Packet Manager was much easier.

Another error came to surface, which some people struggle with. As far as the people on forums can tell, it is not due to a specific reason. When adding a script to a game object, the Script Component of the game object gives an error, not allowing you to link other game objects to that script that you need to make the game, although it can likely be due to the file name and class name not matching, an error that seems common with Unity's C# history.

The fix seems to be either deleting the meta files of the scripts and reimporting them to unity, given you have no compile errors, or take the scripts to a folder out of the project, delete the ones within the project folder, build the project, reimported all the scripts from the backup folder one by one.

Due to the complexity of the firebase package in Unity, the scripts have not been coded by me, but I have the responsibility to try and implement them in my game as far as unity allows me to.

### Firebase Scripts

#### **AuthUIManager**

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
using TMPro;
```

```
public class AuthUIManager : MonoBehaviour{
```

```
    public static AuthUIManager instance;
```

```
    [Header("References")]
```

```
    [SerializeField]
```

```
    private GameObject checkingForAccountUI;
```

```
    [SerializeField]
```

```
    private GameObject loginUI;
```

```
    [SerializeField]
```

```
    private GameObject registerUI;
```

```
    [SerializeField]
```

```
    private GameObject verifyEmailUI;
```

```
    [SerializeField]
```

```
    private TMP_Text verifyEmailText;
```

```
    private void Awake()
```

```
    {
```

```
        if (instance == null){
```

```
            instance = this;
```

```
        }
```

```
        else if (instance != this){
```

```
            Destroy(gameObject);
```

```
        }
```

```
    }
```

```
private void ClearUI(){
    loginUI.SetActive(false);
    registerUI.SetActive(false);
    FirebaseManager.instance.ClearOutputs();
}
```

```
public void LoginScreen(){
    ClearUI();
    loginUI.SetActive(true);
}
```

```
public void RegisterScreen(){
    ClearUI();
    registerUI.SetActive(true);
}
}
```

### **Firebase Manager**

```
using System.Collections;
using Firebase;
using Firebase.Auth;
using TMPro;
using UnityEngine;
```

```
public class FirebaseManager : MonoBehaviour{
    public static FirebaseManager instance;

    [Header("Firebase")]
    public FirebaseAuth auth;
    public FirebaseUser user;
    [Space(5f)]
```



```

[Header("Login References")]

[SerializeField]
private TMP_InputField loginEmail;

[SerializeField]
private TMP_InputField loginPassword;

[SerializeField]
private TMP_Text loginOutputText;

[Space(5f)]

[Header("Register References")]

[SerializeField]
private TMP_InputField registerUsername;

[SerializeField]
private TMP_InputField registerEmail;

[SerializeField]
private TMP_InputField registerpassword;

[SerializeField]
private TMP_InputField registerConfirmPassword;

[SerializeField]
private TMP_Text registerOutputText;

private void Awake(){
    DontDestroyOnLoad(gameObject);
    if (instance == null){
        instance = this;
    }
    else if (instance != this){
        Destroy(instance.GetHashCode());
        instance = this;
    }
}

```

```

FirebaseApp.CheckAndFixDependenciesAsync().ContinueWith(checkDependencyTask =>{
    var dependencyStatus = checkDependencyTask.Result;
    if (dependencyStatus == DependencyStatus.Available){
        InitializeFirebase();
    }
    else{
        Debug.LogError($"Could not resolve all Firebase dependencies: {dependencyStatus}");
    }
});
}

```

```

private void InitializeFirebase(){
    auth = FirebaseAuth.DefaultInstance;

    auth.StateChanged += AuthStateChanged;
    AuthStateChanged(this, null);
}

```

```

private void AuthStateChanged(object sender, System.EventArgs eventArgs){
    if (auth.CurrentUser != user){
        bool signedIn = user != auth.CurrentUser && auth.CurrentUser != null;

        if (!signedIn && user != null){
            Debug.Log("Signed out!");
        }

        user = auth.CurrentUser;

        if (signedIn){
            Debug.Log($"Signed in: {user.DisplayName}");
        }
    }
}

```

```

    }
}

```

```

public void ClearOutputs(){
    loginOutputText.text = "";
    registerOutputText.text = "";
}

```

```

public void loginButton(){

}

```

```

public void registerButton(){

}

```

```

private IEnumerator LoginLogic(string _email, string _password){
    Credentials credentials = EmailAuthProvider.GetCredential(_email, _password);

    var loginTask = auth.SignInWithCredentialAsync(credentials);

    yield return new WaitUntil(predicate: () => loginTask.IsCompleted);

    if (loginTask.Exception != null){
        FirebaseException firebaseException =
(FirebaseException)loginTask.Exception.GetBaseException();

        AuthError error = (AuthError)firebaseException.ErrorCode;
        string output = "Unknown Error, Please Try Again";

        switch (error){

```

```

        case AuthError.MissingEmail:
            output = "Please enter your email!";
            break;
        case AuthError.MissingPassword:
            output = "Please enter your password!";
            break;
        case AuthError.InvalidEmail:
            output = "invalid Email";
            break;
        case AuthError.WrongPassword:
            output = "Incorrect Password!";
            break;
        case AuthError.UserNotFound:
            output = "Account does not exist!";
            break;
    }
    loginOutputText.text = output;

}

else{
    if (user.IsEmailVerified){
        yield return new WaitForSeconds(1f);
        GameManager.instance.ChangeScene(1);
    }
    else{
        //Send verification email
        GameManager.instance.ChangeScene(1);
    }
}
}

```

```

private IEnumerator RegisterLogic(string _username, string _email, string _password, string
_confirmPassword){
    if (_username == ""){
        registerOutputText.text = "Please enter a username";
    }
    else if (_password != _confirmPassword){
        registerOutputText.text = "Passwords do not match!";
    }
    else{
        var registerTask = auth.CreateUserWithEmailAndPasswordAsync(_email, _password);

        yield return new WaitUntil(predicate: () => registerTask.IsCompleted);

        if (registerTask.Exception != null){
            FirebaseException firebaseException =
(FirebaseException)registerTask.Exception.GetBaseException();

