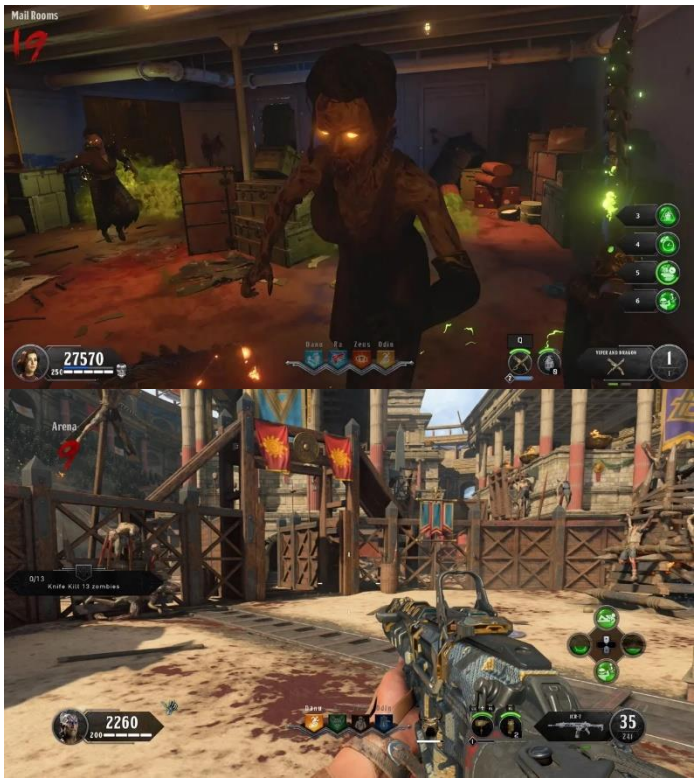
Analysis

Introduction

My general idea for this project is to make a 3D zombie survival game in Unity. My main inspiration for this is the Zombies game mode in Call of Duty of which you must survive endless rounds of zombies getting increasingly tougher each wave, earning points each kill which you can use to buy weapons and powers. However, I plan to make the combat more melee/close combat focused with possibly the ability to pick different characters all with different weapon types and powers. I also plan to add a main menu with the ability to pick different maps and settings which allow you to customize difficulty, controls, and other common settings you would typically get in FPS games. To achieve this, I will need do more research on using unity, creating enemy ai, creating a wave system, combat mechanics and on all the assets I will need. I will also need to consider what my stakeholders want; I could do this through a questionnaire asking people what key features they want in the game.

Research

One game which is like my project is the zombies gamemode in black ops 4



Features I would like to include:

- First person perspective (although may add ability to switch to third person)
- Mystery box randomly placed across map and gives you random weapons
- Ability in later waves to upgrade weapons at some sort of station randomly placed across map
- Wave and point mechanics – Endless waves getting tougher after each one, earning points after each kill which can use to buy weapons/powers
- Layout for HUD

Features I would not like to include:

- Map not fully accessible from start as I do not plan for my map to be too big and would be a bit annoying to implement
- Art style – plan to go for more simplistic low poly look and I do not have the time nor resources for it
- Gun focused combat – want to make it mostly melee/close quarters
- Bot followers – plan to make difficulty scale with number of players

Another game which is like my project is Left 4 Dead 2

Features I would like to include:

- Interactive environment – can destroy windows, move objects etc...
- Fast paced combat – ludicrous number of zombies all with reasonable speed running at you
- Healing items and ammo scattered across the map
- Objectives to complete – although will make these like random occurring events
- Wide variety of zombies, with ability to climb, unique benefits, etc.
- Dark parts of map that require a torch

Features I would not like to include:

- Campaign and levels as I am plan to make the game consist of one gamemode which lasts indefinitely

	• Art style – more cartoonish, simplistic graphics as more focused on functionality as more realistic graphics would be a lot more time consuming

Another game which is like my project is Elden Ring

	Features I would like to include: • Ability to cast spells • Given a set number of flasks which you can distribute to health or utility (opposed to health and mana) • Variety of bosses which are tough, have unique move sets and different stages (will make these boss fights occur after reach a particular wave) • Boss bar appears on screen in boss fights • No natural regen (may make it so this only occurs after waves/boss fights) • Ability to find talismans with special effects • Medieval, magic kind of weapons (although will include some more modern ones too) Features I would not like to include: • Open world • Storyline/quests • Slow-paced movement and combat (may not have stamina bar and have unlimited sprint) • Armour, coins, other rpg items • Enemies drop loot • Mounts

Stakeholders

The target audience for my game will be males as after looking at numerous reliable sources such as Wikipedia fps type games seem to have less than 10% female players. I will also target those from ages 13 to 25 particularly those who are into fps games or survival ones where you must survive endless waves. This is because the game will consist of first person fast-paced combat, with low-poly graphics with possibly a bit of gore making it suitable for younger ages. They will be playing the game in their free time when they are bored. I will make use of opportunistic sampling to conduct my research on desired features in the game. I can achieve this by asking students in my class alongside family members and friends to test the lower and higher ends of the age range.

Requirements

	Feature	Sub-features	Explanation	Justification	Importance
1	Player movement	WASD movement	Player will be able to move forwards, left, backwards and right using the W, A, S and D keys, respectively.	In all 3 games I analysed and almost all modern pc games utilise WASD for basic player movement.	Essential
		Movement abilities	Player will have the ability to jump using the spacebar and sprint indefinitely using the left ctrl key as well as the choice of either a dash or roll ability using the F key.	Most if not all games nowadays will give the player the ability jump, sprint, and some form of dodge ability. And a sprint ability more so than others is essential to make the game feel more fast-paced	Essential
2	Player Health		Player will start off with 500 health which can only be recovered via a health flask or after a boss wave ends	Ensures that player cannot play indefinitely and is required for the game to be a challenge to survive as many waves as possible.	Essential
3	Player Items	Melee weapons	Melee weapons will either be 1 or 2 handed, all with a light attack which can be used by tapping the LMB and heavy attack which can be used by holding the LMB. Most 2 handed melee weapons will also have the ability to block by holding the RMB and push enemies with the LMB while in the block state, some 1 handed melee weapons will also have the ability to be thrown by holding the RMB to aim and tapping the LMB to fire.	For the game to have a heavy focus on melee combat, melee weapons are essential.	Essential
3	Player Items	Ranged Weapons	Ranged weapons will all be 2 handed, with limited ammo and use the RMB to aim and LMB to fire. Weapons with	Provides player with a reliable method of dealing with enemies from afar.	Essential

			bullets for ammo will do additional damage for headshots.		
		Wonder Weapons	This weapon will be exclusively available from a weapon mystery box and will take up both the melee and ranged weapon slots, players will be able to select their desired wonder weapon in the loadout section on the main menu	Provides player with a more unique powerful weapon they can obtain every now and then due to it being harder to obtain.	Desirable
		Magic	All spells will have a time required to cast (in which the player will be unable to use items or sprint) as well as either a limited number of uses, limited number of uses per wave and or a cooldown between each use. Each spell will either have some sort of support ability e.g. heal player, make a shield to block enemies or an attack ability e.g. shoot fire at enemies, freeze enemies, etc...	Provides player with a way to hinder/stagger enemies as well as temporarily buff themselves	Essential
		Upgrades	Players will be able to start upgrading weapons after they clear the 1st boss fight, they will be able to do this at a magical forge which will appear at a random point in the map appearing somewhere else after a set number of uses. When upgrading their weapon, it will cost the player a certain number of points depending on how much it has been upgraded and how much the weapon is worth. When upgrading it will take 45s for each time it has been upgraded to complete, e.g. 1st time no wait, 2nd time 45s, 3rd time 90s, etc... Only melee, ranged and wonder weapons can be upgraded.	Is required for later rounds of the game as otherwise player will not have enough power to kill of enemies as they gradually get stronger each wave	Essential
		Talismans	Players will be able to equip talismans which in doing so will provide the player with passive benefits. Some with stronger benefits but downsides too	Provides subtle benefits to the player to help with their playstyle e.g. if they were using a sniper rifle and you had a talisman like headshots with sniper rifles now cause explosions but do no additional damage	Desirable
		Flasks	Player will start with flasks. 6 flasks which can be split between health and 1 specific utility flask. By default they will all be health, unless configured in	Since player has no natural regeneration and health will only be restored naturally after the end of a	Essential

			the loadout menu. These will automatically replenish after each wave.	boss wave, it is essential to regain health during waves.	
		Loadout	Player will be able to have 1 ranged and melee weapon which they can switch between using the 1 and 2 keys respectively as well as 1 talisman, 2 spells and 6 flasks	Prevents user from just picking up every weapon they find in order for the game not to become too easy and forcing the player to specialise on a particular weapon as well as making it easy to switch between each item	Essential
4	Player HUD		• Top left of screen will display the current wave the player is on • bottom left player health bar with icon of player on left side, • bottom right side player melee and ranged weapons with the weapon currently in use being displayed and its ammo stats below it • to the left of the weapon display is the dash ability which will be represented by a boot icon with some wind behind it and when it has been used will display the cooldown in seconds to 1dp it will also have a display for the key to activate it • above the weapon display on the right side of the screen will be the flasks represented by a flask icon with a different colour liquid for each type and like the dash ability, display the key to activate it on the bottom of the icons and in the top left of the icon the number of flasks, • Finally, to the left of the flasks will be the 2 scrolls with a simple scroll icon for each but a variation of what is in the scroll for each one. and again, it will have the key used to activate it, displayed on the bottom of the icon.	A player HUD is essential to provide the player with necessary information as otherwise the player will have no easy method of knowing how much health they have, keys to press playing game, ammo left, etc... and as such will not know when to do basic stuff like retreat when low on health switch weapons when getting low on ammo have to constantly go to pause menu to familiarise themselves with controls, etc...	Essential
5	Player Perspective		Game will be in first person perspective	For more fast-paced games I believe a FPP would be best especially for ranged combat as it is much easier to aim, and it is also extremely common for zombie survival games to be FPP	Essential

6	Enemies	Pathfinding	Enemies will calculate shortest route and follow it without running into obstacles.	Allows enemies to track and "chase" player around the map	Essential
		Combat AI	Enemies will start to attack player when within range switching between light and heavy attacks	Is a critical part of the game for the enemies to attack the player	Essential
		Unique behaviour	Enemies will have a variety of different behaviours that determine how they move and track the player as well as how they attack them	Makes enemies attacks more unpredictable and provides a unique experience fighting each enemy	Desirable
		Different enemy types	Will be a variety of different enemies with different attributes e.g. some are fast but weak, some slow but strong, some explode on death etc...	In all 3 games I analysed and most games nowadays there is a variety of enemies all of which will require different methods to defeat making the game more challenging and less repetitive for the player	Desirable
7	Bosses	Main	At the end of every 10^{th} wave player will have to defeat a boss which will be significantly more challenging to defeat, after wave ends players, health will regenerate back to full.	In almost all modern fighting games players will encounter a boss after reaching certain checkpoints in the game. These are often difficult providing the player challenge every now and then.	Essential
		Different stages	Each boss will have 2 or 3 stages it transitions to after its health reaches a certain threshold. These stages will partially change how the enemy attacks and possibly giving them new abilities.	In one of the games I analysed, Elden ring, as well as the other soulslike games bosses contain multiple stages which make them far more difficult and require you to adapt to the changes in the boss's attacks when fighting them.	Desirable
8	Main Menu	Design	Main Menu will have some art/screenshots of the game in the background with the title of the game a "Start Game", "Settings", "Loadout" and "Exit Game" button aligned horizontally in the centre of the screen and each component below the one mentioned prior.	Some form of layout is needed for the game to be presentable and easy for the user to navigate.	Essential
		Start Game	A button labelled "Start Game" which when will pressed will send the user to a new screen with a difficulty row and map row as well as play button at the button and go back button in the top left which can also be triggered by the esc key and will send you back to the main menu. The difficulty row will feature columns of images with increasingly more skulls (from left to right) each with a label for the	Is essential for player to play the game as otherwise they have not method of starting it.	Essential

			difficulty name. The map row will instead have columns of images featuring different maps and their respective names (Multiple maps may noy be implemented). Once the player has picked a difficulty and map the player will be able to click a play button and will then be sent to a loading screen with some images of the game a loading symbol and some random facts (possibly) until all the game resources load and then will be promptly sent to the game.		
		Settings	A button labelled “Settings” which when pressed will send the user to a new screen with a bunch of configurations the user can change e.g. FOV, sensitivity, key bindings etc... Each categorised which you can switch between from the category names at the top of the screen there will again also be a go back button in top left which can also be triggered by the esc key and will send you back to the main menu.	Allows user to customise the game to their liking e.g. adjust controls to something they are more familiar with, adjust graphics so it is not too demanding on their device, etc...	Essential
		Loadout	A button labelled “Loadout” which when pressed will send the user to another screen in which the user will be able to customise their starting ranged and melee weapons (from a select number of weapons) alongside their flasks, a dash or roll ability and (possibly) the character they play each option presented as a clickable tile, which will cover the screen each with an image to represent their category and a ‘go back’ button, in top left, which can also be triggered by the esc key and will send you back to the main menu.	Gives player more freedom on their playstyle and a way to change their appearance which almost all games provide as generally users the ability to customise their appearance.	Desirable
		Exit Game	A button labelled “Exit Game” and when pressed will close the application and send the user to their desktop.	Is essential for user to close the game as otherwise game will run indefinitely unless closed by the user externally.	Essential
9	Pause Menu	Design	If the player clicks the esc key while in game the game will remain in a “paused” state and will display 4 buttons “Resume Game”, “Settings”,	Some form of layout is needed for the game to be presentable and easy for the user to navigate.	Essential

			"Exit to Main Menu" and "Exit to Desktop" ordered from top to bottom in that order and aligned horizontally in the centre. The background will be the POV of the player when they paused (possibly blurred).		
		Resume Game	A button labelled "Resume Game" which when pressed will resume the user's session it can also be triggered by the esc key.	Is required for user to resume their play session as otherwise user can only exit the pause menu by returning to the menu or closing the game.	Essential
		Settings	Same as "Settings" button from Main Menu except the go back button sends you to the Pause Menu	Allows player to adjust features after experiencing it in-game without having to wait for the game to end or ending your current session to adjust them	Essential
		Exit to Main Menu	A button labelled "Exit to Main Menu" which when pressed will end your current play session and send you to the main menu	Allows player to return to main menu rather than having to close and then relaunch the game	Essential
		Exit to desktop	Same as "Exit Game" button from Main Menu just labelled "Exit to Desktop"	Allows player to simply close the game from the pause menu opposed to returning to the main menu and then closing the game	Desirable
10	Start of Game		When game begins player will have all 6 flasks and their starting melee and ranged weapons specified in their loadout and will spawn in front of the main building entrance.	Without any weapons players would struggle to kill enemies which are essential to earning points to buy new weapons.	Essential
11	End of Game		When the player dies (reaches 0 health) the game will display the text "GAME OVER" and a button labelled Next will appear in the bottom right off the screen upon clicking the user will be directed to a stats screen displaying the number of waves survived, time survived, number of zombies killed, points earned with a button labelled "Return to Main Menu" in the bottom right with the same functionality of the Pause Menu "Exit to Main Menu" button	Is essential for player to replay game after they die without closing and relaunching their game	Essential
12	Sound	Main	Sound effects for attacks, actions, ui, movement etc... e.g. player reloading, clicking button on one of the menus, enemies getting hit etc...	Makes the game more immersive and overall, a more pleasurable experience	Essential

		Background Music	Some background music suitable for the theme of the game which will become more suspenseful when the player's health drops or will change to a unique soundtrack when fighting a boss	Provides the player with an enjoyable soundtrack that suits the theme of the game.	Desirable
13	Points		Player will get a particular number of points upon each kill depending on the type of enemy which can be used to buy/upgrade weapons	Provides a way for players to buy/upgrade weapons	Essential
14	Map	Main	Player will be limited to a reasonably sized play area surrounded by tall walls on the border to prevent the player from jumping off. Map will contain one main central building and a more open outside area.	Is necessary for game to have a play area	Essential
		Multiple Maps	When starting the game player will have a variety of maps to choose from each with a different season, theme etc...	Provides the player with a more varied experience as well as an alternative if they don't like a particular map	Desirable
		Generation	When the player starts a game, it randomly places weapons, spell chests, the mystery box and magical forge across the map and (possibly) it will generate a new interior for the main central building and environment outside each game although still with the same theme each generation.	Provides player with a fresh experience each player and prevents them from getting more used to the map so they know where everything is located making it a bit more challenging.	Desirable
		Weapons	Weapons will be scattered across the map and purchasable for a set number of points	Provides a way for players to obtain weapons	Essential
		Spell Chests	Across the map, spell chests will be scattered and will require the player to solve a puzzle in order to open. upon opening, players will receive a random spell that they have not received before	Provides a way for players to obtain spells	Essential
		Mystery box	Will cost a fixed amount of points and gives the player a random weapon (melee, ranged or wonder). after a set number of uses, it will disappear, reappearing at a random location on the map	Allows player to gamble their points possibly getting a better weapon as well as the possibility of a wonder weapon	Desirable
		Beacons	The Mystery Box and Magical Forge will have a beacon at their current location	Allows player to find Mystery box and Magical Forge with ease	Desirable

Limitations

Limitation	Explanation
Assets	Would love to have more detailed assets and have everything look the way I want however due to game being 3d and the sheer number assets I will have to comprise using the resources available to me.
Language	As I am only fluent in English, and it is not a necessary feature for the game to function my game will exclusively be in English which will severely inconvenience non-English speakers
Character Customisation	I would like to add in-depth character customisation in which the player has complete freedom to customise the character they play however would be a lot of effort to implement and is an unnecessary feature for the game to function
Weapon Customisation	To provide the player with something to work towards over the long term I would've liked to add an attachment system for each weapon in which the player would have the ability to customise the different parts of the weapon and its functionality e.g. magazine, gun scope, add a retractable wire to a sword so they can throw it and bring it back, etc... However, would require many more assets and a lot more code and is again not a strictly necessary feature
Multiplayer	Most if not all games in this genre will have the option of multiplayer however, it is not a necessary feature and if I were to implement, it separately for a gaming platform like steam, it would be a far too demanding feature to implement

Hardware and Software Requirements

Hardware:

- Game will be 3D so will require a reasonably powerful PC
- Game will either require a mouse and a keyboard for all game controls or a plugged-in controller
- Game is likely only to be a couple of GBs so won't require much storage
- Device will need speakers to hear game music and sound effects

Software:

- User can either run my game by downloading the unity file and running it on unity itself or download and run the executable via steam
- Other than that, there should be no specific software requirements

Computational Methods

My idea is suitable to be solved by a computer as:

- There will be a decomposition, in which various parts of the game will be made into smaller, simpler problems so that they are easier to manage e.g., main game broken down into player, enemies map etc... And a player can be broken down into player health, movement, items, HUD, etc... Moreover, I will make use of object-oriented programming in order to remove unnecessary repetition of code e.g., creating a scriptable object for melee weapons so that you can adjust the attribute values to suit each weapon.

- I will also make use of abstraction and visualization to help convey information to the player more simply through acts such as representing different components with icons, removing unnecessary details from the screen or interacting with the system with tools like sliders and drop-down boxes. One such example would be the player HUD where Health is indicated through a bar and how full it is and abilities are represented through icons. This makes this a suitable idea to be solved by a computer as without out a virtual display this information would lack clarity and be far harder to convey.

- In addition, I will make use of heuristics to create an optimal pathfinding algorithm for the enemies so they can track the player without too much processing power. This is because the player will be constantly moving and thus the game will need to constantly track the fastest route to the player for all the enemies. if inefficient, this can lead to enemies being motionless or slow in changing in their route alongside making the game far more demanding to run.
- Finally, there will be many operations running concurrently. This will be things such as event listeners, pathfinding and combat algorithms for enemies, rendering approaching areas of the map and un-rendering previously loaded areas etc... Thus, a computer is needed as the immense number of operations you would need to monitor/calculate would be too much for a human.

Design Overview

GUI Designs

Main Menu Screen:

Text: I have gone with a bold and capitalised font to make the text easy and clear to read. I have also made the buttons rather large to further improve the menu's visibility and make the buttons very easy to click. Furthermore, I have made the "START GAME" button bigger than the others as it is of more importance and is the first thing the user should be drawn to when opening the game. Additionally, I have chosen to make the text of the buttons white to contrast the background colour scheme to ensure they stand out. Lastly, when hovering over the button I plan to add some indicator most likely in the form of the text being highlighted or changing colour.