            AuthError error = (AuthError)firebaseException.ErrorCode;
            string output = "Unknown Error, Please Try Again";

            switch (error){
                case AuthError.InvalidEmail:
                    output = "Email is invalid!";
                    break;
                case AuthError.EmailAlreadyInUse:
                    output = "Email is already in use!";
                    break;
                case AuthError.WeakPassword:
                    output = "Password is too weak";
                    break;
                case AuthError.MissingEmail:
                    output = "Please enter your email!";
                    break;
            }
        }
    }
}

```

```

        case AuthError.MissingPassword:
            output = "Please enter your password";
            break;
    }
    registerOutputText.text = output;
}
else{
    UserProfile profile = new UserProfile{
        DisplayName = _username,
    };

    var defaultUserTask = user.UpdateProfileAsync(profile);

    yield return new WaitUntil(predicate: () => defaultUserTask.IsCompleted);

    if (defaultUserTask.Exception != null){
        user.DeleteAsync();

        FirebaseException firebaseException =
(FirebaseException)defaultUserTask.Exception.GetBaseException();

        AuthError error = (AuthError)firebaseException.ErrorCode;
        string output = "Unknown Error, Please Try Again";

        switch (error){
            case AuthError.Cancelled:
                output = "Update User Cancelled!";
                break;

            case AuthError.SessionExpired:
                output = "Session has expired!";
                break;
        }

        registerOutputText.text = output;
    }
}

```

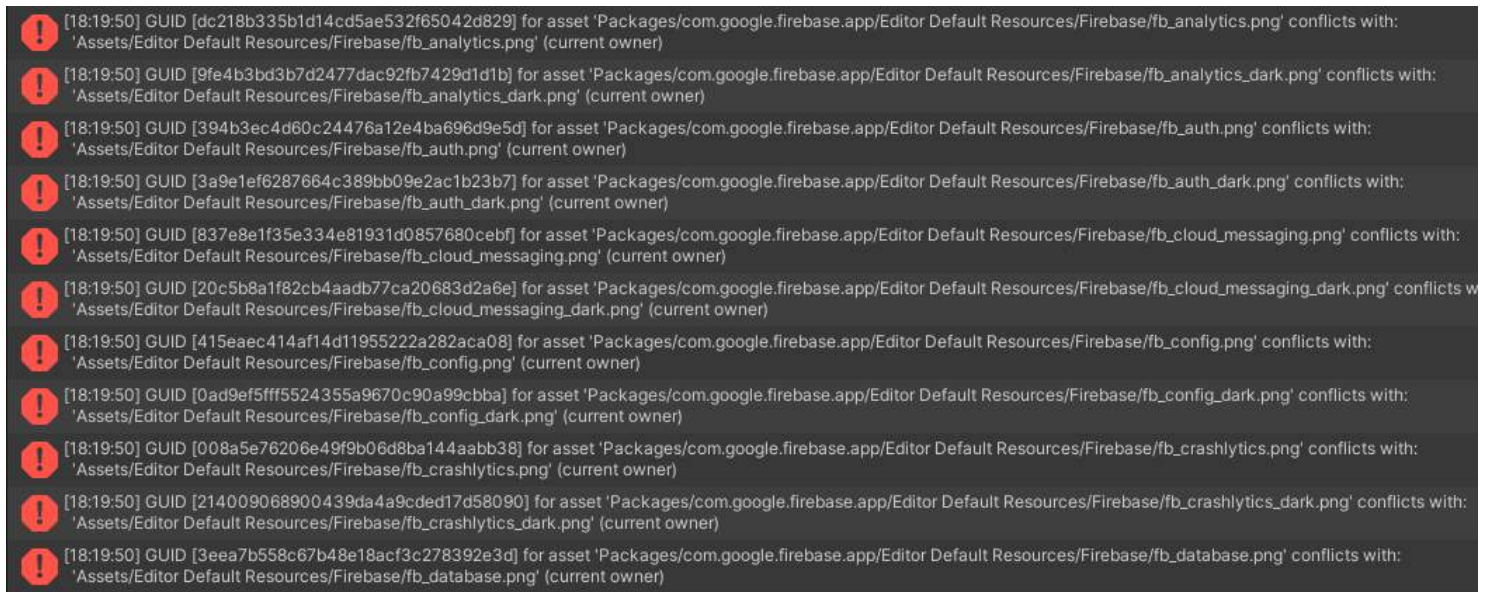


```
SceneManager.LoadSceneAsync(_sceneIndex);  
}  
}
```



## Firebase Errors

After importing the packet manager, we see that the Package conflicts with the Asset EDR. This can be fixed by deleting the contents of the EDF as they are already in the Package.



Another error has been starting to spread among multiple scripts within the Menu Scene is that the script cannot be loaded. This is a harder error to track as it does not show up in the console. There are no compile errors, and the script is within the assets folder so there is no reason why the script cannot be fetched.

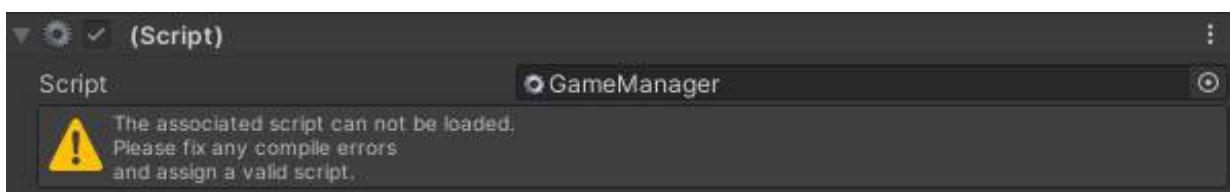
Deleting the Meta files of the scripts and trying to reimport the scripts while also reloading the scene does not seem to help.

I was also unable to find any fixes as the vague has no error code and can be due to many things on Unity's side of the processing.

I ended up downgrading the version of Unity to the oldest version of the program I can o while also keeping the Package Manager feature, which is 2020.1.17f1. This also did not seem to do anything significant.

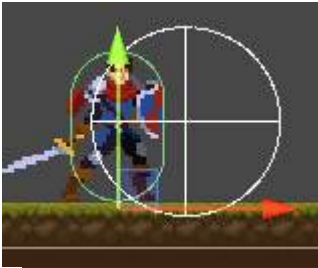
Before, this error showed up as the filename and class name not matching which is also false. This error seems to have no fix, leading to the ultimate decision of pausing the idea of Implementing firebase in order to keep going with the rest of the development.

If I will have time later in the development stage, I will give firebase another change. I couldn't find other alternatives that would make as much sense as Firebase for a game. The other solution would be to still use firebase but use the google authentication system instead of the email and password combination, due to it being supported very well along side other unity games, especially on mobile.

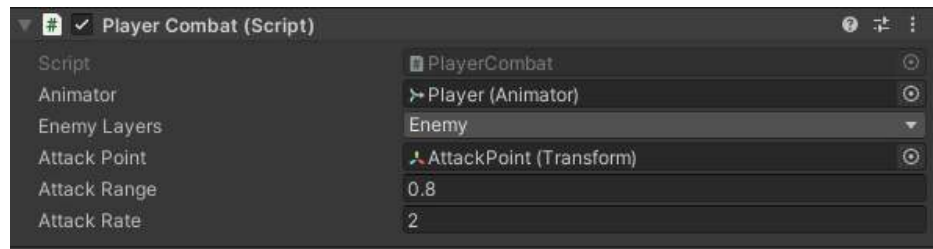


### Melee Combat Player

To make the combat system we need to give our player a new script that would simulate the tip of the sword and the collider of the enemy. If these two points are overlapped when the player makes their attack, then the enemy will take damage. This can be simulated as the point having a radius and whatever enemies there are within that circle will take a certain amount of damage.



This will give the effect along with the script and animator that the player does damage to the enemy as long as the enemy is in front of the player and



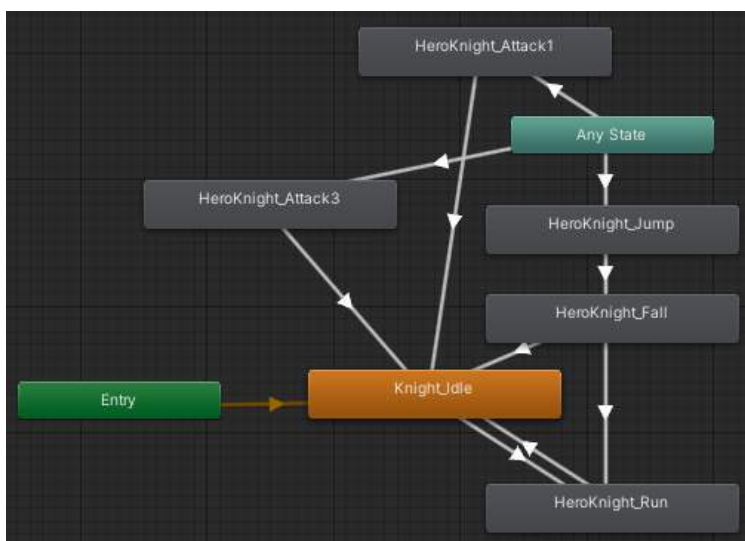
within the attack range.

This is the combat game component of the player made up of the script itself. It's main purpose is to take the players input in this case being the LMB and cause damage to the enemy in two stages. The swing animation happens regardless, only when he LMB is clicked. The enemy takes damage only when they are within the range, or the circle drawn you can see in the scene view.

Another important aspect of the melee combat that needs to be regulate is how often the player can choose to attack the enemy. Spamming the LMB would result in them doing more damage to the enemy than it should be allowed, not even the animation could keep up and so the attack can only be run if enough time has passed until the last attack. This would make the melee combat more interesting to learn.

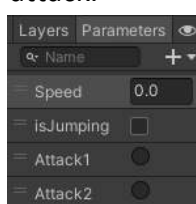
All of this is of course an ad on to the current player. The extra animation also changes the animator, and adds another parameter, which is a trigger, called attack. This is activated only when the player presses the LMB and it triggers the attack animation.

### New Animator



The animator is similar what it used to be before. The only thing it has changed is the addition of the attack animation. Here I added two types of animation, represented by the LMB and the RMB.

The animations although very similar, as well as the amount of damage done by each attack.



These are also the parameters used to trigger all of the new animations.

### Player Combat Script

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
public class PlayerCombat : MonoBehaviour
```

```
{
```

```
    public Animator animator;
```

```
    public LayerMask enemyLayers;
```

```
    public Transform attackPoint;
```

```
    public float attackRange = 0.5f;
```

```
    public float attackRate = 2f;
```

```
    float nextattackTime = 0f;
```

```
    // Update is called once per frame
```

```
    void Update()
```

```
    {
```

```
        if (Time.time >= nextattackTime)
```

```
        {
```

```
            if (Input.GetKeyDown(KeyCode.Mouse0))
```

```
            {
```

```
                Attack1();
```

```
                nextattackTime = Time.time + 1f / attackRate;
```

```
            }
```

```
            if (Input.GetKeyDown(KeyCode.Mouse1))
```

```
            {
```

```
                Attack2();
```

```

        nextattackTime = Time.time + 1f / attackRate;
    }
}

void Attack1()
{
    animator.SetTrigger("Attack1");

    Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(attackPoint.position, attackRange,
enemyLayers);

    foreach (Collider2D enemy in hitEnemies)
    {
        enemy.GetComponent<Enemy>().TakeDamage(20);
    }
}

void Attack2()
{
    animator.SetTrigger("Attack2");
}

void OnDrawGizmosSelected()
{
    if (attackPoint == null)
        return;

    Gizmos.DrawWireSphere(attackPoint.position, attackRange);
}