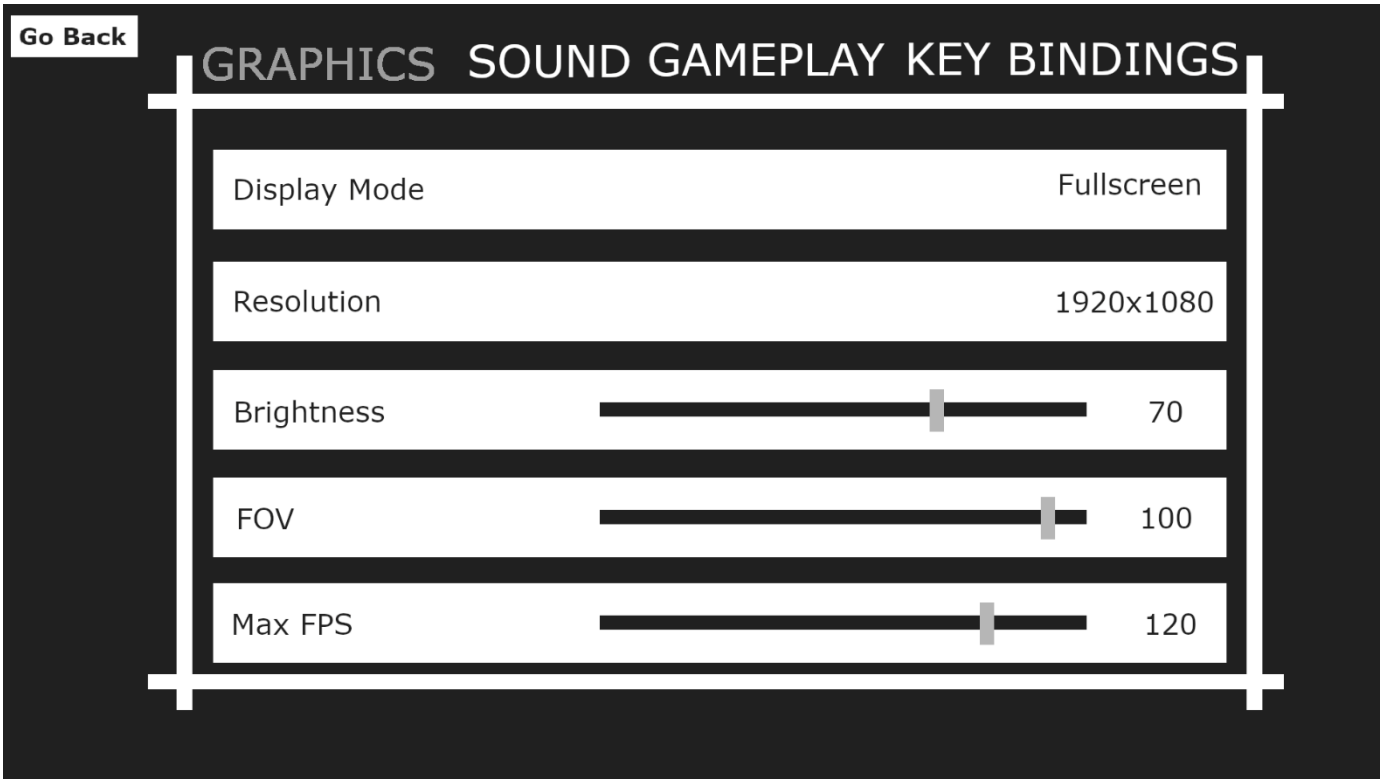
Positioning: Most games will position the buttons in either the centre or of left-hand side of the screen when they have something they want to present. For this reason, I have decided to go with the centre of the screen as there is nothing I want to showcase on the right-hand side of the screen. Moving on, for the vertical alignment of the buttons I have followed the typical order where the start game button is at the top and the exit game is at the bottom with the rest of the buttons in the middle. Furthermore, I have put the settings button above the loadout button as I feel it is of more importance as it can greatly affect the user's experience in-game.

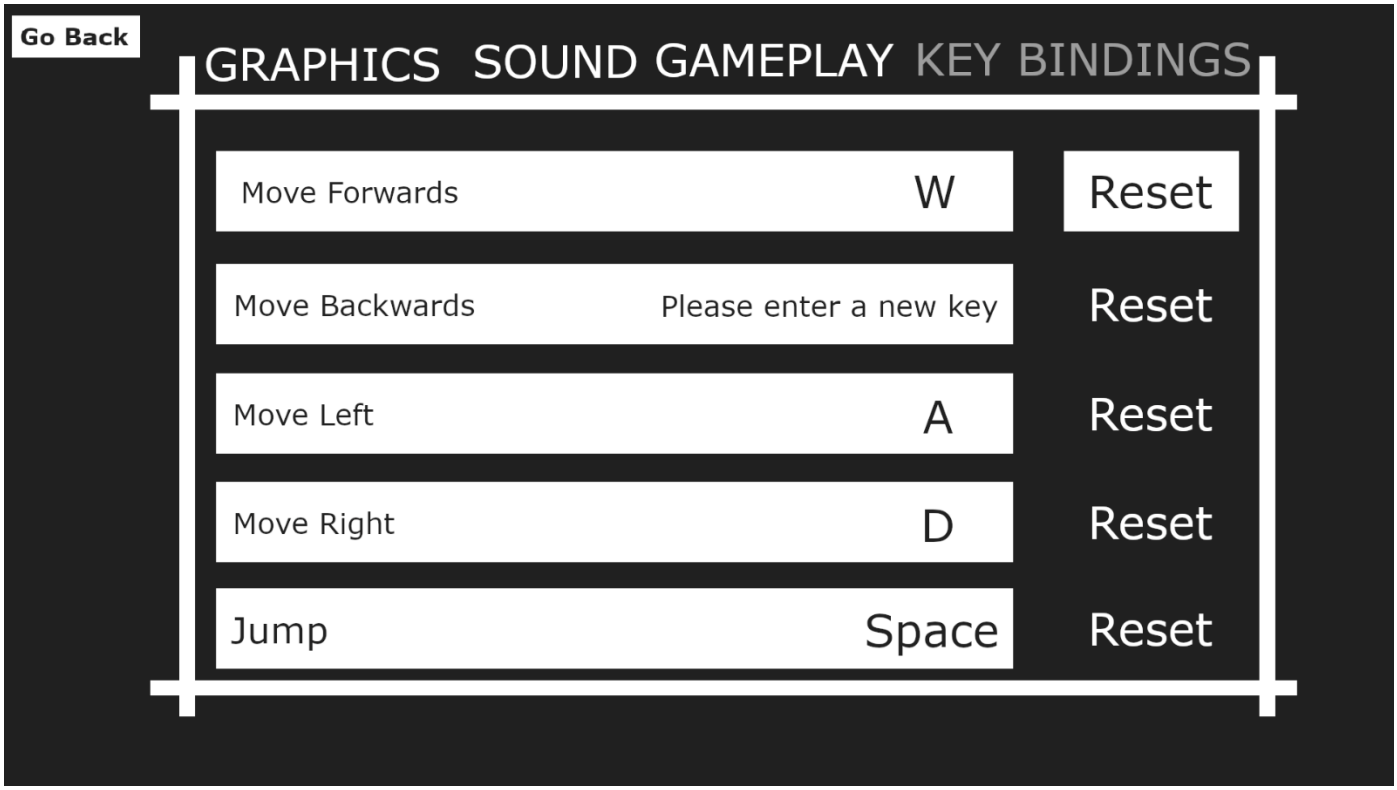
Background: For the background I have added a cinematic screenshot of the game as I have found this to be a common feature among many games and is good way to represent the game the user is about to play.

Additional Notes:

- In this example I have used a screenshot from another game as my background that has somewhat similar theme to my own game.
- As specified in the analysis section the start game button leads to the user to a new screen to select the game difficulty and (possibly) map. Furthermore, the settings button the settings menu, the loadout button the loadout menu and the exit game button will close the game and send the user to their desktop

Settings Menu:





Headings: In the settings menu I have added a heading for each category highlighting the selected category by changing the text from white to grey and adding a "glow" behind the category the user's mouse is hovering over. From the games I have seen it is typical for the category options to either be stacked vertically on the left-hand side of the screen or stacked horizontally at the centre screen, I decided to go with the centre of the screen. I find it more appropriate, for long category names, as otherwise, you would have to reduce the size and thus make the menu lose readability.

Options: For each option, I have encased it in a white rectangle so it will stand out on the screen alongside making sure they are not too small so each option can easily be selected. Furthermore, each setting will have either a drop-down box like the "Display Mode" and "Resolution" options with each available option displayed or a slider in which the user can drag the small grey rectangle along the black bar as shown in the "Brightness", "FOV" and "Max FPS" options. The only exception to this, will be, in the key bindings category, in which the user can double click the key bind they want to change and will be prompted as shown in the "Move Backwards" key bind and will be changed upon pressing a new key or esc to exit and set no key. On the right is the Reset button, which can be pressed, when the user has changed the default key bind and wants to restore it. The button press action is signified by the button background going from black to white and the text white to black (from the button shown in key bind 2,3,4 and 5 to the button in option 1).

Colour Scheme: I have made the settings screen black and white, with the darker/black tones representing the less important/background aspects and the parts highlighted in white are the more important parts. I have done this so that the screen is easy on the eyes and not too bright for the user andt easy for the user to navigate the menu and easily spot all key information on the screen.

Additional Notes:

- Each category will require the user to use their scroll wheel to view the additional options

- As specified the go back button will send the user to the Main Menu Screen when clicked and alternatively the user can use the esc key

Loadout Menu:

- 

- 

Loadout Options: For the loadout options, I have displayed each in a black box with a double black border and a Heading for its option, so that each option is clearly defined and to improve its overall appearance. Moreover, for each weapon option, I have displayed a white icon representing their currently selected weapon so that it's clear to the user that the weapons they have equipped. especially as it's easier to recognise it by a visual representation, as opposed to just its name. Furthermore, for the player colour option, I have displayed a circle of the player model's current colour, which is showcased on the right. A typical feature in many games, so. that players will have an idea of what they look

like in the game. Moving on, for the potion option, I have displayed an icon for the health potion on the right and an icon of the player's currently selected utility potion on the left. Each followed by a number in the colour of the potion representing the number of that potion the player will start with. When hovering over a number as shown in the 2nd image, it will display 2 arrows where upon clicking an up arrow, it will increment the number by 1 it and decrement the other number by 1 and the opposite when clicking the bottom arrow, the numbers will go as low as 0 and as high as 6. Upon clicking any of the weapon option icons or utility potion icons, the player will enter a selection menu as showcased in the 3rd image and a separate one for player colour, upon clicking the sphere as showcased in the 4th image. In addition, I plan to add a glow effectwhen hovering over any of the icons or the numbers in potion category, excluding the health potion icon to indicate to the player it can be clicked on.

Selection Menus: For my 1st type of selection menu, as showcased in the 3rd image, it will display the title of the option at the top with a number of weapons/potions the player can select below each with the name of the item and its icon below. The user can then navigate to additional options by clicking the arrows on the right and left of the screen. The player can exit the selection menu, by clicking anywhere on the background or clicking one of the items. Additionally, when hovering over one of the boxes it will add a glow effect around the box indicating the player can select it. For my 2nd type of selection menu as showcased in the 4th image it will again display the title of the option at the top with a box to pick the shade of the colour and a slider to change colour with a preview of the colour on the right. this is a very common form of picking colours ion many games, software, websites so the user should be accustomed to using it.

Font & Background: For the menu, I went for an oldish theme with an "old paper" background with an oldish font to match it. Additionally, for the selection menus, I used a blurred version of the loadout menu, so that it is clear the player is still in the same menu as well as put more focus on it, so that each element is easy to spot.

Additional Notes:

- Again, as specified the go back button will send the user to the Main Menu Screen when clicked and alternatively the user can use the esc key

Difficulty Selection Menu:

Difficulty options: Many games I have personally played or seen, have some form of visual representation for the difficulty, either by a change in colour or icon. In this case, I have decided to represent

the difficulty, by an increasing number of skulls with the name of that difficulty written below. Each difficulty will be selected by the user, clicking the name of the difficulty, which will ”glow” upon the player hovering it and when pressed, will turn the play button from grey to white (as shown above) - indicating that the player can now press the button. Furthermore, when the play button is in this state,it will also give off a glow effect, when hovered over, as opposed to nothing, when it cannot be pressed.

Font & Background: To match the theme of the skulls, I went with an old and worn theme and so I again used an “old paper image” for the background and oldish looking font that suited it.

Buttons: For the buttons, besides the difficulty options, I have used a black background with white text to contrast the rest of the screen whilst not using a drastically different set of colours improving their visibility all while still fitting the same theme and relatively easy to spot for the player. Additionally, I have made the “PLAY” button far larger than the “GO BACK” button and placed it in the bottom centre of the screen to draw as it is of far more importance. as for the position, it is typical to leave exit buttons in the top corners of the screen and for some sort of start button to be around the bottom of the screen.

Additional Notes:

- As specified the go back button will send the user to the Main Menu Screen when clicked and alternatively the user can use the esc key.
- In addition, the play button as specified will send the user to the loading screen until the game has loaded.

Loading Screen:

Background: For the background, I will use cinematic shots of the map the player is playing on. in this example, I have used a picture of a similar game, rotating between different facts and images at a set interval.

Facts: When the game is loading, I plan to add random facts about the game tailored to their selected difficulty and map. These will be brief, so that the player does not lose interest and they are easy to remember. Some players' loading times may be pretty short, so they won't have time to read it if it's too long.

Player HUD/in-game screen:

Abilities: For my dash ability, magic scrolls and flasks, I have represented each with a small icon that relates to its in-game appearance/characteristics so that, the user can identify each icon with ease. Moreover, each icon is fitted with the key to activate the power in the bottom right corner and additionally for the flasks the

quantity of the item in the top left corner. These are essential in reminding the user, what keys they need to press.whilst movement and weapon key bindings have standards across games, more niche abilities like using specific items will be harder for the user to remember. Furthermore, when any of the abilities are on cooldown, it will display the remainder of the cooldown to 1dp in white at the centre of the icon with a clock cooldown effect using a semi-transparent black overlay. This makes the user aware of when an ability is on cooldown and when it can be used; displaying the time left on the cooldown can be helpful to give the user an exact depiction of when the cooldown will end. In addition, I decided to display the cooldowns to 1dp as some cooldowns will be fairly short.

Healthbar & Player Icon: For the player's healthbar I have used a rectangular black bar filled with red bar representing the player's health split into 5 chunks each chunk representing the 100 health. This is a simple method to represent the player's health, used in almost all games as it is of vital importance, so players know when to be cautious/heal when on low health, etc.... Furthermore, I have used a red bar as it is the most common colour, I have seen used to represent healthbars and thus something the player will be used to. Additionally, I decided to put in a black rectangle as it makes the health bar stand out more. To the left of the player's healthbar, is a spherical sphere which whilst currently blank will display the face of the player's player model. This is a common feature I have seen in many games and so, I thought, it would make sense to include it.

Weapon: For the player's weapons I have displayed a white silhouette of their current weapon encased in a black box with a white outline. I decided to do this as it contrasts with the rest of the screen, making it easier for the user to spot as opposed to, if it wasn't in a box or monochrome. Below the weapon's silhouette, I have displayed a mini icon of its ammo type, to make clear to the player the corresponding ammo type of the weapon.This is followed by the current amount of ammo in its gun which turns from white to red when the ammo in the gun is low to alert the player to reload. Then the ammo the player is carrying, in grey and slightly smaller to distinguish it from the ammo in the weapon and as it is of less importance. These are both vital, so the player knows when they need to reload, be conservative with their shots when they're low on ammo, etc.... Finally, I have decided to place 2 small rectangles at the bottom of the box, one grey and one white, the 1st rectangle is the player's ranged weapon, and the 2nd is the player's melee weapon the white rectangle indicates the weapon the player is currently using and thus the grey the unequipped weapon. I decided to add this as it is a common feature, I have seen in many FPS games.

Wave Number: For the wave number of the game, I displayed it in a "bloody" font to fit the theme of the game. After each wave ends, I will add an animation that transitions to the next number.

Positioning: For the position of the HUD, I decided to go with the typical layout putting the weapon display and ammo stats on the right and healthbar on the left as it made more sense with the size of my displays and the player icon on the left. Furthermore, I decided to place the abilities around the weapon display as it seemed to fit better. Finally, for the wave number I displayed in the top left, as it didn't really fit in with the other displays and it is fairly common for games to display some stats in the top left of the screen. Positioning everything in a typical layout will make it easier for the player to navigate as it will be something they're used to.

Pause Menu:

Background: For the background of the pause menu, I have decided to use a blurred image of the player's in-game screen. This is to make the transition between the two more subtle and give the player a brief overview of their current situation after potentially returning from a break. Moreover, I have blurred to ensure that the pause menu option stand out and are the primary focus of the screen.

Text: Like the main menu I decided to go with a simple bold and capitalized font to make the text easy and clear to read. Moreover, I decided to use different colours for the buttons and "GAME PAUSED" text to distinguish that it's not a button. Additionally, I have used black and white to text to stand out against the background. Finally, I plan to like the main menu add a button indicator by changing the colour to grey when hovering over or adding a "glow" effect.

Positioning: I decided to put the "GAME PAUSED" at the top of the screen since it is of most importance and makes the user aware of the fact that the game is currently paused. And similarly, to the buttons in the main menu I have the button to the play the game at the top and the exit button at the bottom with the other buttons in the middle. However, in this case I have not included the loadout button since the player should not have the ability to change their loadout once beginning again thus making the option unnecessary and I have also added 2 exit buttons so the user can exit the whole application immediately rather than going through the main menu to exit the game or alternatively go to the main menu if they want to restart their session, change their loadout, etc...

Stats Screen:

Font & Background: I have yet again gone with this oldish theme to have some consistency between the screens. To achieve this, I again went with the ”old paper” background and oldish font to match it. Additionally, I have a preview of the player’s model on the right as it is a common feature you see in fps games with the end of game stat screens, and it fit the screens pretty well.

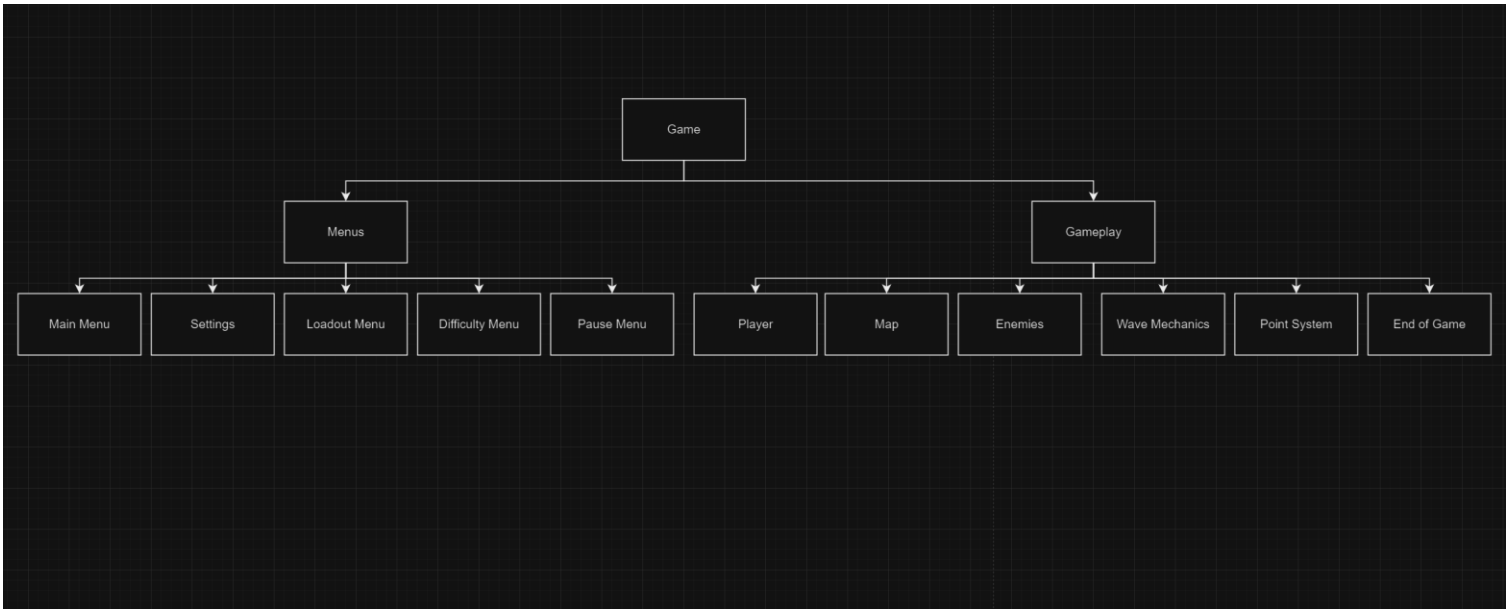
Text: For the text, I have made the heading and the subheading black to contrast the background and make it clearly visible to the player. However, for the values, I used red to distinguish it from the subheadings whilst still standing out among the background. Additionally, I have made the “GAME OVER” text far larger as it’s the heading and is,thus, of more importance.