```

## Enemy

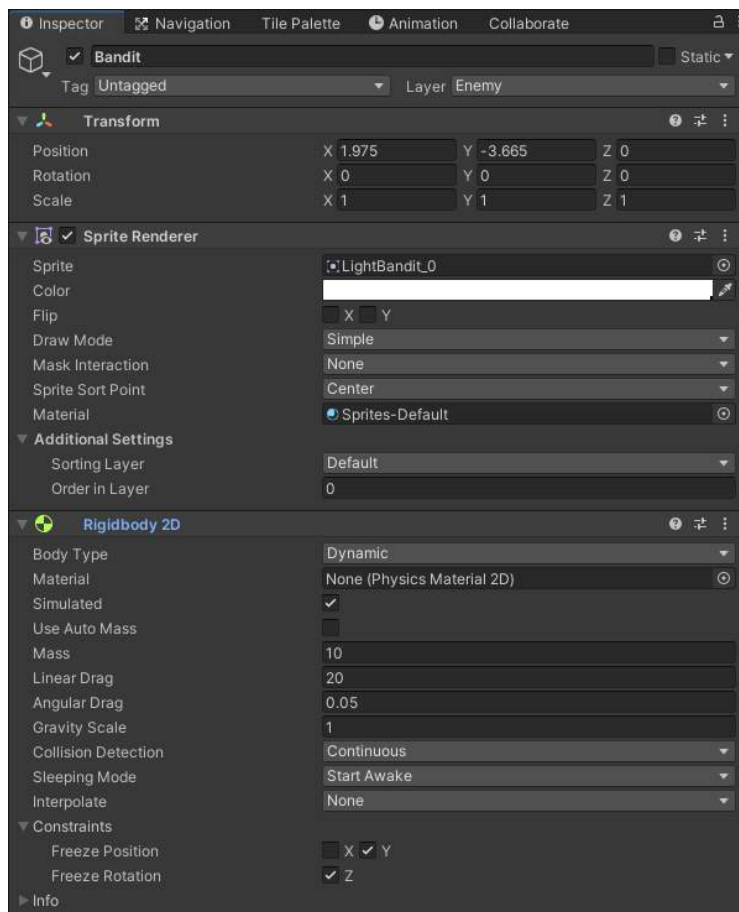
In order to make our game somewhat fun, we need to give our main character someone to fight. Creating the enemy is similar to the player in their core mechanics. They are a rendered sprite/idle animation that has an animation controller, some other components for the physics of the character and CS scripts that determine what they do.

Sprite Resource:



This is the basic idle state of the enemy/bandit. Rather than making all the animations, I picked this character instead for a couple of reasons:

- He is almost the same pixel height and detail as the main character (They do not look like they are from two completely different games)
- It already has enough animation done to him on the original sprite sheet.
- He appears to have the same body proportions as the player, and the same rate of speed when it comes to movement (if the rate of change in the sprite is the same between the two), as they have the same no of images to depict a run animation, or a slash, etc.



Before adding the scripts, we have to set up his Physics and Layer system

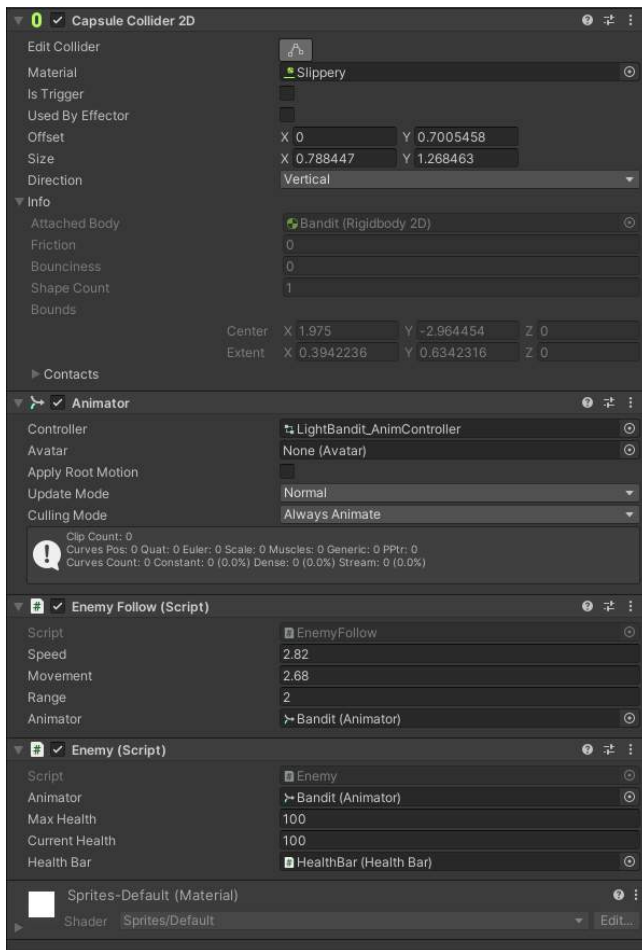
Firstly we see that he is on a different layer, in order for the player and all Enemies to interact naturally.

We also see the RigidBody component, giving our character the characteristic of a game enemy.

The rotation in the Z axis is frozen so that the game doesn't show weird bugs, but also the movement in the Y axis.

Giving our character the ability to jump is not a necessity. That should be the players advantage over the enemy. This is also much harder to implement into the Enemy's AI as jumping is only useful in specific situations.

We also need the continuous collision detection for later implementation of the melle combat.



Here we have the Capsule collider that outlines the character. He also has the slippery material on the y axis, if jumping will ever be implemented to the enemy characters.

The animator takes care of the characters animation controller. The animation is just as complex as the players animation thus far.

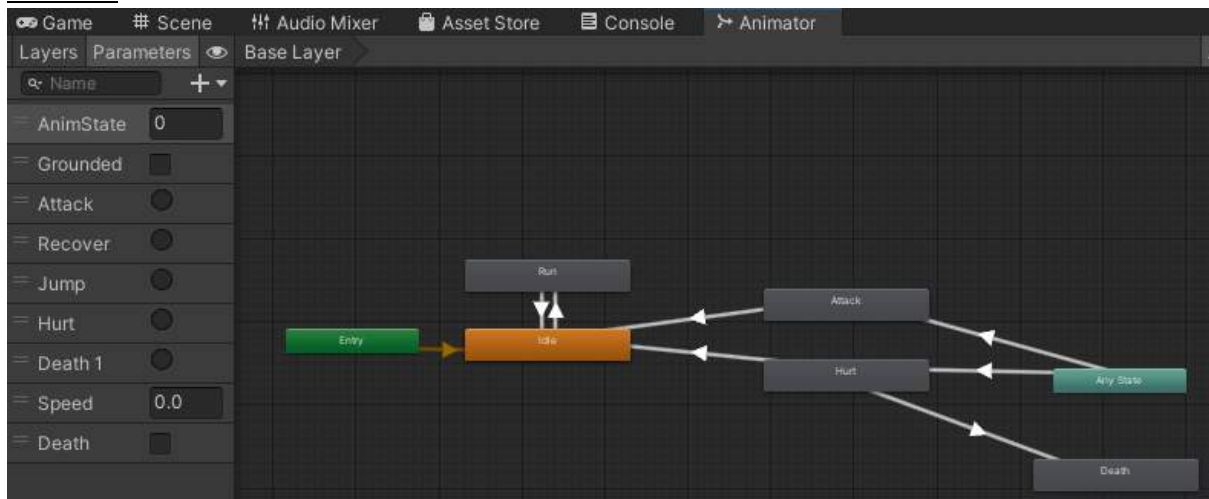
We then have two scripts, the Enemy Follow, this script makes the Enemy follow the player wherever he goes, and the Enemy script that takes care of the enemys health stats, along with their die function for when they're health reaches 0

We can also see the variables in those components. We have max speed that the player can run at and current speed which is down as movement.

We also have the range, so how far away the player has to be for the enemy to stop. This is usually the attack range which will later be implemented.

Aswell as linking the animator to both of these scripts in order to change the animato parameters and change the anmiation according to the enemies actions and health stats.

## Animator



These are all the animations we need for the enemy, along with the attack animation being an extra. We see that idle is the default state, that can then transition to the run animation if the characters speed is increased and vice versa.

The health bar will also be later implemented in order to visually show the enemies health going down.

### Enemy Follow Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyFollow : MonoBehaviour
{
    public float speed;
    public float range;
    private Transform target;
    public Animator animator;

    // Start is called before the first frame update
    void Start()
    {
        target = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>();
    }

    // Update is called once per frame
    void LateUpdate()
    {
        if (Vector2.Distance(transform.position, target.position) > range) {
            var targetPos = new Vector2(target.position.x, transform.position.y);
            transform.position = Vector2.MoveTowards(transform.position, target.position, speed *
Time.deltaTime);
            Move();
        }
    }

    void Move()
    {

```

```

        animator.SetFloat("Speed",speed);
    }

}

```

### Enemy Script

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour
{
    public Animator animator;
    public int maxHealth = 100;
    public int currentHealth;
    public HealthBar healthBar;
    // Start is called before the first frame update
    void Start()
    {
        currentHealth = maxHealth;
        healthBar.SetMaxHealth(maxHealth);
    }

    public void TakeDamage(int damage)
    {
        currentHealth -= damage;
        healthBar.SetHealth(currentHealth);

        //Play hurt animation
        animator.SetTrigger("Hurt");
    }
}

```



```

    if (currentHealth <= 0)
    {
        animator.SetTrigger("Hurt");

        Die();
    }
}

void Die()
{
    //Die animation

    animator.SetBool("Death", true);

    animator.SetFloat("Speed", 0f);

    //Disable enemy

    GetComponent<Collider2D>().enabled = false;
    this.enabled = false;
}
}

```

### Sprite Sheet



This sprite sheet can be split into the multiple animations and used on our Enemy character to get the desire movement/ animation.

There are more than enough options on animation allowing the integration of later features such as a revive/recover animation, more attacks and idle animations that can be used later in order to make the game more immersive and more fun to play.

The sprite sheet is very similar a the one used for the main character.

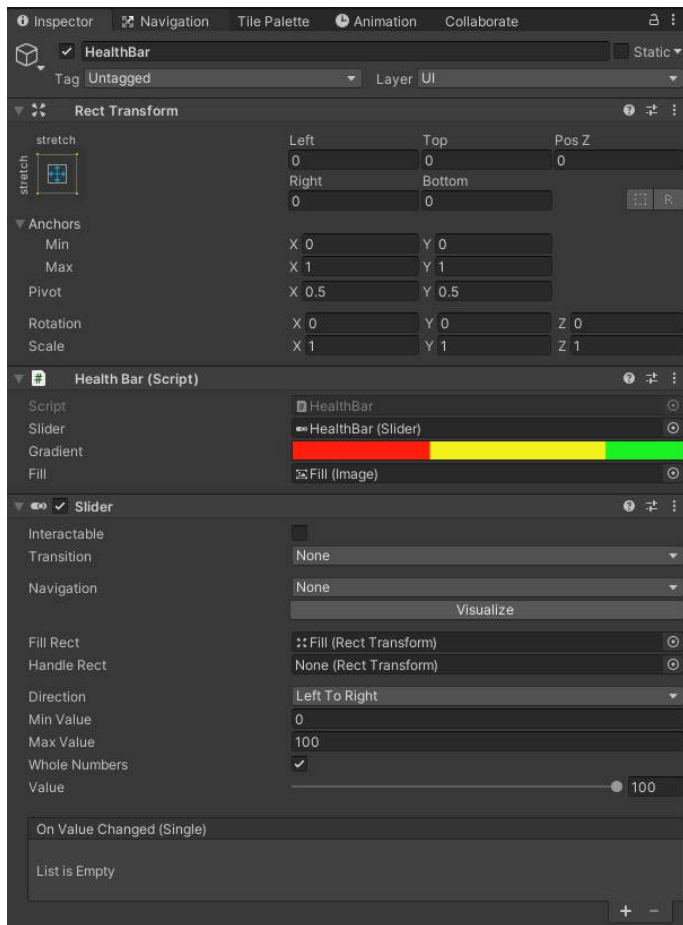
## Health Bar



This is another feature of the enemy character as each enemy has their health displayed over the sprite as a health bar. This health bar is made up of a fill and a border that act as a slider which we can control from within the script linking the players health and the health bar.