Additional Notes:

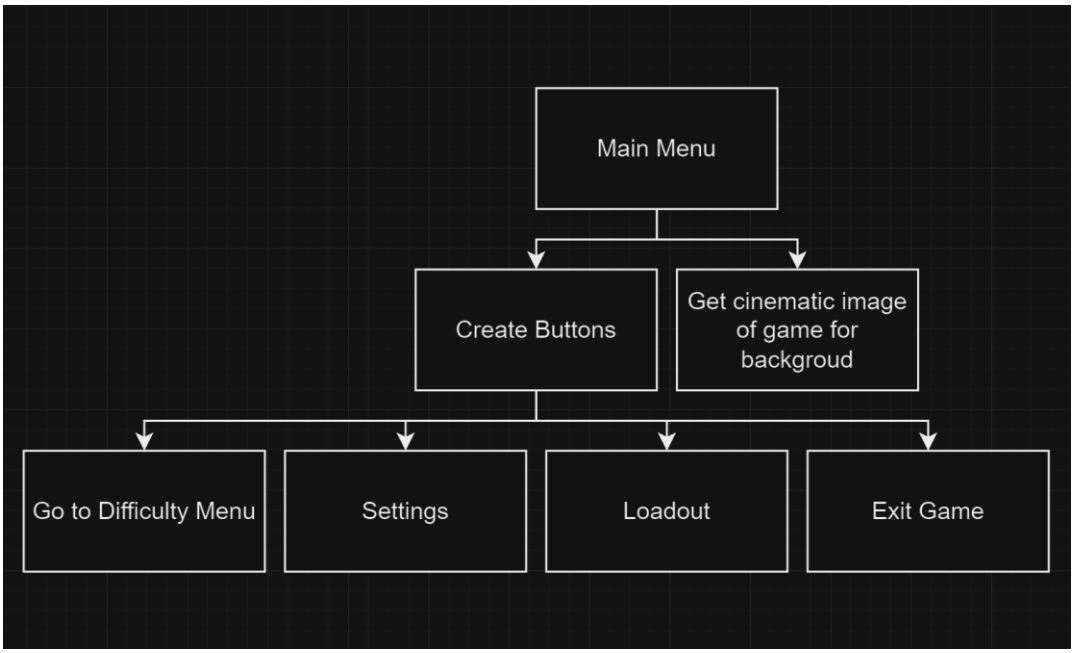
- As specified the return to Main Menu will send the user to the Main Menu Screen when clicked and alternatively the user can use the esc key.
- Plan to add “glow” effect around button when user hovers over it

Structure Diagram

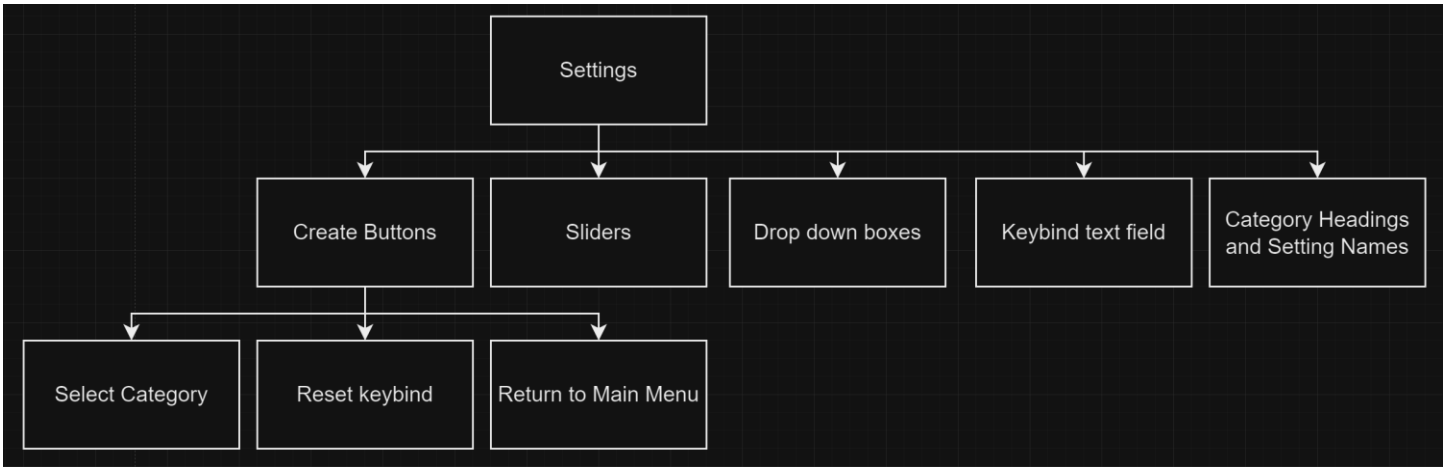
General:



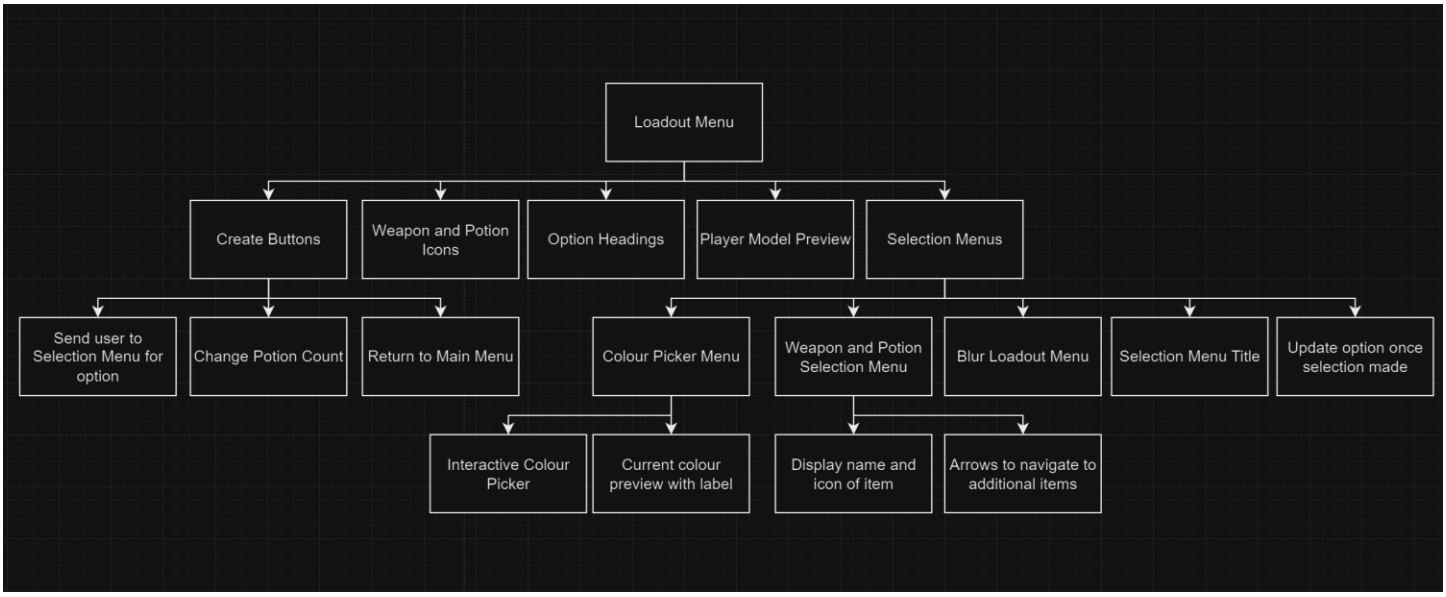
Main Menu:



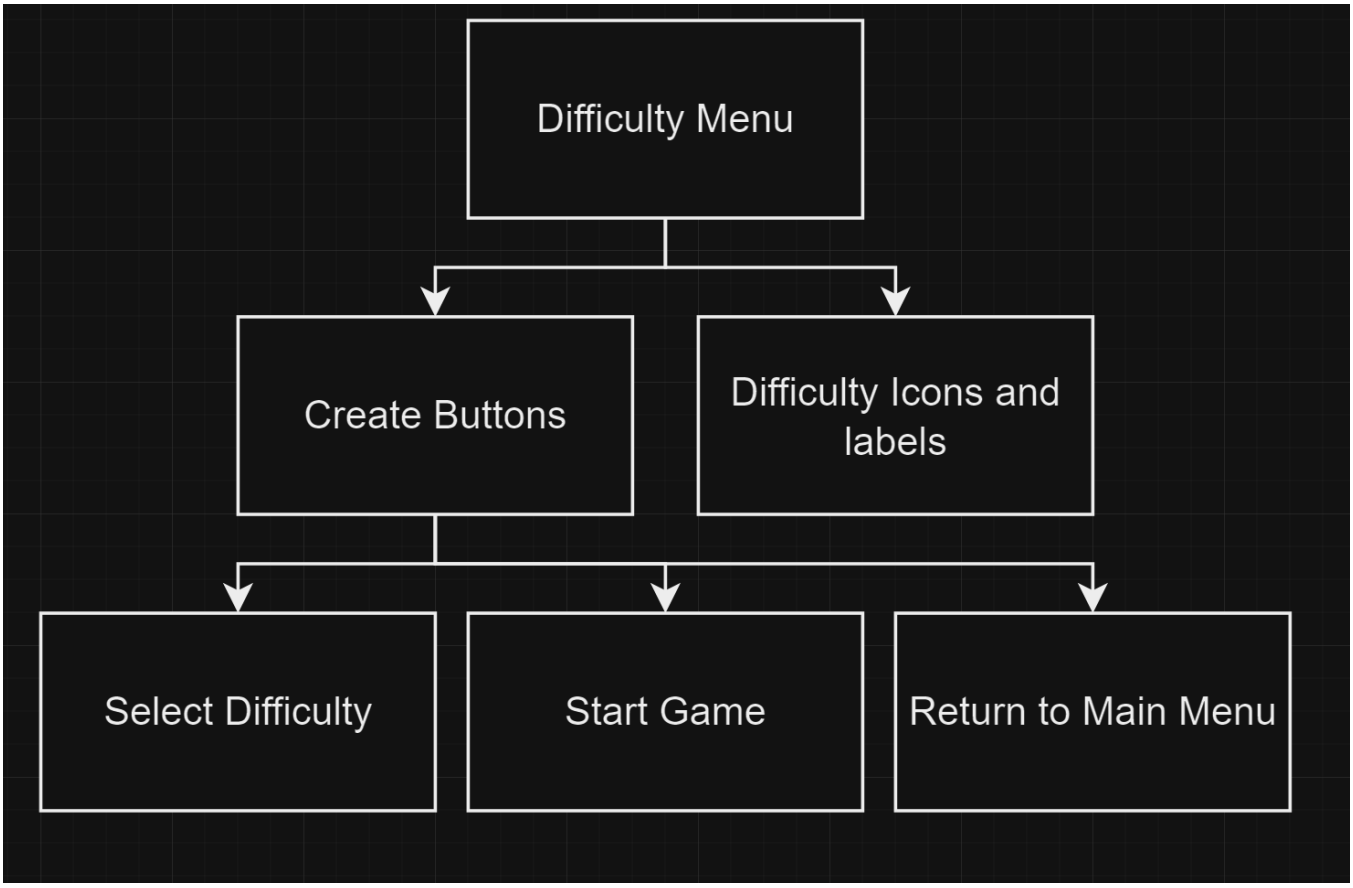
Settings:



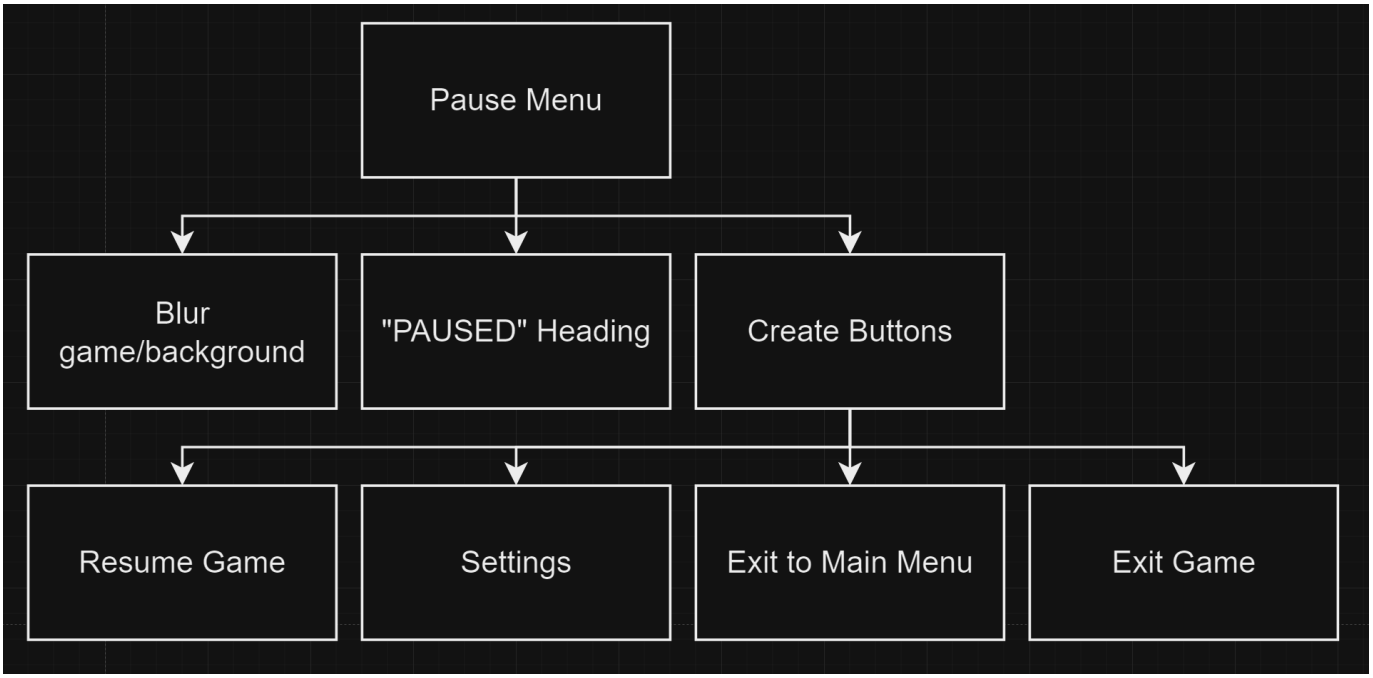
Loadout Menu:



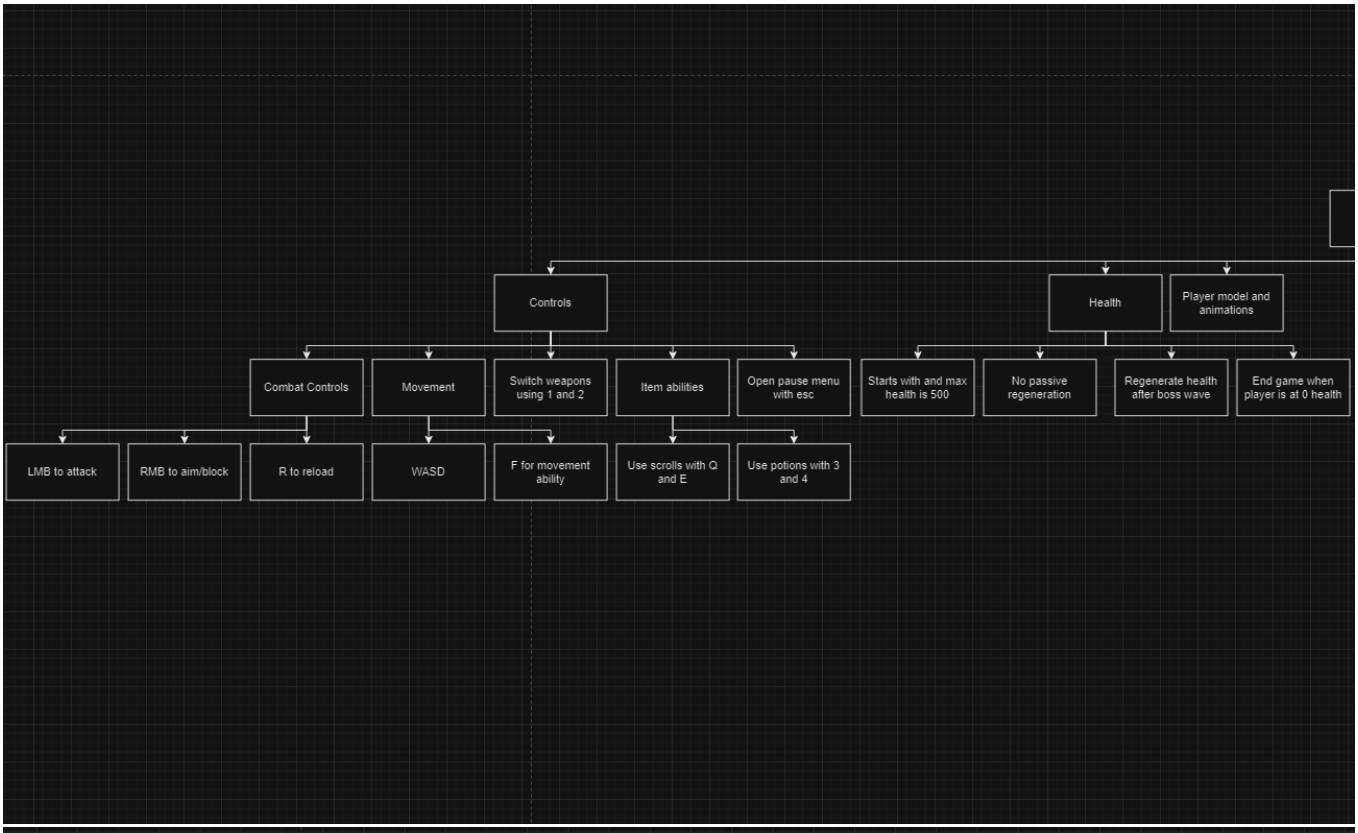
Difficulty Menu:

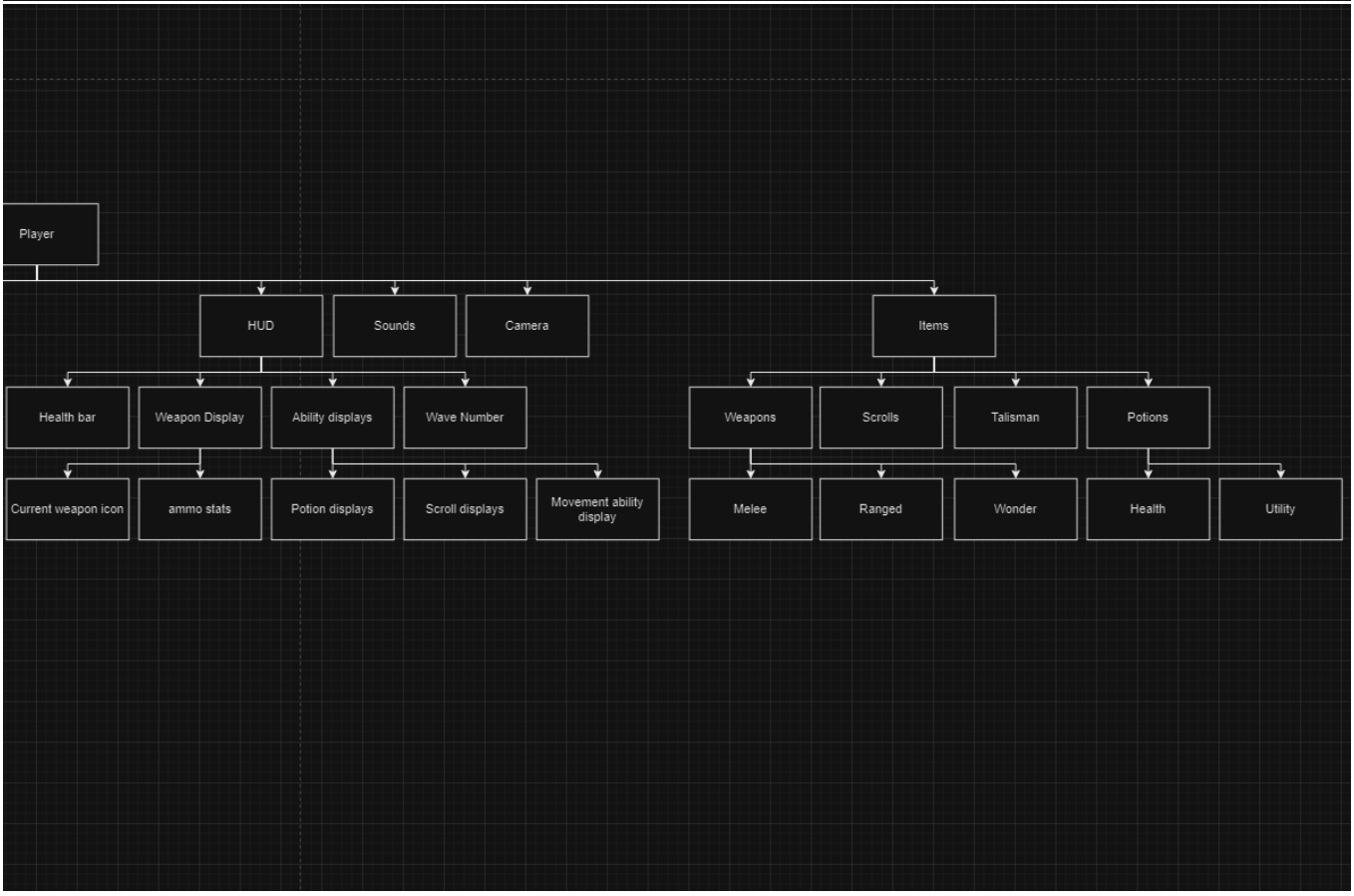


Pause Menu:

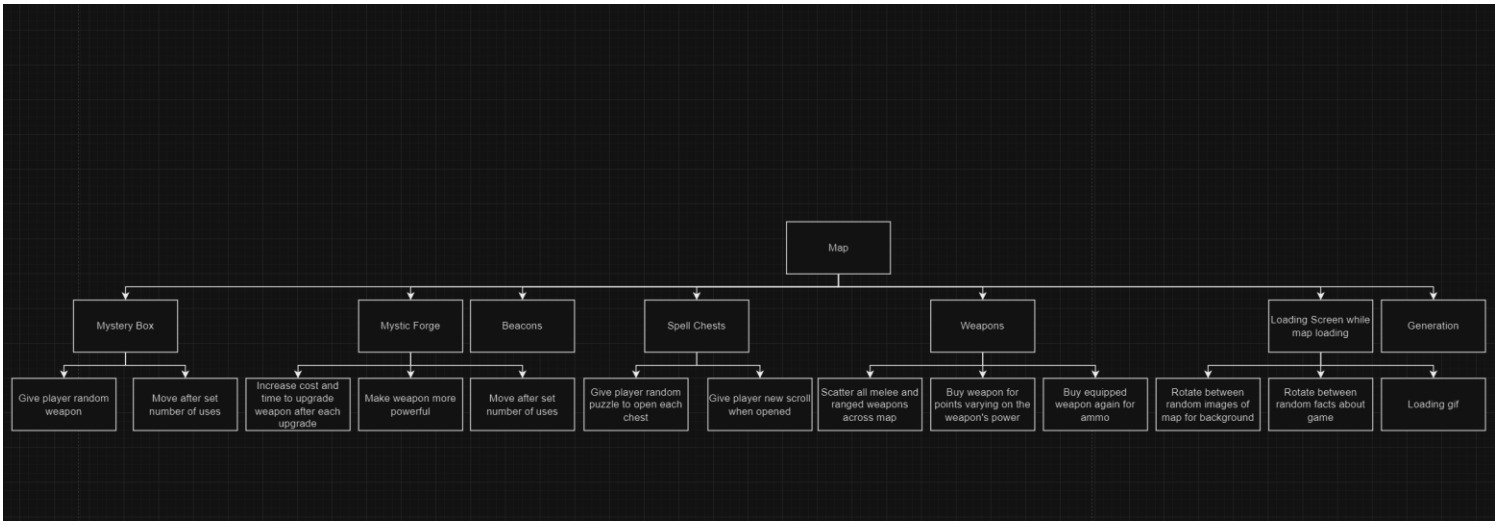


Player:

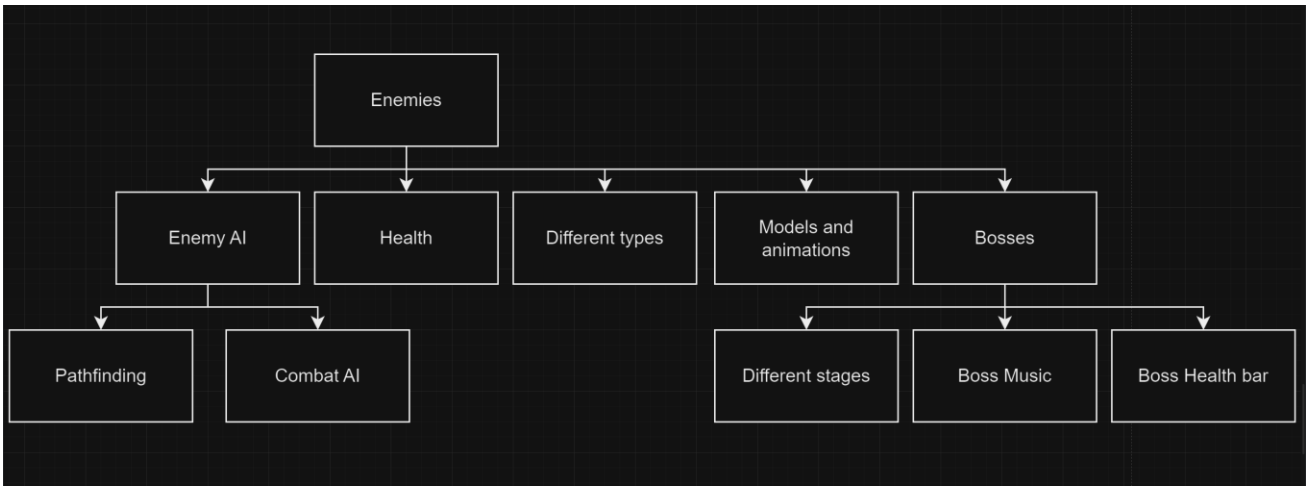

Controls
Health
Player model and animations
Combat Controls
Movement
Switch weapons using 1 and 2
Item abilities
Open pause menu with esc
Starts with and max health is 500
No passive regeneration
Regenerate health after boss wave
End game when player is at 0 health
LMB to attack
RMB to aim/block
R to reload
WASD
F for movement ability
Use scrolls with Q and E
Use potions with 3 and 4


Player
HUD
Sounds
Camera
Items
Health bar
Weapon Display
Ability displays
Wave Number
Weapons
Scrolls
Talisman
Potions
Current weapon icon
ammo stats
Potion displays
Scroll displays
Movement ability display
Melee
Ranged
Wonder
Health
Utility

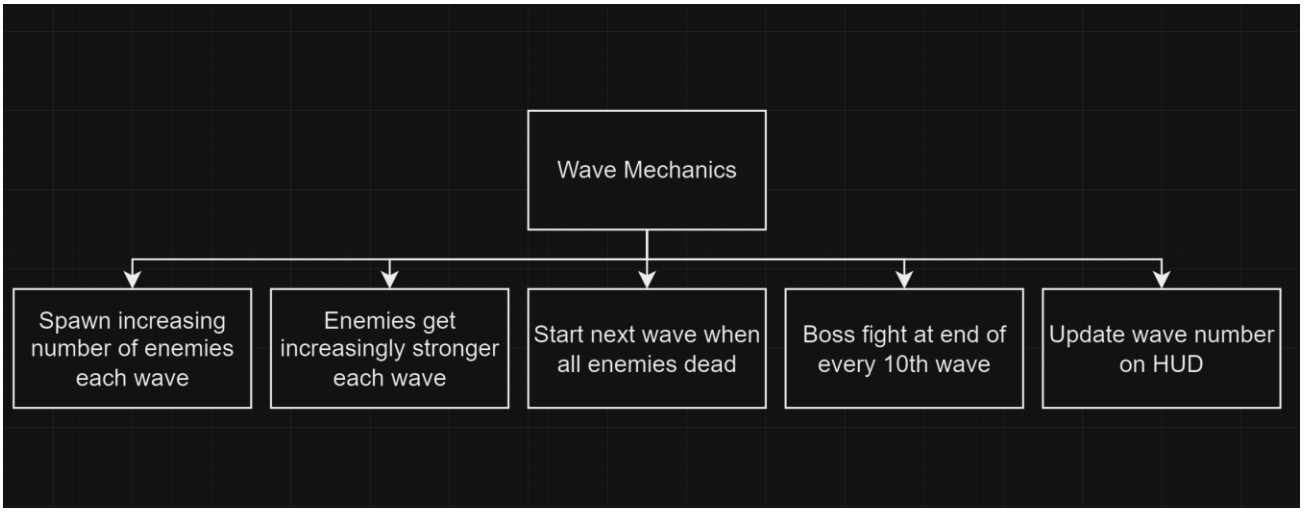
Map:



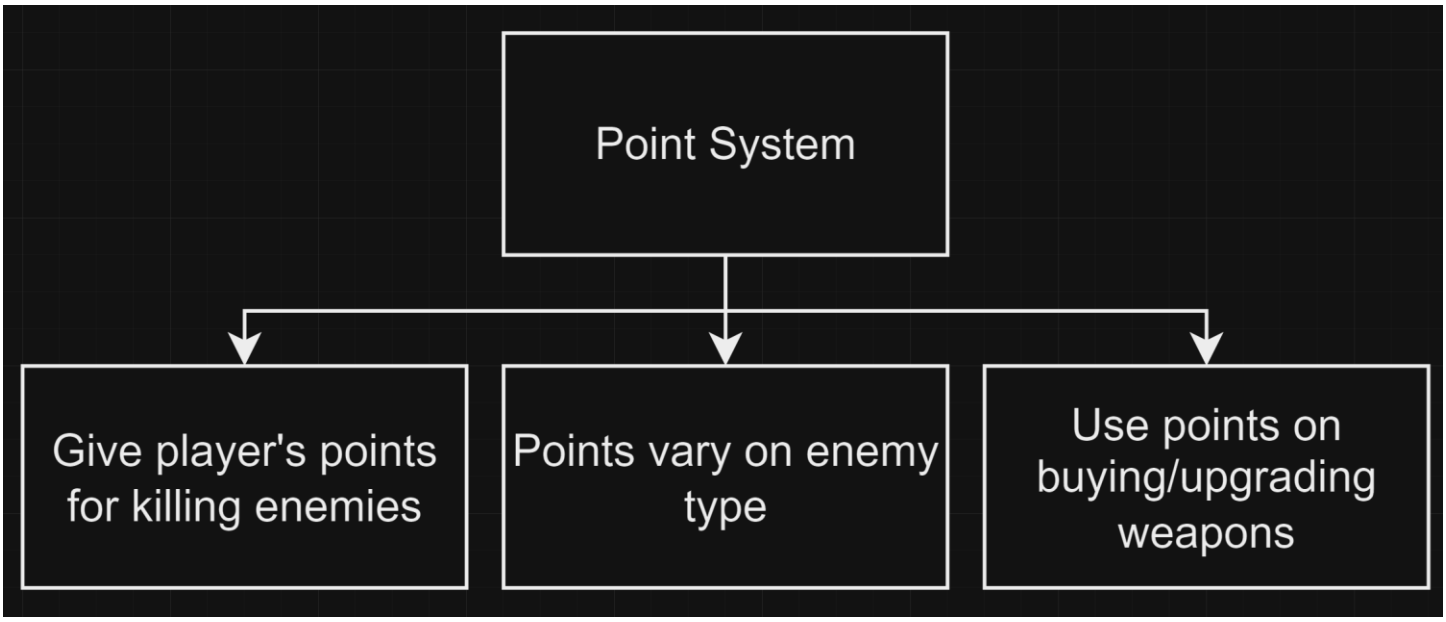
Enemies:



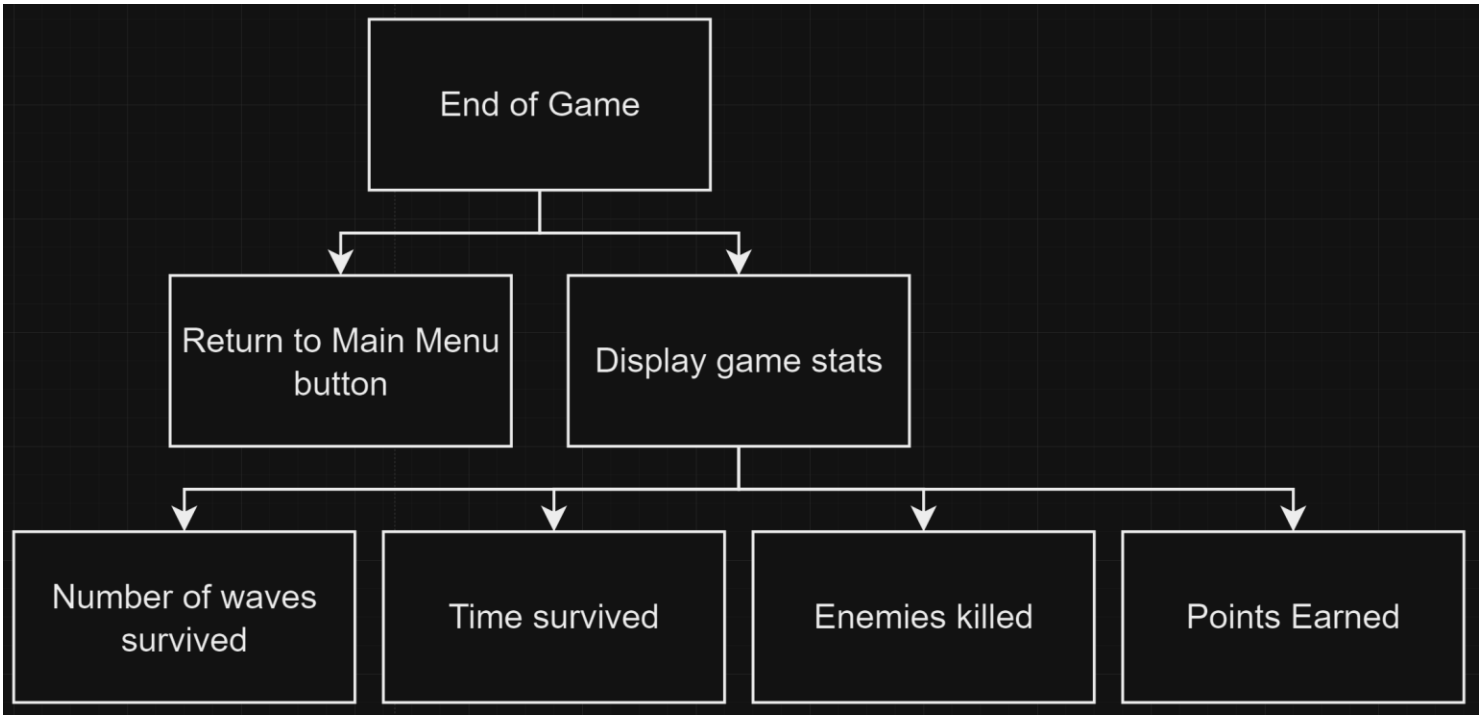
Wave Mechanics:



Point System:



End of Game:



Development Plan

Stage 1 (Main Player Mechanics - 2 weeks):

At the start of the development, I will need to add the player and give them the basic controls to move along with the animations. This is the main reason why I am implementing this first is because I believe it to be the most essential part of the game as without the player there is nothing you can do in the game.

Features to implement:

1. Create basic player model
2. Set up WASD movement
3. Create and set up player camera
4. Set up player sprinting
5. Create animations for idle, walk and sprint
6. Create player block collision
7. Set up player interaction and create a layer for interactable objects
8. Create player jump ability

Class Diagrams:

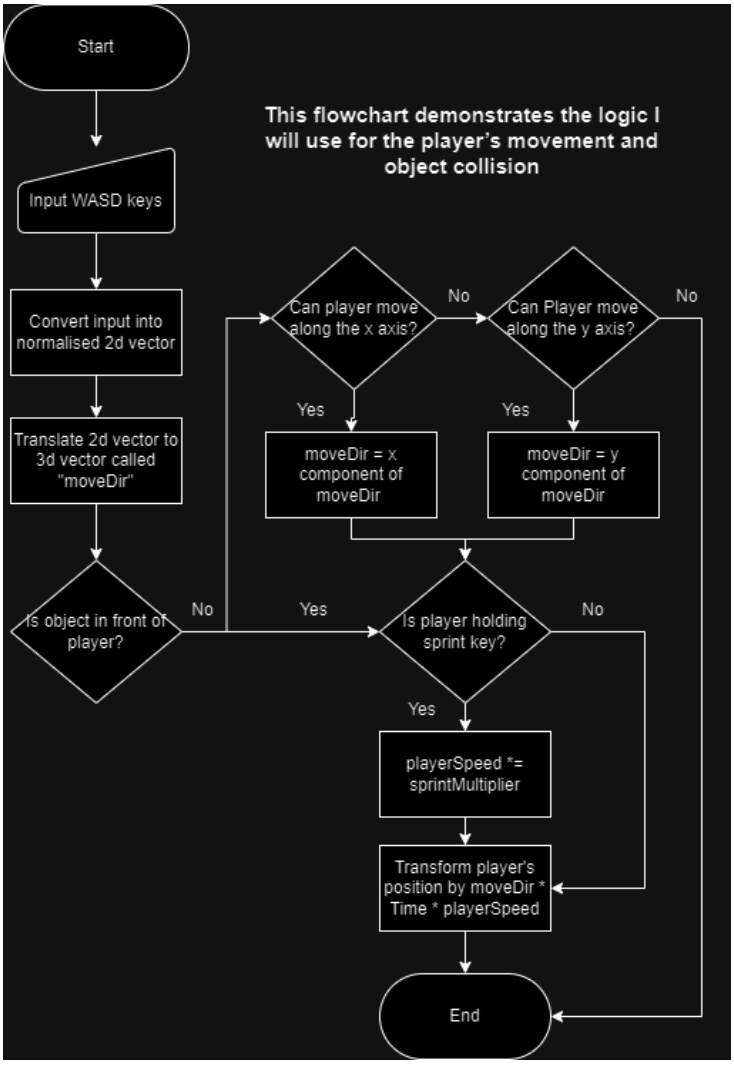
TBA

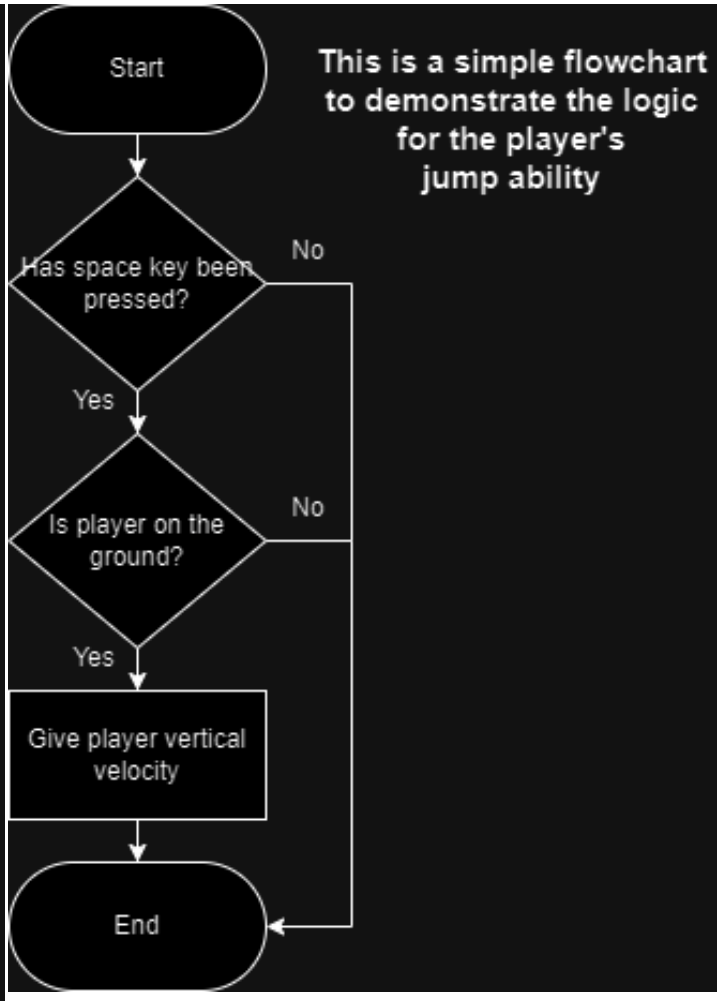
Data Dictionary:

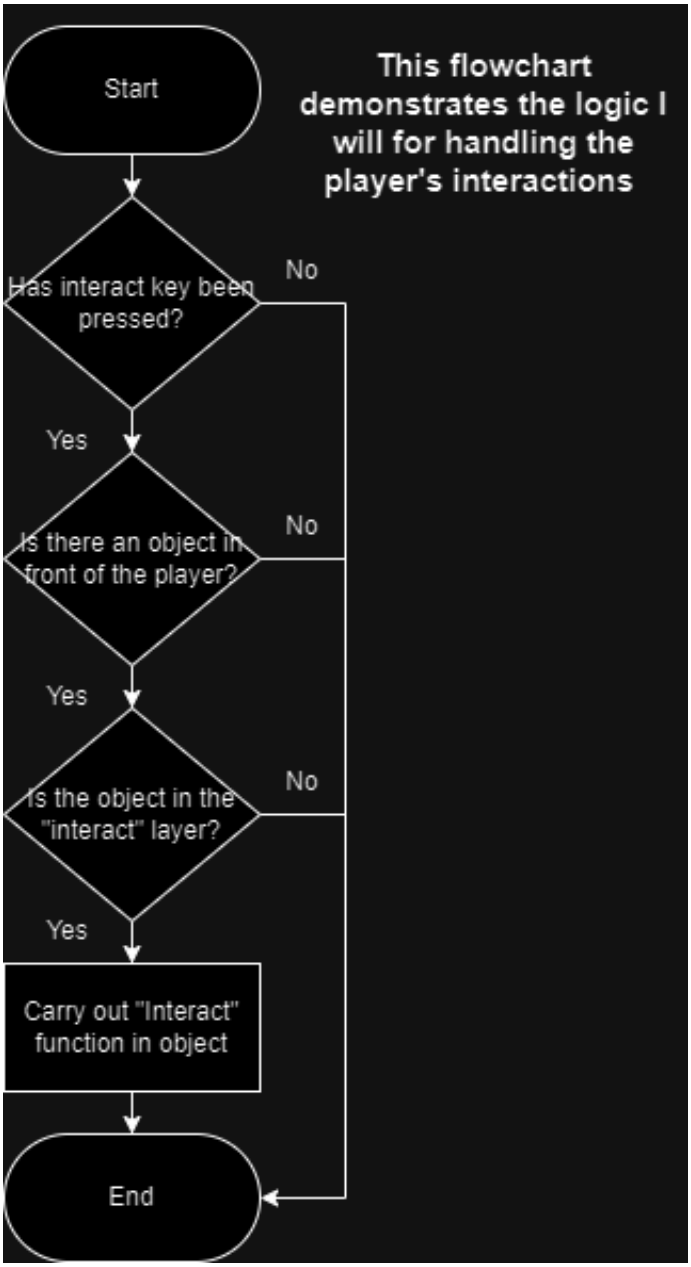
Name	Data Type	Description	Validation
movDir	Vector3	Direction player is moving in 3D space along the x and z axis	Vector is normalised
movSpeed	float	Speed at which player moves	Must be >0
sprintMultiplier	float	% increase in movSpeed	Must be >=1
isWalking	Boolean	Player is moving (movDir does not have a magnitude of 0)	Can be either true or false
isSprinting	Boolean	Player is pressing sprint key	Can be either true or false
isJumping	Boolean	Player activated jump ability	Can be either true or false
gravity	float	A constant used to represent the magnitude and direction of the gravitational force applied to the player	
jumpHeight	float	Height player will jump to when using the jump ability	Must be >0
v	Vector3	Player's vertical velocity	
isGrounded	Boolean	Bottom of player is touching object in the "ground" layer	Can be either true or false

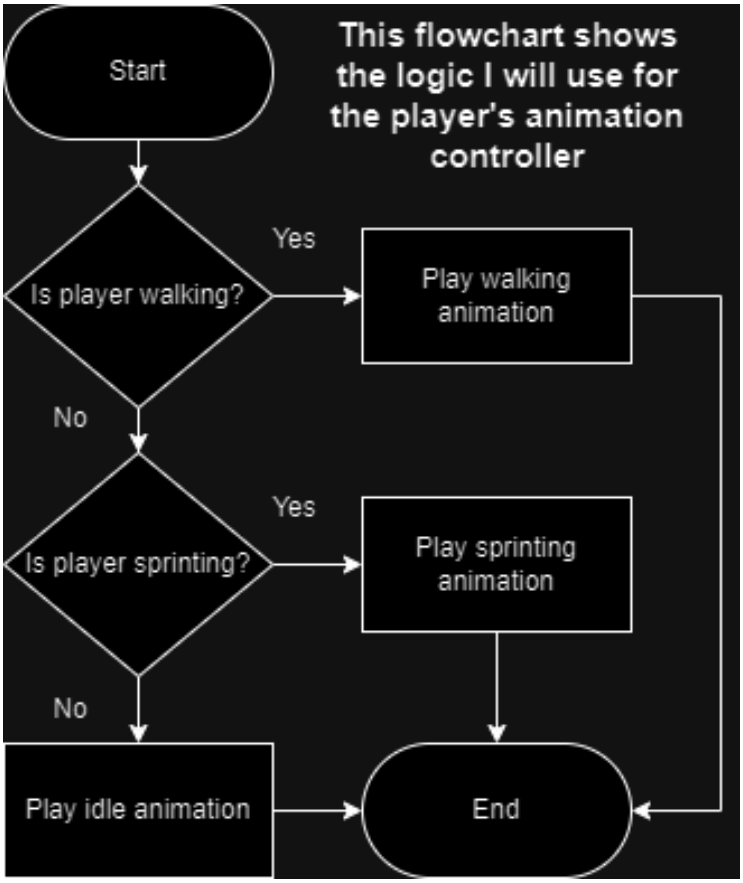
mouseSensitivity	float	Sensitivity of player's mouse	Must be >0
mouseX	float	Mouse movement across the screen horizontally	
mouseY	float	Mouse movement across the screen vertically	
xRotation	float	Camera's local rotation on the x axis	Must be between -90 and 90
rotateSpeed	float	Speed at which player rotates	Must be >0

Algorithmic Design:



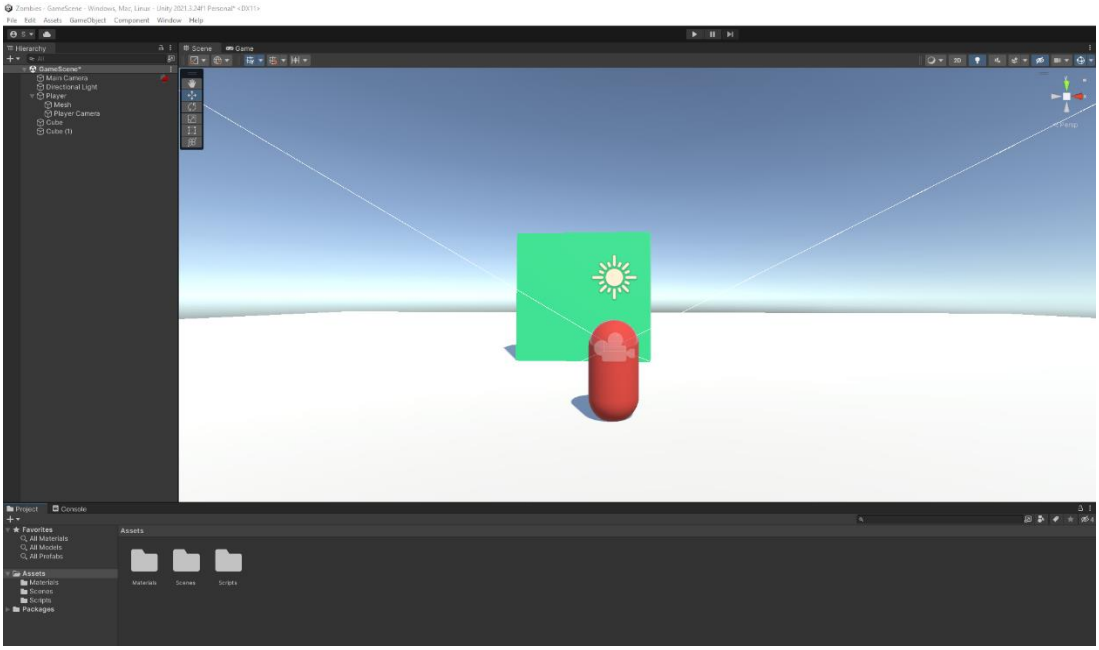



This flowchart demonstrates the logic I will for handling the player's interactions
Start
Has interact key been pressed?
No
Yes
Is there an object in front of the player?
No
Yes
Is the object in the "interact" layer?
No
Yes
Carry out "Interact" function in object
End


This flowchart shows the logic I will use for the player's animation controller
Start
Is player walking?
Yes
Play walking animation
No
Is player sprinting?
Yes
Play sprinting animation
No
Play idle animation
End

Development:

The first thing I did when creating my game, was to give the player a basic model and set up the environment.