This health bar also naturally always follows the enemy as it is part of the same parent game object.

Not only that, but the health bar is on the UI layer and so it won't affect other game objects that are on the different layers.



Here we can see the components of the Health Bar parent game object.

The child game objects, the fill and the border are called in this game objects Slider component which allows me to resize the fill to represent the health going down.

This is changed by the Health Bar as the slider value is made equal to the characters health, which is calculated as the no of hits multiplied by the damage per hit that the player does to the enemy.



Here we can see the health bar of the enemy going down as we do more and more damage to them, starting out at green and turning to yellow then red.



## Health Bar Script

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
using UnityEngine.UI;
```

```

public class HealthBar : MonoBehaviour
{

    public Slider slider;
    public Gradient gradient;
    public Image fill;

    public void SetMaxHealth(int Health)
    {
        slider.maxValue = Health;
        slider.value = Health;

        fill.color = gradient.Evaluate(1f);
    }

    public void SetHealth(int Health)
    {
        slider.value = Health;

        fill.color = gradient.Evaluate(slider.normalizedValue);
    }
}

```





## Evaluation



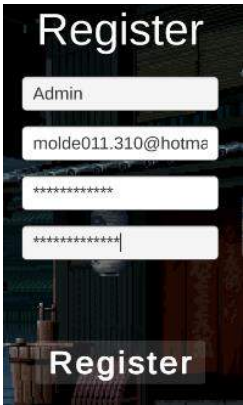

Testing of the program has been done mostly during the development process as Unity allows you to run the program in game view and quickly test the new aspects you may have tried to implement. This allows me to test during the development cycle and without telling unity to build my game before I can run it. Gives a whole aspect of trial and error to the coding cycle, especially for some small game object component settings that may give an error in the console.

These tests are mostly done to assure the client that the game is running the same as it was in the game view of unity before it was built.



## Testing




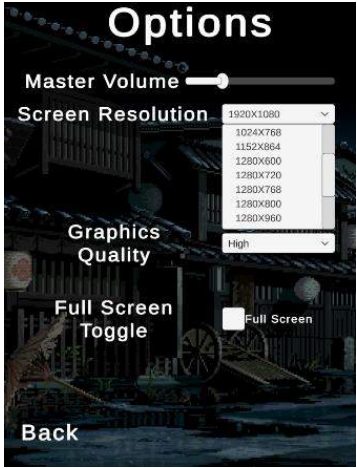
### Log In System Tests



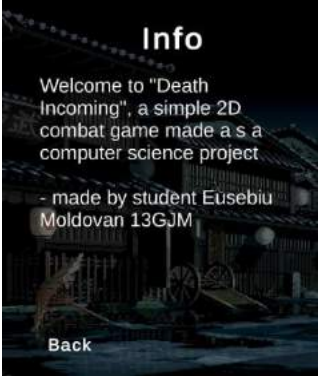

No.	What is being tested?	Input	Expected Outcome	Actual Data	Pass/ Fail
1	As the game is opened this is the First parent UI game object that should show up in the first scene	N/A	This menu should show up first and should be impossible to go past it without registering or logging in.		Pass
2	Do the fields for email, password etc work?	Try and enter some details.	The data entered should remain within the field until you chose to enter your data.		Pass
3	Is your password hidden, so all characters appear as an '*' instead	Enter some details in the password fields.	This should protect the users password		Pass
4	Pressing the login button should allow the player to play the game	Input the correct details	Once clicked with the correct data, the game proceeds to the game main menu		Pass

5	Entering the wrong details would not allow the player to proceed	Input the wrong details	The game doesn't proceed and asks you to try again	 <p>Switches to</p> 	Fail
6	Making an account and pressing the Register button should allow you to proceed.	Input new details in the register section	The data entered is then taken to firebase and stored.	 <p>Nothing was added to firebase. The email, password, etc. have not been added to firebase and cannot be used to log in.</p> 	Fail
7	Logging back into the game with the new details should allow you to access the game	Input the new details in the log in section	The game should then take you to the main menu giving access to the rest of the game.	Cannot happen as the accounts have not been saved, also switches to the game no matter what details have been entered.	Fail

### Menu Tests


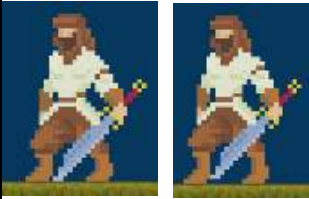




No.	What is being tested?	Input	Expected Outcome	Actual Data	Pass/Fail
1	Game should go to the main menu after loading, and displays it to the user.	Run the program	It does as specified.		Pass
2	The buttons should darken when hovered over them	Hover over all the possible main menu buttons.	It does so, as well as going back to the original colour.	 <p>The Play button here is darker, but it has a level of subtlety due to aesthetics.</p>	Pass

3	The play button should lead to the actual gameplay scene when clicked	Click the play button	Does as expected, lead straight to the gameplay with almost no loading time	 <p>Switches to</p> 	Pass
4	The options button should lead to the option menu without changing the scene.	Click the options button	It does not change the scene, it only deactivated the main menu parent game object, and activated the options menu.		Pass
5	The options menu should have multiple settings such as drop down, sliders, etc.	Open options menu	It has all those things, and they are all easy to interact with by the user. Opening some of the settings		Pass







6	These settings also change the game itself so they should have an effect on the game.	Change some settings	I expect those changes to actually apply to the game when they are changed.	 <p>The settings do change the actual game settings within the build.</p>	Pass
7	The player should then be able to exit the options menu by clicking the back button.	Click the back button from the options menu.	Very similar to all the other buttons, just smaller and once clicked, it takes you back to the main menu.	<p><b>Back</b></p> <p>Switches to:</p> 	Pass
8	The user can access the info menu from the main menu.	Click on the info menu	Once accessed, the user can see a text message with some details about the game		Pass
9	They can also use the back button from the Info menu to regain access to the main menu.	Click on the back button.	The button should work in the same way as the button that is in the options menu.	<p><b>Back</b></p> <p>Switches to:</p> 	Pass
10	The player can chose to quit the game by pressing the quit button from the main menu.	Click the quit button	Once clicked, this button will close the game and brings the user back to their home desktop as the game is now closed.	<p><b>Quit</b></p> <p>Quits the game without any message, goes straight back to desktop.</p>	Pass





Gameplay Tests

No.	What is being tested?	Input	Expected Outcome	Actual Data	Pass/Fail
1	Does the world along with the tiles, ground etc., show up on screen	Play the game	All the different tiles show up on screen just like the game view	 <p>All of the expected sprites and tiles are loaded into the game.</p>	Pass
2	The enemy character on the screen is in the idle animation	N/A	The enemy sprite renderer animates the enemy correctly	 <p>His idle animation mostly just a shoulder raises to simulate breathing.</p>	Pass
3	The enemy starts running towards the player as soon as the game starts	N/A	Does as expected, also enemy stops right in front of the player, within the attack range.	 <p>Here is the enemy sprinting towards the player, ready to fight.</p>	Pass
4	The enemies health bar is also rendered above the enemy	N/A	The health bar is full and generated above the enemy	 <p>The health bar appears full in the beginning and follows the enemy</p>	Pass