I then moved on to the movement script for the player, first adding some code to move the player in the direction they're facing based on their input followed by adding a virtual camera for the player via Cinemachine. However, upon implementing rotation so the player turns to the direction they're moving, I ran into some errors where the screen would shake intensely when the player moves backwards and rotate in circles when moving backwards diagonally (demonstrated in "movement rotation bug.mkv"). Additionally, I was having trouble moving the player based on the direction the camera was facing.

```
5   public class Player : MonoBehaviour {
6
7       [SerializeField] private float movSpeed = 7f;
8       [SerializeField] private float rotateSpeed = 12f;
9
10      // Start is called before the first frame update
        Unity Message | 0 references
11      void Start()
12      {
13
14      }
15
16      // Update is called once per frame
        Unity Message | 0 references
17      void Update()
18      {
19          HandleMovement();
20      }
        1 reference
21      private void HandleMovement() {
22          //creates a movement vector from player's input
23          Vector2 inputVector = Vector2.zero;
24          if (Input.GetKey("w")) {
25              inputVector.y += 1;
26          }if (Input.GetKey("a")) {
27              inputVector.x -= 1;
28          }if(Input.GetKey("s")) {
29              inputVector.y -= 1;
30          }if(Input.GetKey("d")) {
31              inputVector.x += 1;
32          }
33          //translates input vector to 3D space
34          Vector3 movDir = inputVector.y*transform.forward+inputVector.x*transform.right;
35          //normalises vector so player maintains same speed diagonally
36          movDir.Normalize();
37
38          //updates player's pos and rotation based on direction they're facing
39          transform.position += movDir * Time.deltaTime * movSpeed;
40          transform.forward = Vector3.Slerp(transform.forward,movDir,Time.deltaTime*rotateSpeed);
41
42
43      }
44  }
45
```

After trying numerous methods to solve the errors, I decided to follow a tutorial and rewrite the code from scratch, this time instead rotating the player with the camera. I first began with writing a script to rotate the camera with the mouse adding in a lerp to smooth out the rotation to fix the stuttering I got when testing. I then began to work on the player's movement, using Unity's character controller making use of its "move" function to move the player provided the movement vector this time taking in the player's input via Unity's built-in input system.

```
public class MouseLook : MonoBehaviour {
    [SerializeField] private float mouseSensitivity = 100f;
    [SerializeField] private Transform player;
    [SerializeField] private float rotateSpeed = 10f;
     private float xRotation = 0f;
    Unity Message | 0 references
    void Start() {
        Cursor.lockState = CursorLockMode.Locked;
    }
    Unity Message | 0 references
    void Update() {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation,-90f,90f);

        transform.localRotation = Quaternion.Lerp(transform.localRotation,Quaternion.Euler(xRotation,0f,0f),rotateSpeed * Time.deltaTime);
        player.Rotate(Vector3.up * mouseX);
    }
}
```

```
public class Player : MonoBehaviour {

    [SerializeField] private CharacterController characterController;
    [SerializeField] private float movSpeed = 7f;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start() {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update() {
        HandleMovement();
    }
    1 reference
    private void HandleMovement() {
        //creates movement vector relative to the player's position based on their input
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");
        Vector3 movDir = transform.right * x + transform.forward * z;

        //provides charcter controller with movement vector for player
        characterController.Move(movDir.normalized*movSpeed*Time.deltaTime);
    }
}
```

Shortly after, I started on the player's sprint ability. this was relatively simple although I ran into some errors as I initially did not consider that the variable "sprintMultiplier" would maintain its value after each update. but it did not take long to resolve this by resetting its value every update.

```
private void HandleMovement() {
    //reseting sprint multiplier
    sprintMultiplier = 1f;
    //creates movement vector relative to the player's position based on their input
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 movDir = transform.right * x + transform.forward * z;
    //adds sprint multiplier if sprintkey pressed
    if(Input.GetKey(KeyCode.LeftShift)) {
        sprintMultiplier = 3f;
    }
    //provides charcter controller with movement vector for player
    characterController.Move(movDir.normalized*movSpeed*sprintMultiplier*Time.deltaTime);
}
```

After testing the player movement some more, I decided to implement the player's jump ability alongside gravity mechanics so the player can fall. For the falling, I applied a constant downwards vector onto the player whilst they were not touching an object in the "ground" layer which I would check by creating a sphere at the player's feet as well as resetting their vertical velocity once reaching the ground. I then worked on the player's jump ability by setting the vertical velocity to initial velocity required to reach the desired jump height upon using the jump ability by rearranging the equation $V^2 = U^2 + 2as$.

```
private void HandleMovement() {
    //creates sphere at bottom of player to check if they have reached the ground
    isGrounded = Physics.CheckSphere(groundCheck.position,groundDistance,groundLayer);
    //sets velocity to -2 when player reaches the ground
    if(isGrounded && v.y<0) {
        v.y = -2f;
    }
    //reseting sprint multiplier
    sprintMultiplier = 1f;
    //creates movement vector relative to the player's position based on their input
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 movDir = transform.right * x + transform.forward * z;
    //adds sprint multiplier if sprintkey pressed
    if(Input.GetKey(KeyCode.LeftShift)) {
        sprintMultiplier = 3f;
    }
    //provides charcter controller with movement vector for player
    characterController.Move(movDir.normalized*movSpeed*sprintMultiplier*Time.deltaTime);

    if(isGrounded && Input.GetButtonDown("Jump")) {
        v.y = Mathf.Sqrt(jumpStrength * -2f * gravityMag);
    }

    //adds gravity to player
    v.y += gravityMag * Time.deltaTime;
    characterController.Move(v * Time.deltaTime);
}
```

For handling the interactions, I used a raycast to check if there's an object within 2 units of the player in the "interact" layer and the direction they're facing only carrying it out when the player presses the "E" key. However, for now I have decided to add the "interaction" function when I develop the interactable objects.

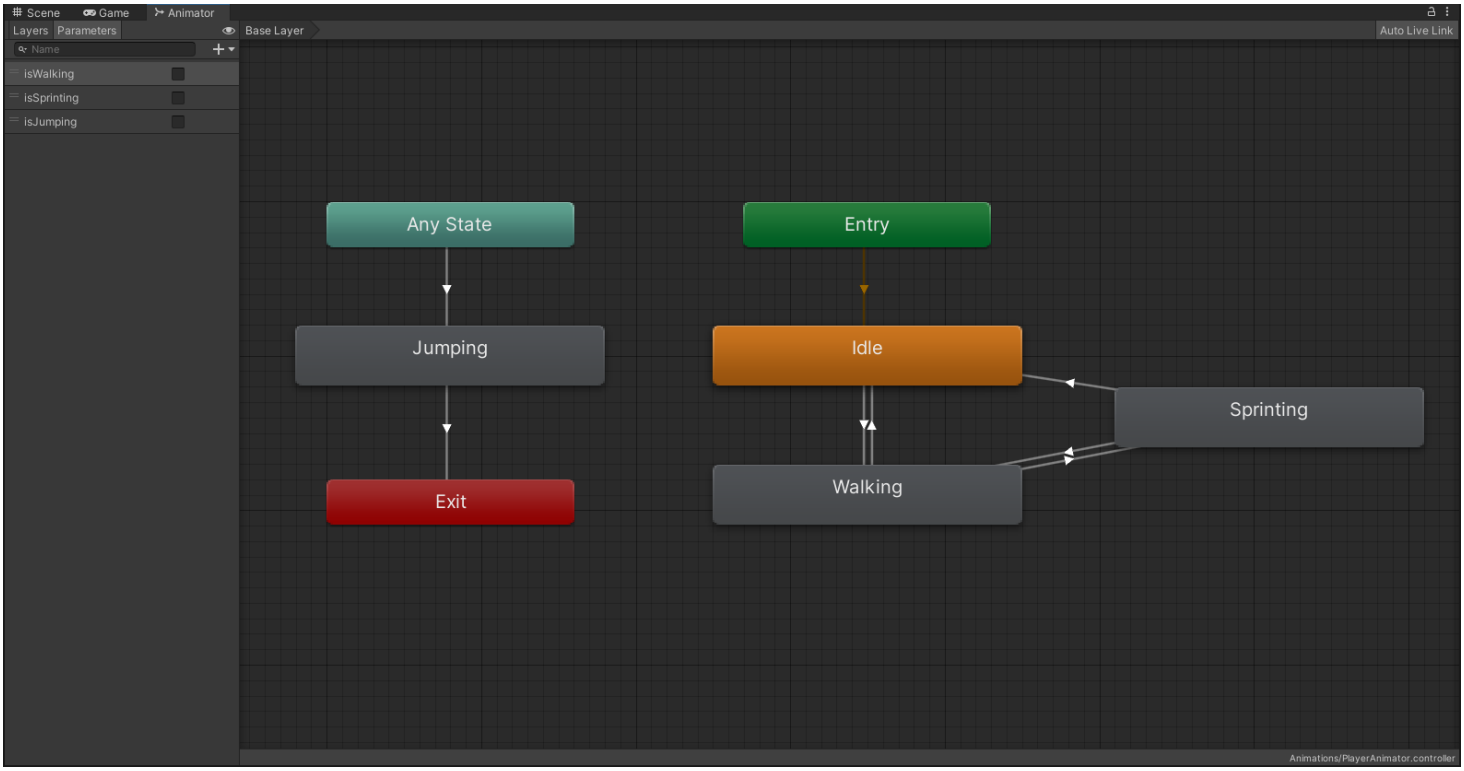
```
void Update() {
    HandleMovement();
    if(Input.GetKey(KeyCode.E)) {
        HandleInteractions();
    }
}
```

```
private void HandleInteractions() {
    //creating 3D vector for direction player is looking/moving
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 movDir = transform.right * x + transform.forward * z;

    //checking if object in the interact layer is within 2 units of the player in the direction they're looking
    if(Physics.Raycast(transform.position,movDir,out RaycastHit hitInfo,2f,interactLayer)) {
        Debug.Log("Interact!");
    }
}
```

Finally, I worked on the animations for the player. First importing a better player model from the unity asset store with plenty of animations, sounds and more. I then proceeded to adjust the groundCheck and camera to the new player

model. Next, I worked on the animator creating the different animated states: idle, walking, sprinting, jumping and falling along with transactions between them which I would control through Booleans. Although after later I deemed a falling animation unneeded and removed it.



After creating the animator, I added in the Booleans to my player script to track when each state should occur and separately some functions to return them so I could access them elsewhere.

```
//reseting all animation states
isWalking = false;
isSprinting = false;
isJumping = false;
```

```
//making isWalking true when player is moving
if(movDir != Vector3.zero) {
    isWalking = true;
}
//adds sprint multiplier and activates isSprinting if sprintkey pressed
if(Input.GetKey(KeyCode.LeftShift)) {
    sprintMultiplier = 3f;
    isSprinting = true;
}
```

```
if(!isGrounded&&v.y>0) {
    isJumping = true;
}
```

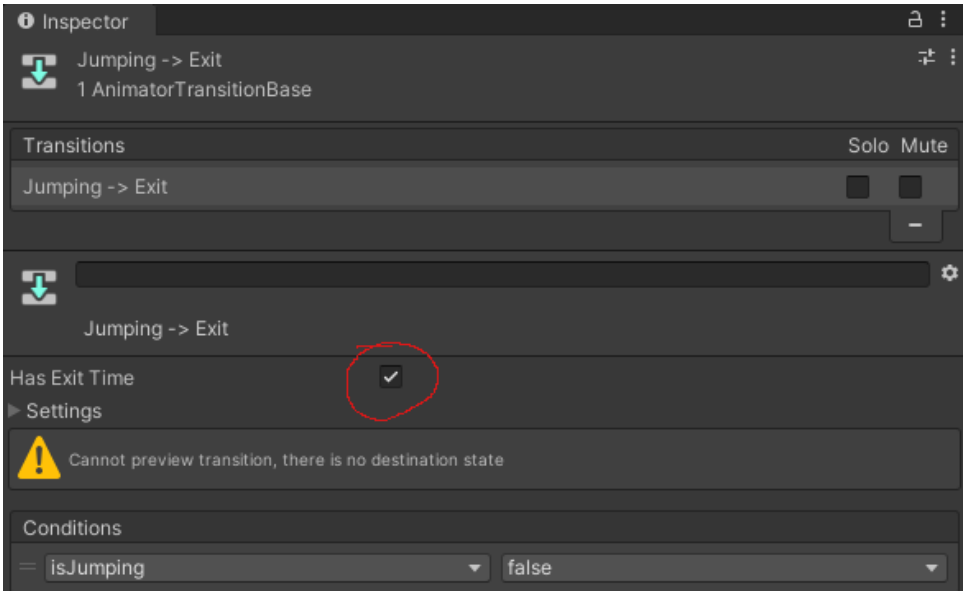
```
public bool IsWalking { get { return isWalking; } }
1 reference
public bool IsSprinting { get { return isSprinting; } }
1 reference
public bool IsJumping { get { return isJumping; } }
```

I then added a script to update the Booleans in the animator with their values on the player script.

```
private void Awake() {
    animator = GetComponent<Animator>();
}
Unity Message | 0 references
private void Update() {
    animator.SetBool("isWalking",player.IsWalking);
    animator.SetBool("isSprinting",player.IsSprinting);
    animator.SetBool("isJumping",player.IsJumping);
}
```

After some testing I realised the jump animation was looping when I jumped (demonstrated in “jump loop bug.mkv”). I then did some debugging and realised I should only make the jump ability active for the update it is activated and give the exiting transition exit time, so it finishes the animation before exiting the state.

```
//executes jump ability if player is on the ground
if(isGrounded && Input.GetButtonDown("Jump")) {
    //sets velocity to initial velocity required to reach desired jump height
    v.y = Mathf.Sqrt(jumpStrength * -2f * gravityMag);
    isJumping = true;
}
```



Testing Table:

Key: Green – Successful, Red- unsuccessful, Orange - Change

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Player moves correctly with WASD keys	Use W, A, S and D keys	Player moves forward, left, backwards and right	Player successfully moved forwards, backwards, left and right relative to the direction they’re facing
1b	Vector is normalised when moving diagonally	Move diagonally	Vector is normalised and player will move at the same speed	Player maintains the same speed when moving diagonally

1c	Player rotates to direction they're moving	Use WASD keys	Player and camera will rotate to direction they're moving	When moving backwards screen "shakes" intensely and moving backwards diagonally would make the player move in circles
1d	Player moves based on direction camera is facing	Make camera face new direction and move forwards	Player will move forwards in the new direction the camera is facing	Player moves relative to the direction the player is facing. However now instead by rotating the player with the camera.
2a	Camera follows player and remains in FPP	Move with WASD keys	Camera will move with the player and will stay in a FPP	Camera follows the player maintaining a FPP
2b	Player can rotate head with mouse	Move mouse	Player's head will rotate and the camera along with it	Due to the player now rotating with the camera and change in model this will no longer be implemented
3	Player's speed increases whilst holding sprint key	Move and hold sprint key	Player's speed will increase	Player's speed increased exponentially
4a	Idle animation triggers when player is not doing anything	Run game and do nothing	Idle animation will trigger briefly after player is doing nothing	When player is unmoving the idle animation will play
4b	Player transitions to walk animation when using WASD	Use WASD keys	Walk animation will trigger when player begins moving	When the player moves, and the sprint key is not being pressed the walk animation will play
4c	Player transitions to sprint animation when using WASD and sprint key	Use WASD keys and sprint key	Sprint animation will trigger when player is sprinting	When the player moves and is holding the sprint key the sprint animation will play
4d	Player transitions to jump animation when activating jump ability	Press spacebar when on the ground	Jump animation will play once after player presses jump key	Jumping animation loops as player is moving upwards
5a	Player is stopped when trying to move through a visible object	Move into a visible object	Player will stop moving	Player is unable to pass through visible objects
5b	Player can "wall hug"	Move into wall diagonally	Player will move alongside the wall at a reduced speed	Player will move at a reduced speed when moving into a wall diagonally
6	Player can interact with objects in the correct layer using the interact key	Use interact key when in front of an object in the interact layer	"Interact!" will be printed in the console when using the interact key in front of an object in the interact layer	Upon pressing the interact key (E) when an object is when 2 units of the player in the direction they're facing "Interact!" is printed in the console

7a	Player is launched up and returns to ground when spacebar is pressed	Press spacebar	Player is launched directly up and then returns to the ground	Upon pressing the spacebar, an upwards velocity is applied to the player to send them to the desired jump height and then fall until they reach the ground
7b	Player keeps their momentum and direction when jumping whilst walking or sprinting	Press spacebar whilst walking and or sprinting	Player continues in same direction and keeps momentum whilst jumping	Player's direction across the x and z axis is maintained and momentum is unaffected whilst jumping
7c	Player cannot jump in the air	Try to jump when player is off the ground	Player is unable to jump when not on the ground	Player is unable to use jump ability in the air

Stage 1 – Mini Evaluation:

So far in my 1st stage development, I feel I have done a decent job as I have accomplished all my goals for this stage. Successfully providing the player with a basic player model, which I later replaced with a more "humanoid" one as well as WASD movement, sprinting and jumping for the player alongside animations for each state. Additionally, I set up the player camera and basic player interaction. However, I feel as though I was unprepared in implementing some features and severely underestimated the workload for the stage as a whole.

Stage 2 (Player Weapons – 2 weeks):

Next, I plan to implement the player's weapons as they are crucial part of the game and thus are vital for testing features in future stages.

Features to implement:

Part 1:

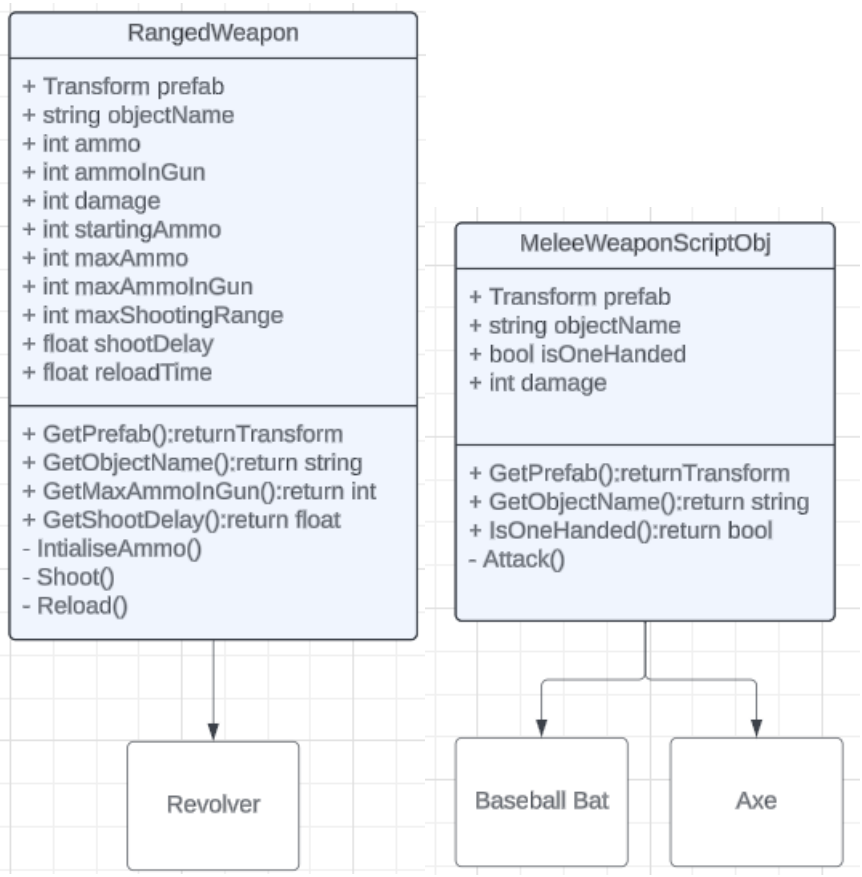
1. Create new scriptable objects for player melee and ranged weapons
2. Create/find visual for player weapons
3. Set up controls for picking up, dropping and switching weapons alongside visual when player is holding them
4. Create shoot ability and animations for ranged weapon
5. Create ammo and reload mechanics
6. Create crosshair
7. Create attack ability and animation for melee weapon and unarmed

Part 2:

8. Create throw ability and animation for melee weapon
9. Create aim ability and animation for ranged weapon

10. Create weapon recoil
11. Create animations for player reloading
12. Create special movement animations for player with weapon equipped

Class Diagrams:



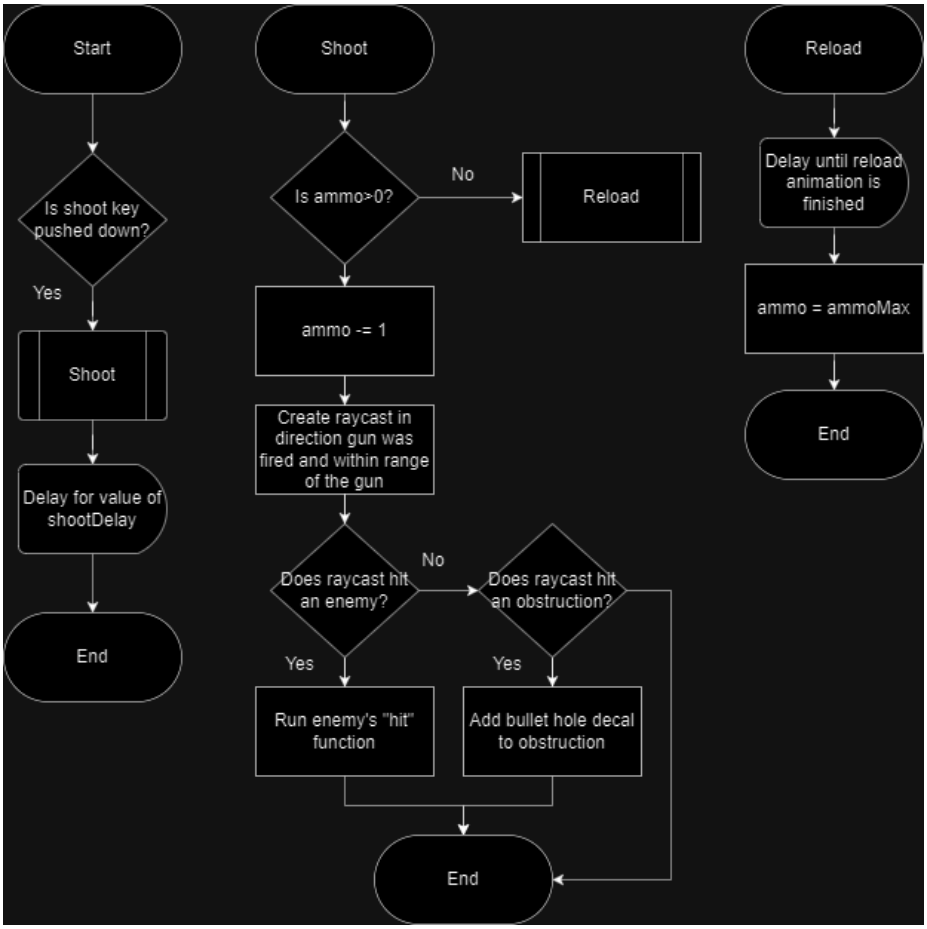
Data Dictionary:

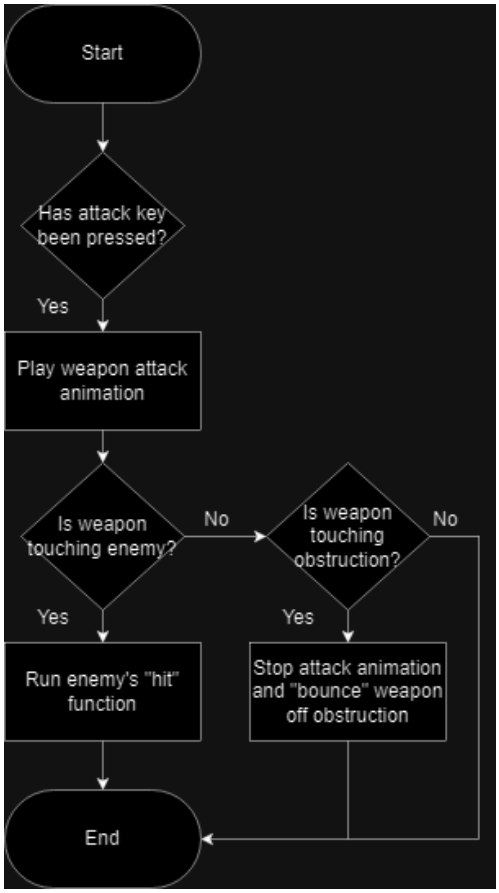
Name	Data Type	Description	Validation
weaponRef	Transform	The object hit by the interact raycast	
weaponInHand	bool	Bool used to track if the player has a weapon in their hand or not (later replaced by rightHandWeapon, leftHandWeapon and rangedWeapon)	Can be either true or false
isAttacking	bool	Player pressed attack key with a weapon in hand	Can be either true or false
isMelee	bool	Player is holding a melee weapon (is later replaced by rightHandWeapon and leftHandWeapon)	Can be either true or false
rightHandWeapon	bool	Player has a melee weapon in their right hand	Can be either true or false
leftHandWeapon	bool	Player has a melee weapon in their left hand	Can be either true or false
playerRightHand	Transform	The player's right hand	

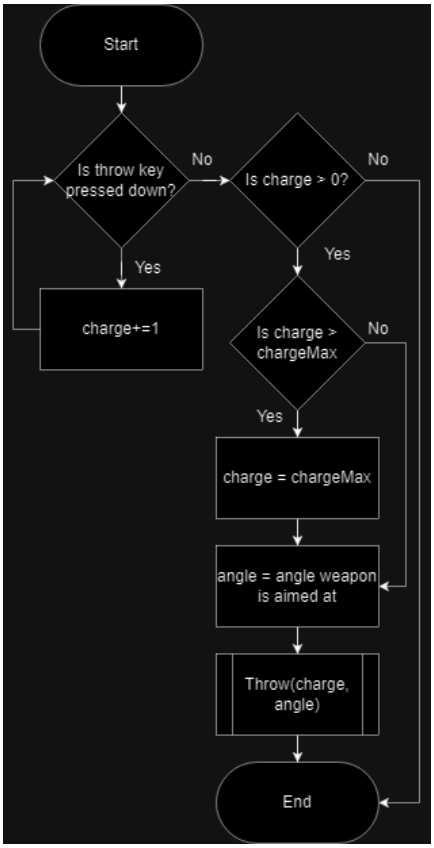
playerLeftHand	Transform	The player's left hand	
rightHandSO	Scriptable object	Stores the scriptable object for the player's right hand melee weapon	
leftHandSO	Scriptable object	Stores the scriptable object for the player's left hand weapon	

Algorithmic Design:

The flowchart below on the left represents the logic I will use for shooting and reloading the player's ranged weapon. And on the right the logic I will use for the melee weapon's attack ability.



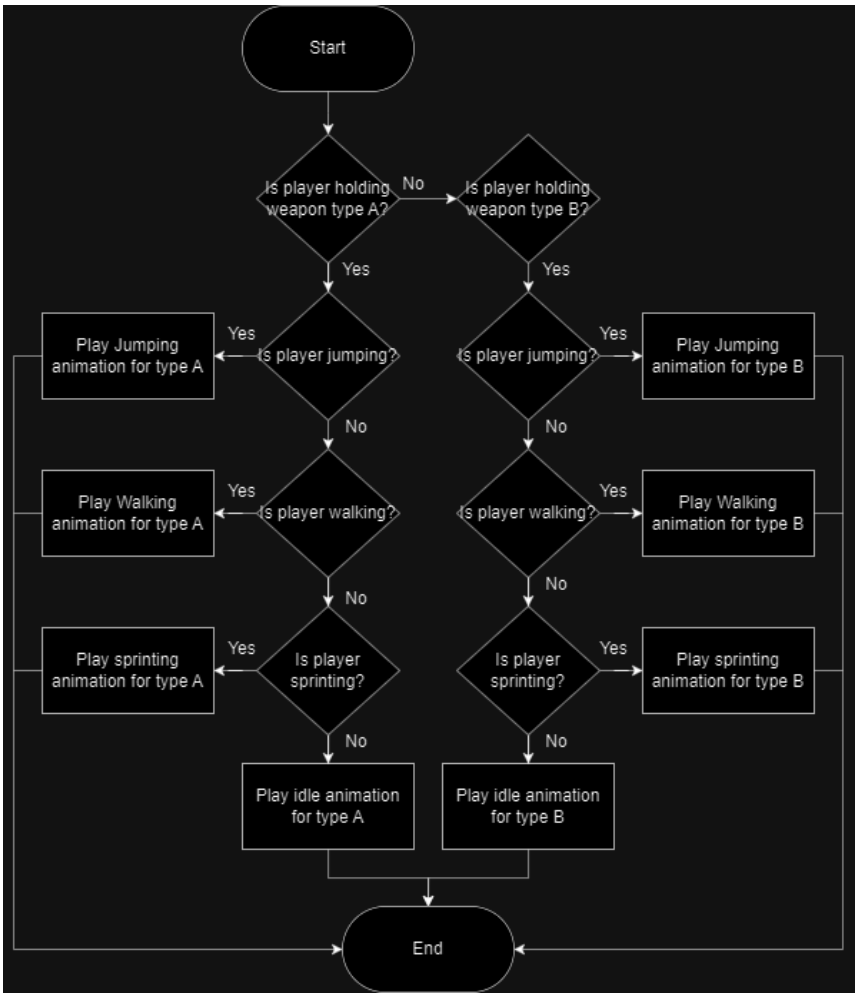





```
Function Throw(int charge, float angle)
    //Creates the movement vector for the projectile
    //and sets the intial vertcial and horizontal velocity
    Vector3 movDir = new Vector3()
    movDir.z = cos(angle)
    movDir.y = sin(angle)
    //multiplies the magnitude of the vector by the value of charge
    movDir *= charge

    //repeats the while loop until object hits something
    While(isInAir)
        //updates pos of prohectile with "move" function
        projectile.move(movDir*Time)
        //updates vertical velocity of projectile
        movDir.y += gravity*Time
    Endwhile
endfunction
```

The flow chart to the left represents the logic I will use to activate the throw ability and to charge the throw. Additionally, the pseudocode above gives a general outline for the code I will write for the “Throw” function which will handle the logic for throwing the weapon.



To flowchart to the left demonstrates the logic I will use to play different sets of animations for each weapon type.

Development:

During the second stage, I started off by tidying up the project and importing some weapon models for the player.

I then moved on to creating a pick up and drop ability for the player, initially I created a duplicate player model and worked on positing a bat in the player's right hand and wrapping the player's fingers around it.

Next, I was going to move onto the actual script, however, I noticed that the player interaction detection wasn't always working. I first tried to solve this by increasing the range of the raycast which didn't seem to have any effect, so I then switched to a capsule cast to detect any interactable objects in front of the player from head to toe which seemed to fix the issue.

```
//checking if object in the interact layer is within 2 units of the player in the direction they're looking
if(Physics.CapsuleCast(transform.position + Vector3.up * playerHeight,transform.position,.5f,movDir,out RaycastHit hitInfo,2f,interactLayer)) {
    Debug.Log("Interact!");
}
```

Afterwards, I started working on the script to pick up the bat using some code I found in an online forum to help me implement it. For the pickup ability after clicking the interact key I check if the object "hit" by the capsule cast has the "Weapon" tag and weaponInhand is false, if so, I then proceed to make the player's righthand the parent of the object setting its local rotation and position to make it appear as if the player is holding the weapon. Followed by setting the weaponInHand bool to true so the player cannot pick up anymore weapons as well as making it kinematic so it's unaffected by physics.

```
    //checking if object in the interact layer is within 2 units of the player in the direction they're looking
    if(Physics.CapsuleCast(transform.position + Vector3.up * playerHeight,transform.position,.5f,movDir,out RaycastHit hitInfo,2f,interactLayer)) {
        if(hitInfo.transform.CompareTag("Weapon")&&weaponInHand==false) {
            weaponRef = hitInfo.transform;
            weaponRef.GetComponent<Rigidbody>().isKinematic = true;
            weaponRef.parent = playerRightHand;
            weaponRef.localRotation = Quaternion.Euler(new Vector3(15f,-15f));
            weaponRef.localPosition = new Vector3(-0.0696f,-0.0002f,-0.0032f);
            weaponInHand = true;
            Debug.Log("Weapon in hand = "+weaponInHand.ToString());
        }
        Debug.Log("Interact!");
    }
}
```

As for the drop ability I check if the player has a weapon in hand and the drop key has been pressed, if so, I set the weapon's parent to null and no longer make the weapon kinematic so it can be affected by physics again.

```
private void HandleControls() {
    //checking if weaponInHand is true and Q has been pressed
    if(weaponInHand && Input.GetKeyDown(KeyCode.Q)) {
        weaponRef.GetComponent<Rigidbody>().isKinematic = false;
        weaponRef.parent = null;
        weaponInHand = false;
        Debug.Log("Weapon dropped");
    }
}
```

I then started working on the player's attack ability first off by creating an attack swing animation and combining it with the idle animation. Next, I added the new bools isAttacking and isMelee to help track which animation states need to be played and subsequently added it to the player animator so I could test it.

```
public bool IsAttacking { get { return isAttacking; } }
1 reference
public bool IsMelee { get { return isMelee; } }
```

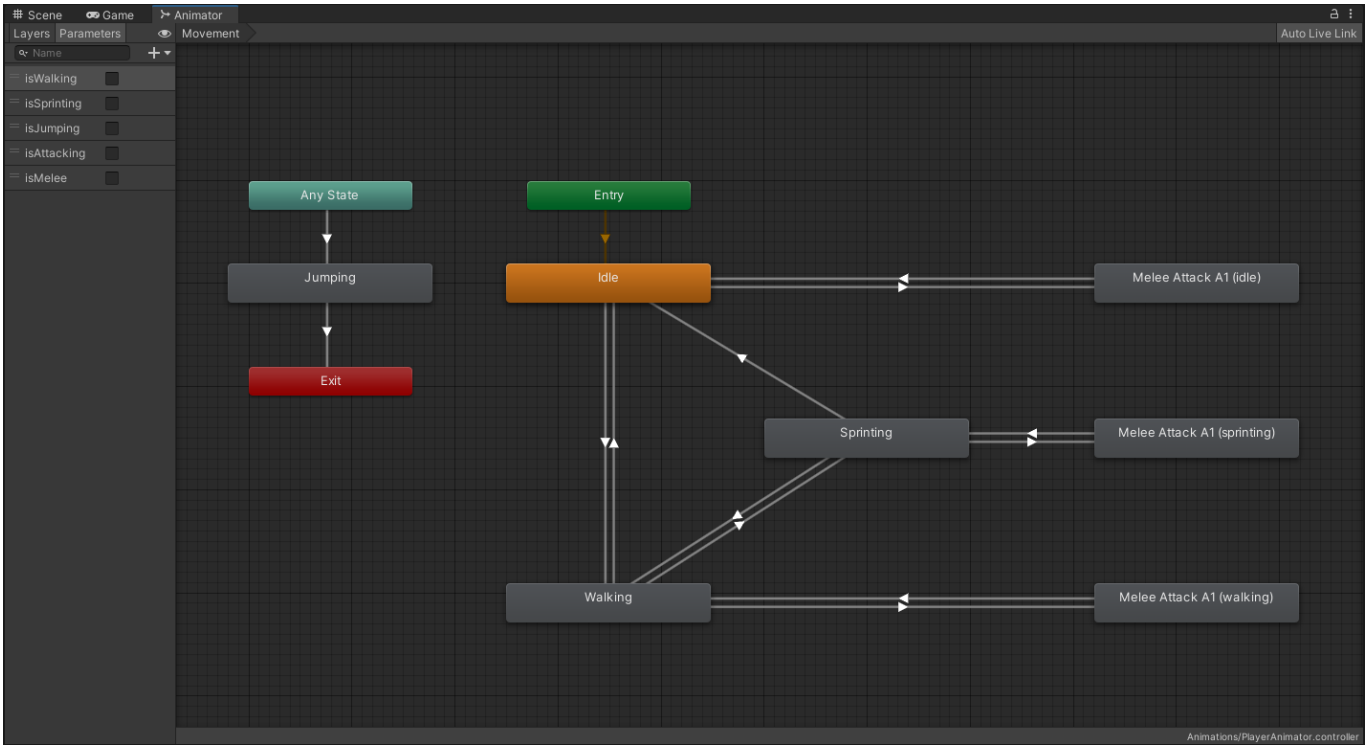
```
private void Update() {
    animator.SetBool("isWalking",player.IsWalking);
    animator.SetBool("isSprinting",player.IsSprinting);
    animator.SetBool("isJumping",player.IsJumping);
    animator.SetBool("isAttacking",player.IsAttacking);
    animator.SetBool("isMelee",player.IsMelee);
}
```

For now, isAttacking becomes true for the update the player presses the attack key and weaponInHand is true and isMelee if the weapon in the player's hand when isAttacking becomes true is "BaseballBat".

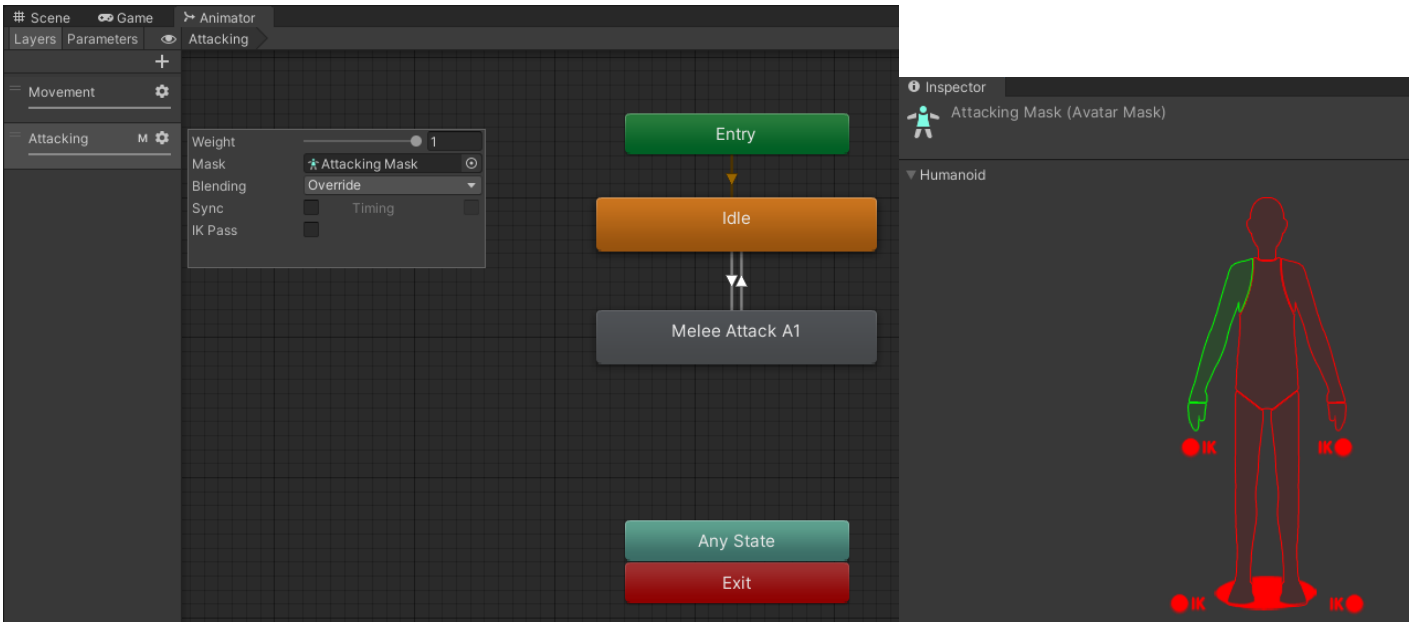
```
//resetting isAttacking
isAttacking = false;
if(weaponInHand && Input.GetKeyDown(KeyCode.Q)) {
    weaponRef.GetComponent<Rigidbody>().isKinematic = false;
    weaponRef.parent = null;
    weaponInHand = false;
    isMelee=false;
    Debug.Log("Weapon dropped");
} else if(weaponInHand && Input.GetKeyDown(KeyCode.Mouse0)) {
    isAttacking = true;
    Debug.Log("Attack!");
}
```

```
if(hitInfo.transform.CompareTag("Weapon")&&weaponInHand==false) {
    weaponRef = hitInfo.transform;
    weaponRef.GetComponent<Rigidbody>().isKinematic = true;
    weaponRef.parent = playerRightHand;
    weaponRef.localRotation = Quaternion.Euler(new Vector3(15f,-15f));
    weaponRef.localPosition = new Vector3(-0.0696f,-0.0002f,-0.0032f);
    weaponInHand = true;
    Debug.Log("Weapon in hand = "+weaponRef.name);
    if(weaponRef.name == "BaseballBat") {
        isMelee = true;
        Debug.Log("Weapon is melee");
    }
}
Debug.Log("Interact!");
```

Thereafter I combined the swing animation with the walk and sprint animation and added it to the player animator.