5	The health bar also follows the enemy everywhere the enemy goes	N/A	The health bar follows the enemy along x axis and stays in the same y coordinate		Pass
6	The enemy plays the hurt animation when the player damages the enemy.	Swing at the enemy.	The animation is played followed again by the idle animation.		Pass



7	The health bar is also decreased by a specific amount when health of enemy is decreased.	Damage the enemy	The health bar decreases by 20 % of full due to damage dealt by the player.		Pass
8	Health bar changes colour depending on the enemy's current health.	Damage the enemy	The health bar will change from green to yellow to red as more damage is dealt to the enemy.		Pass
9	The enemy follows the player around	Run away from enemy	The enemy should follow the player, although being a bit slower than the player.		Pass
10	The enemy also dies once its health reaches 0 or below	Kill the enemy	The enemy should fall to the ground and remain there		Pass
11	Is the main character on the screen in the idle animation	N/A	The sprite renderer shows the character as idle.	 <p>The character is in idle animation, here are two slightly different sprites that appear on after another (the cape/scarf changes)</p>	Pass
12	The character can move across the given space.	A/D	Pressing A/D would result in the player moving, as well as the specific running animation playing for as long as the player is running.		Pass

13	The player can jump	W	The player will be going into the air and drop back down while the jump animation plays.		The jumps are small, and the character quickly comes back on the ground.	Pass
14	The player should also appear to swing to the enemy	LMB	The player swings his sword and damages the enemy if in range			Pass

### Testing against Features of Solution

Requirements	Explanation
A simple main menu that is intuitive and easy to use for newcomers but also has enough settings for more experienced players	This will allow both low and high skilled players to feel right at home
This requirement has been met, the menus are simple and intuitive, as well as being fairly polished due to their simplicity. The buttons do react to the player clicking and hovering over them so that the user can tell what is happening more clearly.	
The design scheme reflects the story and the main character being the Grim Reaper.	This give the game some personality to be associated with, e.g. Minecraft has a 3d square as a logo that resembles dirt and grass.
The sprite of the main character has been changed due to requiring more animation that I struggle to make in pixel art, although the original idea of having the game reflect the main character is still met as the enemy type has also been changed to fit with the new main character. The games new icon can actually be the players shield as it is one of the main pieces o the sprite.	
The whole theme should be dark along with the writing in the menus, backgrounds, sounds, etc.	This goes along nicely with the meaning and feeling of the whole game, which is to be expected.
This has also been met, as the main menu and the whole game use darker sprites for the aesthetic. The background is a dark blue to simulate the night sky, as well as having the other tiles being pixelated and yet have less vibrant colours.	

Most of the menu related requirements have been met mostly because they are simple and easy to implement, as well as having so few. Unlike some of the other requirements relating the gameplay, the menu was much simpler, and harder to improve upon. The only improvement that can be done is add some sound as well as animation to the buttons making the menu much more alive than what it currently is.

Requirement	Explanation
The usual game setting that can be tampered with so that every user can personalize their experience to a certain extent. Includes controls, graphics, gameplay etc.	This will allow higher skilled users to personalize their experience the way they want it, but it is also good for people with more special screen and so on to run the game optimally.
Although though is done to a lesser extent than expected, it is integrated in the game as the changes made do actually affect the game and its performance.	
The game should also have a pause menu that allows you to change some of the in-game settings and maybe even a tip section for those that are lost and do not know what to do	This will allow the users to personalize the game experience but also stop their progress for a little break if they so wish.
The pause menu is not integrated as I did not have enough time to do so in the development section and the need to step away from menus. Making one would definitely improve my game, giving the user more freedom when it comes to	
The ability to change key binds would be a good thing to add	This can mean that users can customize their experience even more and could allow room for connecting a controller later on.
Changing key binds is not a feature I have added to my game yet. My skill set is very limiting and so adding this to my game is still a new concept to me and so I wasn't able to do this in development.	

A save button, although the game will save automatically, having the option to do so manually can prove to be beneficial and some people prefer it.	This will allow users to ensure that their progress is saved after they are done playing for the day.
There is no progress to save and so making a save menu for no reason seems like a waste in time at this stage of development and so this has not been done, although once the game will become more complex and the progression system will be made up of more levels and such, a save system is necessary.	
Having a cheat menu or just cheats to add some more fun to the game.	After the user has finished the game, they can replay it using cheats which might just make it more fun and hilarious.
The game is not yet complex enough for a cheat menu to have any real use. The game is easy and there aren't many things the player can think of making easier for themselves. Once resources and more complexity has been added, then a cheat menu would make sense.	

A lot of those requirements have not been met due to either the lack of time spent in development or the complexity of my initial idea along with my lack of skill when it comes to unity, and coding in C#. This inexperience shows itself in the things that I am not yet able to implement.

Requirement	Explanation
Standard computer peripherals: Computer with a keyboard mouse and monitor.	The user needs basic peripherals in order to play the game and to progress through it.
The game so far can only be played using a mouse and keyboard, although it can easily be made suitable for a console controller, and even mobile phones, but it has not been integrated yet in the Input Manager system.	
Preferably headphones or speakers for the sound effects, although it shouldn't affect the user performance a lot.	This is not a must, but it is recommended to get the full game experience.