However, after testing I realized that I couldn't transition between movement animations whilst maintaining the attack swing if they were combined and rather, I had to play two animations simultaneously so after looking around online I figured out I needed to create a separate layer for the attack animations and add an avatar mask to only override the animation on the right arm.

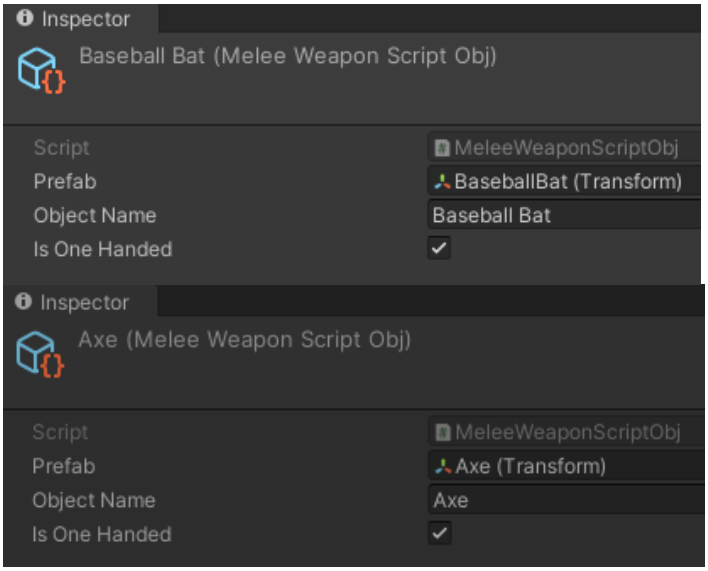


After doing so and testing it a few times I decided to move on and create a scriptable object for the player's melee weapons to make retrieving info about them much simpler. For now, I have simply given them 3 attributes the weapon's prefab, name and whether it is one handed or not.

```
[CreateAssetMenu()]
Unity Script | 0 references
public class MeleeWeaponScriptObj:ScriptableObject {
    [SerializeField] private Transform prefab;
    [SerializeField] private string objectName;
    [SerializeField] private bool isOneHanded;

    0 references
    public Transform GetPrefab() {
        return prefab;
    }
    0 references
    public string GetObjectName() {
        return objectName;
    }
    0 references
    public bool GetIsOneHanded() {
        return isOneHanded;
    }

}
```



Next, I started to refactor the pickup code a bit so the player could dual wield and remove any now unnecessary variables. I did this by first replacing weaponInHand with rightHandWeapon and leftHandWeapon to check the players hands separately and now simply attaching the tag “Melee” instead of checking the objects name to check if it is a melee weapon.

```
weaponRef = hitInfo.transform;
if(weaponRef.CompareTag("Melee")) {
    if(rightHandWeapon == false) {
        weaponRef.GetComponent<Rigidbody>().isKinematic = true;
        weaponRef.parent = playerRightHand;
        weaponRef.localRotation = Quaternion.Euler(new Vector3(15f,-15f));
        weaponRef.localPosition = new Vector3(-0.0696f,-0.0002f,-0.0032f);

        rightHandWeapon = true;
        Debug.Log("Right Weapon in hand = " + weaponRef.name);
    } else if(rightHandWeapon == true && leftHandWeapon==false) {
        weaponRef.GetComponent<Rigidbody>().isKinematic = true;
        weaponRef.parent = playerLeftHand;
        weaponRef.localRotation = Quaternion.Euler(new Vector3(15f,-15f));
        weaponRef.localPosition = new Vector3(-0.0696f,-0.0002f,-0.0032f);

        leftHandWeapon = true;
        Debug.Log("Left Weapon in hand = " + weaponRef.name);
    }
}
```

```
if(Input.GetKeyDown(KeyCode.Q)) {
    if(leftHandWeapon==true) {
        leftHandWeapon=false;
        weaponRef = playerLeftHand.transform.GetChild(5).transform;
        weaponRef.GetComponent<Rigidbody>().isKinematic = false;
        weaponRef.parent = null;
        Debug.Log("Left Weapon dropped");
    } else if (rightHandWeapon==true) {
        rightHandWeapon=false;
        weaponRef = playerRightHand.transform.GetChild(5).transform;
        weaponRef.GetComponent<Rigidbody>().isKinematic = false;
        weaponRef.parent = null;
        Debug.Log("Right Weapon dropped");
    }
```

I then worked on retrieving the corresponding scriptable object to the weapon the player picked up. Initially I was unsure of a good way to add this and so after looking around a bit I decided to create a new script “WeaponList” to store all the scriptable objects for melee weapons and a function to retrieve them through the name of the object.

```
public class WeaponList : MonoBehaviour {
    List<MeleeWeaponScriptObj> meleeWeapons = new List<MeleeWeaponScriptObj>();
    [SerializeField]private MeleeWeaponScriptObj baseballBat, axe;

    Unity Message | 0 references
    void Start() {
        meleeWeapons.Add(baseballBat);
        meleeWeapons.Add(axe);
    }
    2 references
    public MeleeWeaponScriptObj FindSO(string objectName) {
        for(int i = 0; i < meleeWeapons.Count; i++) {
            if(string.Equals(meleeWeapons[i].GetObjectName(),objectName)) {
                Debug.Log(meleeWeapons[i].GetObjectName());
                return meleeWeapons[i];
            }
        }
        return null;

    }
}
```

```
weaponRef = hitInfo.transform;
if(weaponRef.CompareTag("Melee")) {
    if(rightHandWeapon == false) {
        rightHandSO = weaponList.FindSO(weaponRef.name);
        isOneHanded = rightHandSO.IsOneHanded();

        weaponRef.GetComponent<Rigidbody>().isKinematic = true;
        weaponRef.parent = playerRightHand;
        weaponRef.localRotation = Quaternion.Euler(new Vector3(15f,-15f));
        weaponRef.localPosition = new Vector3(-0.0696f,-0.0002f,-0.0032f);
        rightHandWeapon = true;
        Debug.Log("Right Weapon in hand = " + weaponRef.name);
        //checking if weapon is 2 handed
        if(isOneHanded==false) {
            leftHandWeapon = true;
            Debug.Log("Left Weapon in hand = " + weaponRef.name);
        }
        //checking player is holding weapon in right hand and left hand is free
    } else if(rightHandWeapon == true && leftHandWeapon==false) {
        leftHandSO = weaponList.FindSO(weaponRef.name);
        if (leftHandSO.IsOneHanded()==true) {
            weaponRef.GetComponent<Rigidbody>().isKinematic = true;
            weaponRef.parent = playerLeftHand;
            weaponRef.localRotation = Quaternion.Euler(new Vector3(15f,-15f));
            weaponRef.localPosition = new Vector3(-0.0696f,-0.0002f,-0.0032f);
            leftHandWeapon = true;
            Debug.Log("Left Weapon in hand = " + weaponRef.name);
        } else {
            Debug.Log("Hands are full");
        }
    }
}
```

I then made use of this in the player script storing the scriptable objects for the melee weapons in the player's right hand and left hand in rightHandSO and leftHandSO respectively. Moreover, I then used the IsOneHanded method (renamed GetIsOneHanded to IsOneHanded) so the player could not pick up another weapon if holding a 2 handed weapon or 2 one handed weapons and not pick up a 2 handed weapon if already holding a 1 handed weapon.

After testing the weapons some more, I discovered a bug where the player can pick up the weapon they are holding in their right hand with their left hand (Demonstrated in "Picking up right hand weapon in left.mkv"). To fix this I simply had to check the object being picked up in the left hand does not have the player's right hand as a parent.

```
else if(rightHandWeapon == true && leftHandWeapon==false && weaponRef.parent != playerRightHand)
```

With the main features done for the melee weapons I decided to move on to creating the ranged weapons. I began by creating a scriptable object for ranged weapons and updating the "WeaponList" script to include ranged weapons.

```
public class WeaponList : MonoBehaviour {
    List<MeleeWeaponScriptObj> meleeWeapons = new List<MeleeWeaponScriptObj>();
    [SerializeField] private MeleeWeaponScriptObj baseballBat, axe;

    List<RangedWeaponSO> rangedWeapons = new List<RangedWeaponSO>();
    [SerializeField] private RangedWeaponSO revolver;

    Unity Message | 0 references
    void Start() {
        //adding scribtable objects to list for melee weapon
        meleeWeapons.Add(baseballBat);
        meleeWeapons.Add(axe);
        //same for ranged weapons
        rangedWeapons.Add(revolver);
    }
    //finds scriptable object with weapon's name
    2 references
    public MeleeWeaponScriptObj FindMeleeSO(string objectName) {
        for(int i = 0; i < meleeWeapons.Count; i++) {
            if(string.Equals(meleeWeapons[i].GetObjectName(),objectName)) {
                Debug.Log(meleeWeapons[i].GetObjectName());
                return meleeWeapons[i];
            }
        }
        return null;
    }
    1 reference
    public RangedWeaponSO FindRangedSO(string objectName) {
        for(int i = 0; i < rangedWeapons.Count; i++) {
            if(string.Equals(rangedWeapons[i].GetObjectName(),objectName)) {
                Debug.Log(rangedWeapons[i].GetObjectName());
                return rangedWeapons[i];
            }
        }
        return null;
    }
}
```

```
[CreateAssetMenu()]
Unity Script | 5 references
public class RangedWeaponSO:ScriptableObject {
    [SerializeField] private Transform prefab;
    [SerializeField] private string objectName;

    0 references
    public Transform GetPrefab() {
        return prefab;
    }
    2 references
    public string GetObjectName() {
        return objectName;
    }
}
```

I then began to work on the player script adding the ability for the player to pick up and drop ranged weapons similarly to the way I did for melee weapons alongside tidying up large chunks of code by replacing them with subroutines.

```
weaponRef = hitInfo.transform;
//checking if weapon is melee through "Melee" tag
if(weaponRef.CompareTag("Melee"))...
else if(weaponRef.CompareTag("Ranged")) {
    if(rangedWeapon == false) {
        Debug.Log("Picked up ranged Weapon");
        rangedWeaponSO = weaponList.FindRangedSO(weaponRef.name);
        IntitialiseWeapon(weaponRef,floatingGun,Quaternion.identity,Vector3.zero);
        rangedWeapon = true;
        if(IsMeleeEquipped()) {
            if(rightHandWeapon == true) {
                ToggleWeapon(playerRightHand,5,false);
            }
            if(leftHandWeapon == true && isOneHanded == true) {
                ToggleWeapon(playerLeftHand,5,false);
            }
        }
    }
}
```

```
if(Input.GetKeyDown(KeyCode.Q)) {
    if(IsMeleeEquipped())... else {
        if(rangedWeapon) {
            rangedWeapon = false;
            DropWeapon(floatingGun,0);
            Debug.Log("Gun dropped");
        }
    }
```

Next, I moved onto the ability for the player to switch between their melee and ranged weapon. I did this by setting the active state of the player's melee weapon(s) to true and range weapon to false when the player pressed 1 and had a melee weapon(s) equipped. And I then did the opposite for when the player pressed 2 and had a ranged weapon equipped. Moreover, I made sure to toggle off the weapon currently in the player's hand (if they had one) when picking up a different type of weapon e.g. picking up melee with a ranged weapon is in the player's hand.

```
if((Input.GetKeyDown(KeyCode.Alpha1) && IsMeleeEquipped() == false)|| (Input.GetKeyDown(KeyCode.Alpha2) && IsMeleeEquipped() == true && !rightHandWeapon)) {
    if(rightHandWeapon == true) {
        ToggleWeapon(playerRightHand,5);
    }
    if(leftHandWeapon == true && isOneHanded == true) {
        ToggleWeapon(playerLeftHand,5);
    }
    if(rangedWeapon == true) {
        ToggleWeapon(floatingGun,0);
    }
}
//checks if player has no weapon in right hand
if(rightHandWeapon == false) {
    //finding scriptable object for weapon
    rightHandSO = weaponList.FindMeleeSO(weaponRef.name);
    isOneHanded = rightHandSO.IsOneHanded();
    IntitialiseWeapon(weaponRef,playerRightHand);
    rightHandWeapon = true;
    Debug.Log("Right Weapon in hand = " + weaponRef.name);

    //sets leftHand bool to true if weapon is 2 handed
    if(isOneHanded==false) {
        leftHandWeapon = true;
        Debug.Log("Left Weapon in hand = " + weaponRef.name);
    } if(rangedWeapon) {
        ToggleWeapon(floatingGun,0,false);
    }
```

```
    //checks if player only has weapon in right hand and picked object isn't the weapon in their right hand
    else if(rightHandWeapon == true && leftHandWeapon==false && weaponRef.parent != playerRightHand) {
        leftHandSO = weaponList.FindMeleeSO(weaponRef.name);
        //makes sure weapon isn't two handed
        if (leftHandSO.IsOneHanded()==true) {
            IntitialiseWeapon(weaponRef,playerLeftHand);
            leftHandWeapon = true;
            Debug.Log("Left Weapon in hand = " + weaponRef.name);
        } else {
            Debug.Log("Hands are full");
        }if(rangedWeapon) {
            ToggleWeapon(floatingGun,0,false);
        }
```

```
if(rangedWeapon == false) {
    Debug.Log("Picked up ranged Weapon");
    rangedWeaponSO = weaponList.FindRangedSO(weaponRef.name);
    IntitialiseWeapon(weaponRef,floatingGun,Quaternion.identity,Vector3.zero);
    rangedWeapon = true;
    if(IsMeleeEquipped()) {
        if(rightHandWeapon == true) {
            ToggleWeapon(playerRightHand,5,false);
        }
        if(leftHandWeapon == true && isOneHanded == true) {
            ToggleWeapon(playerLeftHand,5,false);
        }
    }
}
```

After some quick testing I realised the player could no longer switch back to a ranged weapon when a melee weapon was equipped and when only a ranged weapon was equipped 1 would equip and unequip the weapon instead of 2 (demonstrated in "gun equipping bug 1.mkv"). To fix this I added `&&rightHandWeapon` to the 1st condition as otherwise it would toggle the player's ranged weapon if no melee weapon was equipped as `IsMeleeEquipped()` was false when the active state of the player's melee weapon(s) was false, or the player hadn't picked up a melee weapon rather than just when the active state of the player's melee weapon(s) was false. I also changed `IsMeleeEquipped()`

`== true && !rightHandWeapon` in the 2nd condition to (`IsMeleeEquipped() == true || !rightHandWeapon`) as `IsMeleeEquipped()` was only true when `rightHandWeapon` was true so it would always return false.

```
if((Input.GetKeyDown(KeyCode.Alpha1) && !IsMeleeEquipped() && rightHandWeapon) || (Input.GetKeyDown(KeyCode.Alpha2)) && (!rightHandWeapon || IsMeleeEquipped())) {
```

However, I then quickly encountered another bug where when equipping a 2nd melee weapon with a gun currently in the player's hand 1 would now equip the 1st melee weapon and the gun and 2 would equip the 2nd melee weapon (demonstrated in "gun equipping bug 2.mkv"). I soon discovered this was a result of me not toggling the player's right hand melee weapon to true when picking up a 2nd melee weapon. To solve this, I simply added a line to toggle the active state of the player's right hand melee weapon to true when equipping a 2nd melee weapon.

```
else if(rightHandWeapon == true && leftHandWeapon==false && weaponRef.parent != playerRightHand) {
  leftHandSO = weaponList.FindMeleeSO(weaponRef.name);
  //makes sure weapon isn't two handed
  if (leftHandSO.IsOneHanded()==true) {
      IntitialiseWeapon(weaponRef,playerLeftHand);
      leftHandWeapon = true;
      Debug.Log("Left Weapon in hand = " + weaponRef.name);
  } else {
      Debug.Log("Hands are full");
  }if(rangedWeapon) {
      ToggleWeapon(floatingGun,0,false);
      ToggleWeapon(playerRightHand,5,true);
  }
```

With picking up and switching weapons out of the way I began to work on creating a shooting ability for ranged weapons. I did this by firing a raycast from the gun in the direction the player is looking up to the gun's shooting range and checking if anything in the hittable layer mask got hit and returning an appropriate message.

```
else if(Input.GetKeyDown(KeyCode.Mouse0) && (rightHandWeapon || rangedWeapon)) {
  isAttacking = true;
  //Debug.Log("Attack!");
  if(!IsMeleeEquipped()&&rangedWeapon) {
      if(Physics.Raycast(floatingGun.position,getMovDir(),rangedWeaponSO.GetMaxShootingRange(),hittableLayer)) {
          Debug.Log("Something got hit");
      } else {
          Debug.Log("You can't shoot");
      }
  }
```

Although upon testing the raycast did not seem to be consistent in firing so I used Debug.DrawRay to visualise the raycast and try and fix the bug (demonstrated in "gun shooting bug.mkv"). After testing it out I realised the issues simple was I was using the player's movement vector opposed to the direction the player is looking for the direction to fire the gun.

```
if(Physics.Raycast(floatingGun.position,this.transform.forward,rangedWeaponSO.GetMaxShootingRange(),hittableLayer))
```

```
//ammo functions
0 references
public void Shoot() {
    ammoInGun -= 1;
    if(ammoInGun == 0) {
        Reload();
    }
}
1 reference
public void Reload() {
    //checking if gun is already fully loaded and player has ammo to reload
    if (ammoInGun!=maxAmmoInGun&&ammo!=0) {
        //subtracting the amount that needs to be loaded from ammo
        ammo -= (maxAmmoInGun - ammoInGun);
        //checking if gun can be fully loaded
        if(ammo >= 0) {
            ammoInGun = maxAmmoInGun;
        }
        //if not partially loading gun with what is remaining
        else {
            ammoInGun = maxAmmoInGun + ammo;
            ammo = 0;
        }
    }
}
```

I then added the ammo mechanics for the gun to do this I used the variables ammoInGun to track how much ammo is currently in the player's weapon, maxAmmoInGun to determine the maximum number of shots the player can take before reloading and ammo to track how much ammo the player has the reload the weapon. Furthermore, I used the method shoot to decrement ammoInGun by 1 unless ammoInGun is 0 in which it calls the reload method which can also manually be called by pressing R if the weapon is not "full" (ammoInGun==maxAmmoInGun) and there is ammo to reload (ammo!=0).

Next, I decided to move the raycast hit logic for the ranged weapon inside the shoot method and add a delay between each shot and another for reloading.

```
else if(Input.GetKeyDown(KeyCode.Mouse0) && (rightHandWeapon || rangedWeapon)) {
    isAttacking = true;
    //Debug.Log("Attack!");
    if(!IsMeleeEquipped()&&rangedWeapon&&counter<=0) {
        Debug.Log("Shot fired");
        rangedWeaponSO.Shoot(floatingGun.position,playerCamera.forward,hittableLayer);
        counter = rangedWeaponSO.GetShootDelay()*Time.deltaTime;
    } else if(counter>0) {
        counter-=Time.deltaTime;
    }
```

```
public void Reload() {
    //reload delay
    Debug.Log("Reloading...");
    Thread.Sleep(reloadDelay * 1000);
    Debug.Log("Before Reloading - Ammo: "+ammo+" Ammo in gun: "+ammoInGun);
    //checking if gun is already fully loaded and player has ammo to reload
    if (ammoInGun!=maxAmmoInGun&&ammo!=0) {
        //subtracting the amount that needs to be loaded from ammo
        ammo -= (maxAmmoInGun - ammoInGun);
        //checking if gun can be fully loaded
        if(ammo >= 0) {
            ammoInGun = maxAmmoInGun;
        }
        //if not partially loading gun with what is remaining
        else {
            ammoInGun = maxAmmoInGun + ammo;
            ammo = 0;
        }
    }
    Debug.Log("After Reloading - Ammo: " + ammo + " Ammo in gun: " + ammoInGun);
}
```

```
public void Shoot(Vector3 pos, Vector3 dirVector, LayerMask hittableLayer) {
    if(ammoInGun == 0) {
        Reload();
    } else {
        ammoInGun -= 1;
        if(Physics.Raycast(pos,dirVector,maxShootingRange,hittableLayer)) {
            //Debug.Log("Something got hit");
        } else {
            //Debug.Log("You can't shoot");
        }
    }
```

After testing the shoot delay it seemed very inconsistent so after reviewing my code (demonstrated in “shoot delay bug.mkv” the shoot delay in the video is 5 seconds), I realised counter was only being deducted when the attack key was pressed down so instead of multiplying the value of GetShootDelay() by Time.deltaTime I simply had to deduct the value of GetShootDelay() by Time.deltaTime every update.