There are no game sounds implemented as that is a very new concept to me and my skill is too limited to be able to integrate this efficiently in my game.	
Minimum Computer Specs: (enough to run the software)	The computer needs to be able to run the game. Although not a lot of image processing is needed, some still is.
The game is very easy to run, not only on my system but also other systems. It takes very little processing power out of my computer to the point that leaving the game open is hard to tell, consuming no more than 3% of my processing power, ensuring that other people with a lesser version of the i7-7700k can also run this game with ease.	
Windows, Mac or Linux operating systems	In order to run the program and to store the data, you need an operating system.
The game will only run on a Windows computer/laptop due to this being the only module I used to build the game for, from unity.	

The game is easy to run on almost any system because it is such a simple game. Making it run on mobile phones would then be easy because of how easy it is to run. In order to build the game for these platforms, the modules have to be added, as well a new input system, especially for the mobile phones users.

### Questionnaire

Henry

The menu is very simple and easy to navigate. It feels like I have seen it before because it is so similar to other games, most people expect for something like this to be the case.

The game play is nice with smooth animations, and the controls are extremely intuitive for me, and even for beginners.

The games lacks a sense of progression which is why spending a lot of my time on this game seems impossible as there is rarely a feeling of satisfaction when playing it.

The game has a lot to improve upon. It has an interesting concept but comes too short for me to be replayable.

Danika

The menus are easy to navigate through, easy enough even for me.

The gameplay is lacking but I am a fan of the design style of the characters and the rest of the game. Pixel art is new to me and I found it to be extremely underrated. I would love to see more games like this.

The game play seems nice, although the progression is lacking. The characters seem somewhat real in their interactivity and very responsive to my inputs.

For me, as a casual gamer, it is hard to spend time playing this game due to it's lack of progression.

Andrea

The menu system is super easy to follow, although I do not like the dark theme the game menus have. I personally prefer something more vibrant and animated.

The gameplay and animations blend well. The pixel art style is very appealing, aswell as having the animations is this style give the game a sense of uniqueness I was never interested in before. This game definitely made me a fan of pixel art.

There is no progression to the game so it very hard for me to invest time into the game and play it more than maybe a few minutes, let alone streaming it which I don't think would appeal at this stage.

A lot of improvements can be made.

Conclusion: The focus seems to be the lack of progression. Focusing on this in later version would seem to extract the most value out of the game, as it would attract more people. The idea of pixel art seems to be appealing when put into a game.

The menu could also have some work done to it, but the main focus would be adding more complexity to the game so that the players would invest more time into it.

### Maintenance and Limitations

There are limitations to my game currently as I have not achieved my initial idea as development is not finished. These are bugs, inconsistencies, things not included and improvements that can be made and should be made to the original game.

### Improvements:

- The log in menu system is not functional, and so making it functional would help a lot with people being able to make an account in order to help the game, regulating people playing the game but also saving progress.
- There is a story lacking as it has not been yet made using the tile system of making different levels and a progress through different worlds. This can and should be implemented but it is very time consuming which is why it has not been done in development.
- The enemy AI is very simple and basic. It does not even hit back on the player and so the game is easy to win and progress through. Making the AI harder and adding multiple types of enemies that have different attacks would make the game more challenging as it builds a learning curve as otherwise the game cannot progress, and the player will struggle unless they learn.
- Because there is not much of a progression through the game, there is nothing to save so there is also the lack of a save system. Adding more complexity to the current game would be the first step, and then make a saving system in order to ensure that the user can pick up from where they've left off.
- The level is also basic. Adding more objects from the asset pack could make it seem much more alive and interesting.
- Adding a combat system to the enemy so that the player has more of a challenge while fighting. The addition of a learning curve to the game would make the game more challenging and so players would invest more time trying to learn the game.

### Complexity:

- Add the special ability to the main character, and the cooldown system.
- Allow the player to block enemies' attacks with their shield as well as having the ability to roll.
- Add some other platformer aspects, such as sliding down walls.
- Give the user an inventory menu and different resource management that the player can collect from the different levels.
- Add more effects using the particle system.
- Add the log in system using firebase.

### Bugs:

- One of the first bugs is that the enemy when it runs towards the player, it jitters. It is very subtle, but the movement is not as fluid as I would like it to be. Once the Enemy might get a more improve AI, even using a path finder instead of a simple follow script, this issue could be fixed.
- Another bug is that even though the player appears to be on the floor and should not affect the player, it still follows the player around. In the die() function the follow script is deactivated but for some reason the enemy still follows the player. This can also be fixed after the new AI would be integrated.
- While the game is in windowed mode, changing the resolution of the game, changes the actual size of the window to fit those pixel sizes. This does not happen in full screen

however, which is what I expect most of the players to be using but it is still an issue that needs fixing.

- Less of a bug but should be fixed is that the enemy health bar still appears after the enemy has died which is fine but will not look good after more enemies are added to the game.
- The sprites used for the health bars look glitchy. Maybe due to the drop in or change in size but they do not fit the rest of the game that looks clean in the pixel art style.
- Spamming the W key for jump makes the animation glitch as the players animation shows the player to keep going up although that is not the case.