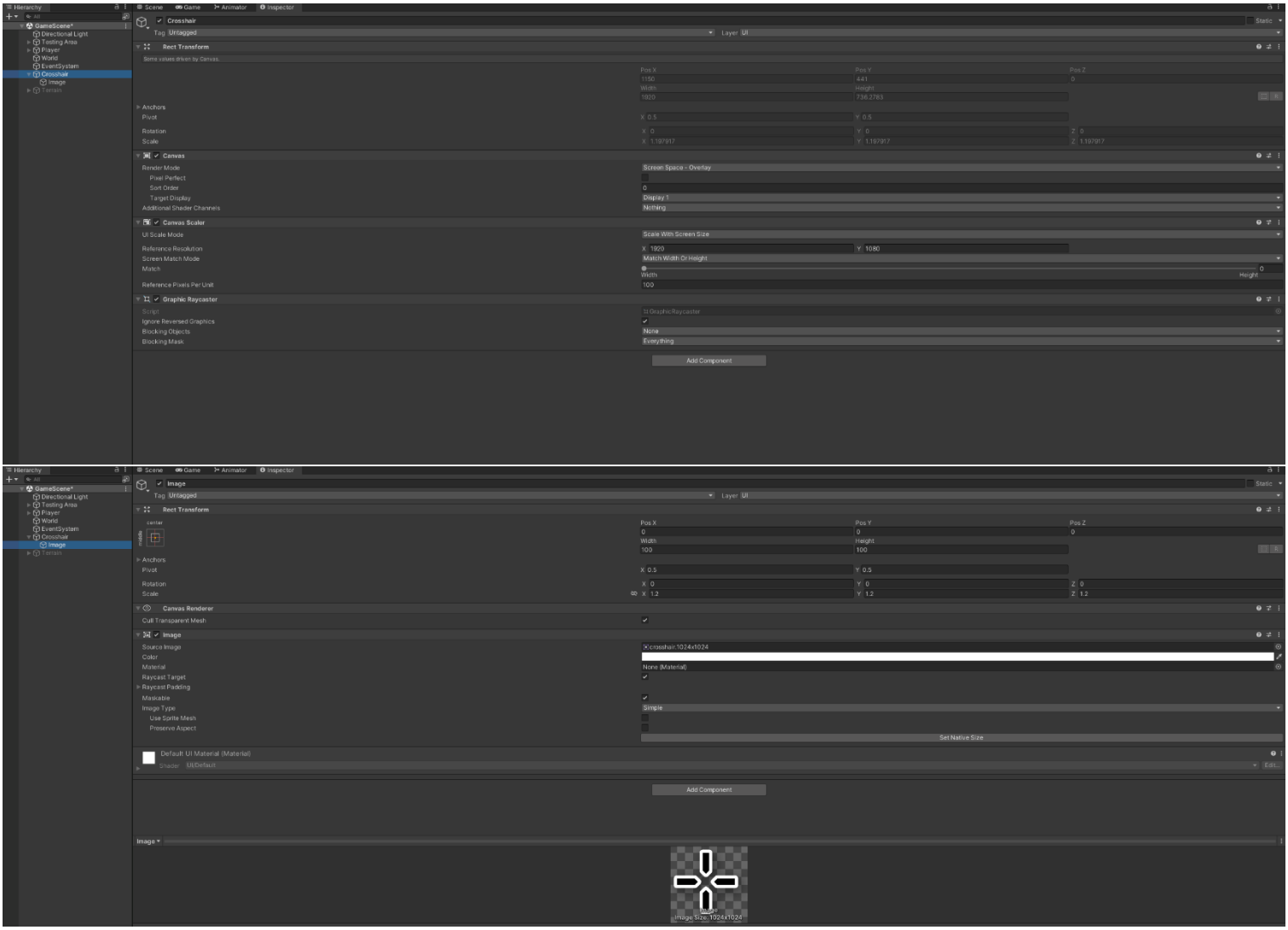
```
1 reference
private void HandleControls() {
    //resets isAttacking each update so anim doesn't play continuous
    isAttacking = false;
    //drops the weapon currently in the player's hand
    if(!AttackAnimPlaying())...
    if(counter>0) {
        counter -= Time.deltaTime;
    }

}
```

Additionally, when testing the reload delay I realised Thread.Sleep would delay the whole program (demonstrated in “reload bug 1.mkv” the reload delay in the video is 2 seconds) so after looking around I decided to try a while loop delay however the program seemed to just skip it/do it instantly (demonstrated in “reload bug 2.mkv” the reload delay in the video is 2 seconds) so I then tried using Invoke however I was unable to call it within a scriptable object so since I was short on time and did not see a clear way to solve this I left the reload method without any delay.

```
float i=0;
//reload delay
Debug.Log("Reloading...");
while(i<reloadDelay) {
    i += Time.deltaTime;
}
```

Finally, I added a crosshair in the middle of the player’s screen. Since I was initially unsure of how to create an overlay, I followed a tutorial online which showed me I had to create a canvas and then simply attach and position an image of the crosshair.



Test Table:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Player can switch between melee and ranged weapons using their respective equip keys	Use equip keys	Player will switch to their melee weapon(s) when pressing 1 if picked up and to their ranged weapon when pressing 2 if picked up	Upon pressing 1 the player switches to their melee weapon and when pressing 2 they switch to their ranged weapon
1b	Player weapon shows up in hand correctly	Equip weapon	Weapon is correctly positioned in hand	All weapons when equipped correctly show up in their designated positions
2a	Player can only carry a limited number of ammo	Try and pick up ammo when max ammo in inventory	Player cannot pick up extra ammo	TBA
2b	Player can only reload when weapon is not full	Try and reload when weapon is full	Player cannot reload	Ammo in gun remains the same

				and no ammo outside the gun is deducted
2c	Ammo is correctly adjusted upon reload	Reload weapon	Weapon is filled and amounted added is deducted from inventory	Ammo needed is deducted and if ammo=>0 gun is filled otherwise gun is filled with ammo that is left and ammo is set to 0
2d	Reload cancels correctly	Reload weapon and switch or use weapon ability while doing so	Reload cancels when player switches weapon or uses a weapon ability	TBA
3	Crosshair remains in centre of screen	Rotate head	Crosshair will remain in centre of screen	Crosshair does not move from position in screen
4a	Correctly directs weapon's attack animation at crosshair	Left click with melee weapon or nothing equipped	Will print "hit" in the console if the player's crosshair is over a hittable object when left clicking and "not hit" if there is no hittable object	TBA
4b	Correctly detects hittable objects the player's weapon touches	Left click in front of object in the "hittable" layer with melee weapon or nothing equipped	Will print "hit" in the console if the player's weapon touches a hittable object and "not hit" if it touches nothing during the whole animation	Prints "hit" in console if capsule cast hits (not object itself) an object in the hittable layer mask and "not hit" if it doesn't.
4c	Player can only walk at a reduced speed during the attack animation	Try and use other controls during the attack animation	Player will only be able to move at a reduced speed	Player cannot sprint/ use any controls during attack animation. However, reduced speed is TBA
5a	Object is thrown where player is looking	Throw melee weapon	Weapon will be thrown in direction player is looking	TBA
5b	Melee weapon's throw charges correctly	Hold throw button for different lengths of time when holding melee weapon	Object's initial velocity will increase depending on how long player holds the throw button and will stop increasing after a set amount of time	TBA

5c	Thrown weapon responds appropriately when hitting objects	Throw weapon at a hittable and non-hittable object	Object lodges into hittable objects and bounces off non-hittable objects and can be retrieved using the interact key	TBA
5d	Player's arm correctly adjusts as player charges throw	Hold throw button when holding melee weapon	Player's arm will gradually rotate clockwise as player charges weapon	TBA
5e	Player cannot sprint when touching throw key	Hold throw and sprint key	Player will charge throw	TBA
6a	Game zooms in when aim key is held	Hold aim key with ranged weapon	Screen will zoom in towards crosshair with the magnification varying on the weapon	TBA
6b	Player cannot sprint and walks at a reduced speed whilst aiming	Try and use other controls whilst aiming	Player will only be able to jump and walk slowly	TBA
6c	Player holds ranged weapon correctly when aiming	Aim	Player takes shooting stance	TBA
7a	Raycast correctly detects hittable object when player shoots	Shoot at objects in the "obstruction" and "hittable" layers within and outside the weapon's range	Will print "hit" in the console if it detects an object in the "hittable" layer before an object in the "obstruction" layer, "hit obstruction" if it detects an object in the "obstruction" layer first and "hit nothing" if it detects nothing within range	When the raycast is fired if it hits an object in the hittable layer mask it prints "hit" if the object is in the enemy layer and "hit" obstruction" if it hits something else and if they raycast misses it will print "hit nothing"
7b	Bullet hole decal is created when player hits object in the "obstruction" layer	Shoot at an object in the "obstruction" layer	Bullet hole decal momentarily placed where raycast hit obstruction object	TBA
7c	Impact animation if player hits a visible object	Shoot at a visible object	Impact animation plays where raycast hit object with a "spray effect" which colour varies depending on the object it hits	TBA
7d	Muzzle flash animation plays when player fires	Shoot	Muzzle animation plays very briefly	TBA
7e	Brief cooldown in-between player's shots	Shoot	There will be a cooldown in-between	Ranged weapon has a cooldown between shots which can be

			each shot varying on the weapon	changed by altering the attributes of its scriptable object
7f	Player cannot sprint and walks at a reduced speed whilst shooting	Try and use other controls whilst shooting	Player will only be able to jump and walk slowly	TBA
7g	Automatic reload if player shoots with no ammo in weapon	Shoot when no ammo in ranged weapon	Reload ability will activate	The reload function is called if the player tries to shoot with no ammo in their gun
7h	Player holds ranged weapon correctly when firing	Shoot	Player takes same stance when aiming	Since ranged weapon now instead floats in front of player no stance is needed
8	Recoil effect plays after player fires bullet	Shoot	Player's screen will "shake" with its intensity varying on the weapon	TBA
9	Reload animation triggers and cancels correctly	Reload weapon and switch or use weapon ability while doing so	Reload animation triggers when reload key is pressed and cancels when reload is cancelled	TBA
10	Weapon sprinting animations trigger correctly	Sprint whilst holding a weapon	Weapon sprinting animation plays	TBA

Stage 3 (Enemies – 1 week):

For the third stage of development, I now will add the enemies as both the player and a means of killing the enemies have been implemented. Moreover, the aim of the game is to survive the enemies so without them all the player can do is use his weapons and wander around.

Features to implement:

Part 1:

1. Create/find visuals for zombies
2. Give zombies block collision
3. Create simple pathfinding for zombies
4. Set up movement animations for zombies
5. Set up health mechanics for player and zombies
6. Refactor ranged and melee weapon code so it damages zombie if there's a successful hit
7. Create script so zombies attempt attack on player when within range

8. Create attack animation for zombie and damage player if zombie touches player during the animation

Class Diagrams:

TBA

Data Dictionary:

TBA

Name	Data Type	Description	Validation

Algorithmic Design:

TBA

Development:

Test No.	Description	Test Data	Expected Result	Actual Result
1	Zombie cannot pass through visible objects	Make zombie move forward and place an object in front of it	Zombie will be stopped by object	
2a	Zombie moves towards player	Stand still and wait	Zombie will move towards the player	
2b	Zombie avoids objects obstructing its path	Obstruct zombie's path to the player	Zombie will avoid objects and move towards the player	
3	Animation trigger and play correctly	Make zombie move	All animations will play and trigger when they are supposed to	
4a	Player's and zombies' health do not go below 0	Let player be attacked by zombie and zombie be attacked by the player	Player and zombies' health will stop depleting at 0	
4b	Player's and zombies' health do not exceed the max	Make damage give health and let player and zombie be attacked by each other	Player and zombies' health will not increase at max health	
4c	Zombies disappear when they reach 0 health	Attack zombie with weapons	Zombie will disappear when it reaches 0 health	
5	Zombie is damaged appropriately when hit	Attack zombie with attack, throw and shoot abilities	Zombie will be damaged when successfully hit by any of the abilities	
6a	Zombie only attacks player when they are in front of them	Move around a stationary zombie	Zombie will try and attack player when they move in front of them	

6b	Zombie only reattacks after attack animation ends	Stand in front of zombie	Zombie will attack and then after the animation ends attempts to attack again	
7	Zombie only hits player once in attack animation	Stand in front of zombie	Player is only damaged upon the first instance of the zombie touching the player	

~~Stage 4 (Polishing – 1 week):~~ Skipped

1. Create visual for player healthbar which depletes when player loses health, fills when player regains health
2. Create visual for players currently equipped weapon and ammo stats displaying an icon and possibly name of the weapon alongside ammo in ranged weapon and ammo player is holding
3. Add a health and a few utility potions with some visuals
4. Add player movement abilities giving it a visual too
5. Create block ability for melee weapons
6. Add sound effects when player moves, jumps, attacks, blocks, reloads, hits enemy, takes damage, drinks a potion, uses a movement ability, is on low health
7. Add ambience sound effects
8. Add sound effects for when enemies move, attack, hit player, take damage and general noise they always make

Test No.	Description	Test Data	Expected Result	Actual Result
1	Player healthbar fills and depletes correctly	Let player take damage and heal with health potion	Payer healthbar will deplete and then fill a bit upon the player drinking the potion	
2a	Currently equipped weapon updates upon changing weapon	Switch and unequip player's weapon	Will update name, visual and ammo stats when the player changes weapon	
2b	Ammo stats correctly correspond to ammo in ranged weapon and ammo in inventory	Shoot and reload ranged weapon	Ammo in ranged weapon will decrease as player fires and then upon reloading ammo in ranged weapon will be set to the max and amount in inventory will be updated	
2c	Ammo in ranged weapon turns red when the ammo is low	Shoot with ranged weapon until ammo is low	Colour of ammo in ranged weapon stat will turn red	

3a	Player cannot drink a potion when they have 0	Try and drink a potion when potion count is 0	Nothing will happen	
3b	Potion count visual on-screen updates	Drink potion	Potion count for that potion will decrease by 1	
4a	Cooldown triggers when player uses movement ability	Use movement ability	Cooldown starts and clock cooldown effect starts, and time left is displayed to 1dp on in seconds on the movement ability visual	
4b	Player moves correctly in response to direction player is looking	Use movement ability	Player movement ability will propel player in response to direction player is looking e.g., forward for dash	
5	Test all added sounds play correctly for step 5, 6 and 7	Test all abilities that now have sounds	Each sound will trigger when an ability starts and stop when an ability cancels	

~~Stage 5 (Wave and Point Mechanics – 1 week):~~ Skipped

1. Create point system for player in which the player begins with 0 points and gains points for each kill
2. Vary points given depending on how powerful the enemy is
3. Create script to spawn enemies in proximity of the player at suitable locations
4. Create script to randomly spawn different types of enemies making them more powerful and introducing more powerful types as the waves progress
5. Increase number of zombies as waves progress
6. Create visual for wave count and number of points

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Points are received upon killing an enemy	Kill an enemy	Player will gain points	
2	Points received increases when the enemy is more powerful	Kill a weak and powerful enemy	Player will gain more points from the powerful enemy	

3a	Zombies can reach player after spawning	Surround player with obstacles and let zombies spawn	Zombies will not spawn inside objects and will have a route to pathfind to the player	
3b	Zombies do not spawn outside the proximity range	Let zombies spawn	Zombies will only spawn within the specified range of the player	
4a	More powerful zombie types only spawn in later waves	Check zombies spawning on first wave and the wave number each of the more powerful types should start spawning	More powerful zombie types will spawn on the waves they have been set to do so	
4b	Zombies get stronger as the wave count increases	Let same zombie type damage player on a low and high wave and let player kill same zombie type on a low and high wave	Player will receive more damage from the zombie on the high wave and player will need to deal more damage to kill it	
5	Zombie count increases as the waves count increases	Compare zombie count on different wave numbers	Zombie count will increase as the wave number increases	
6	Visuals update correctly	Kill an enemy and end a wave	Visual for players points will increase by amount enemy was worth and wave count visual will increment by 1	

Stage 6 (GUI – 3 weeks):

Part 1:

1. Create Main Menu screen
2. Create Difficulty Selection Menu
3. Create loading screen that has a progress bar and cycles through images
4. Create Pause Menu
5. Refactor wave mechanics code so difficulty affects how tough the waves are
6. Create Settings Menu

Part 2:

7. Create Loadout Menu
8. Design Game Over overlay
9. Refactor health mechanics' code so when player reaches 0 health it triggers Game Over Screen
10. Create End of Game stats screen

11. Add sounds for the menus e.g., clicking buttons, using sliders, etc...

Class Diagrams:

TBA

Data Dictionary:

TBA

Name	Data Type	Description	Validation

Algorithmic Design:

TBA

Development:

Test No.	Description	Test Data	Expected Result	Actual Result
1	GUI Buttons are all mapped and function correctly	Click each button	All buttons will direct the player to the appropriate screen or carry out the appropriate function e.g., settings button takes player to settings menu and exit game button will end the program	
2a	Player can only select one difficulty option	Try and click a second difficulty option after selecting a different one	The selected difficulty option will switch to the second one the user clicked	
2b	“PLAY” button lights up and can only be selected when the player has chosen a difficulty option	Try and click “PLAY” before and after selecting a difficulty option	Before the difficulty option is selected nothing happens but after a difficulty option has been chosen the play button will light up and upon pressing it the player will be sent to the loading screen	
3a	Loading screen cycles through images provided	Initiate loading screen in the Difficulty Selection Menu	Loading screen will change the image displayed at a set interval	

3b	Loading screen accurately displays progress	Initiate loading screen in the Difficulty Selection Menu	Progress bar will display an estimate for how far the program has progressed in preparing to start the game	
4a	Game goes into a “paused” state when player opens the Pause Menu	Click the esc key whilst in the game	Player and enemies will be frozen in position both unable to use any attacks or abilities	
4b	Pause Menu displayed correctly	Click the esc key whilst in the game	The player’s screen is blurred, and a Pause Menu overlay is displayed on top of it	
5	Difficulty changes with the difficulty option selected	Start the game with each difficulty option and compare zombies' speed, health, damage and count on different waves	Zombies’ will be faster on higher difficulties and waves will increase number and powerfulness of enemies faster	
6a	Drop down boxes behave correctly	Change setting option using a drop down box	When clicking the settings current option, a small box will appear below displaying all possible options and upon selecting an option the box will disappear and the settings option will be updated	
6b	Sliders increase and decrease within range	Drag slider bar from far left to far right and try to drag further when at min and max values	As player drags the slider bar to the right it will increase the setting’s value stopping at its max value when it reaches the right end of the bar and as the player drags it to the left the setting’s value will decrease stopping at its min value when it reaches the left end of the bar	
6c	Keybind text fields only allow one key	Click a current key bind and press any key	Upon pressing one of the currently assigned keys it will be replaced with the text “Please enter a new key” and the next key the user presses will be assigned to that binding except for the esc key which makes the binding unbound	

6d	Keys assigned to more than 1 binding are displayed in red	Assign the same key to 2 or more bindings	Upon assigning a key to a second binding both keys will appear in red	
6e	Reset key is only clickable when the default binding is changed	Try and press reset key before and after changing a key binding	Before changing a key binding nothing will happen but after the reset key will light up and upon clicking it will change the current key binding back to the default one and the reset key will return to normal	
7a	Player can increment and decrement potion count by 1 using arrow buttons	Hover over potion count and click arrow buttons	Potion count will increment by 1 and other potion count will decrement by 1 when player presses an up arrow button and the opposite when the player presses a down arrow button	
7b	Player cannot make individual potion count fall below 0 or above 6	Click down arrow button when potion count is 0 and up arrow button when it is 6	Potion count will remain the same	
7c	Selection Menu can be closed by clicking the background	Open selection menu by clicking the appropriate icon and then click the background	Player will exit the selection menu	
7d	Player can cycle through option in selection menu using arrow buttons	Press left and right arrow buttons in the selection menu	Upon clicking an arrow button, it will display the next available options cycling back to the start when at the end of the options	
7e	Hue changes as player drags slider in colour picker menu	Drag slider in colour picker menu	Hue of shade box and current colour updates to hue at position of slider	
7f	Player can drag point to change shade	Drag point in shade box	Shade of colour displayed in the current colour will change to match shade at point in the box	
7g	Colour is updated and displayed on player model after exiting colour picker menu	Change colour and exit colour picker menu	Player model update and player model colour icon will change to match new colour	
8	End of Game screen appears	Let player die to zombie	Upon the player's death the Game Over screen will appear	

9	Verify end of game stats are accurate	Record game stats externally and compare results to end of game stats screen	External results will match end of game stats screen	

Stage 7 (Map – 1 week):

1. Create map terrain adding giant walls at its borders
2. Decorate the maps with some props and buildings scattered across the map
3. Create buyable weapons which the player can purchase using points and repurchase to replenish the weapon's ammo
4. Create/find ammo box model
5. Create script so player can replenish some ammo of held weapon after picking up ammo box
6. Create script to scatter items across the map at suitable locations
7. Create/find model for magical forge
8. Create script so player can upgrade weapon if they have the correct number of points and make it available to be picked up after it's done upgrading
9. Create script so magical forge appears in a random location after the 10^{th} wave and moves elsewhere after a set number of uses

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Player can only buy weapons when points match or exceed the required threshold	Attempt to purchase a weapon with not enough points, the exact number of points and more points than needed	Nothing will happen when player attempts to purchase weapon with not enough points but when player has the exact number or more they will successfully purchase the weapon	
1b	Points are deducted upon purchase	Purchase a weapon	Player's points are deducted by the price of the weapon	
1c	Current weapon is replaced by purchased weapon	Buy new weapon	Current weapon is replaced by the new purchased weapon and player is given 75% of max ammo for the weapon	
1d	Player's ammo is replenished after repurchasing weapon	Buy the same weapon after using some ammo and reloading	Ammo player is holding is set to max for the weapon	
2	Player picks up ammo box after touching it	Walk into ammo box	Ammo player is holding for currently equipped weapon is replenished by a random amount	

3	Verify items are scattered appropriately	Run script to scatter items across the map	Items will be spread out and placed in suitable locations the player can access	
4	Player can retrieve weapon after it finishes upgrading	Upgrade weapon	When the player's weapon finishes upgrading it will reappear floating for the player to pick up	
5a	Weapon disappears if player leaves weapon for too long after upgrading	Upgrade weapon and leave it	Player's weapon will disappear after left for a set interval	
5b	Verify time to upgrade and points required increase each upgrade	Upgrade the same weapon repeatedly	Points required to purchase upgrade and time to upgrade increase with each upgrade	
5c	Max level player can upgrade a weapon to increases after a set number of waves	Try and upgrade a weapon repeatedly on a low and high wave	Player will be able to upgrade the weapon more during the high wave	
6a	Ensure magical forge only appears after the 10^{th} wave	Check if magical forge is present before and after the 10^{th} wave	Magical forge will appear after the 10^{th} wave ends	
6b	Magical forge reappears elsewhere after a set number of uses	Upgrade weapons repeatedly	Magical forge will appear elsewhere on the map after the player has upgraded weapons a set number of times	
6c	Verify magical forge is easily accessible	Spawn magical forge	Magical forge will appear in places easy for the player to find and interact with	

Stage 8 (Bosses – 1 week):

1. Create/find models for a few bosses
2. Create unique attacks for each boss providing them all with an animation
3. Create visual for boss bar
4. Create script to spawn boss at end of every 10^{th} wave and replenish all of player's health after it's defeated

Test No.	Description	Test Data	Expected Result	Actual Result
1	Bosses' attacks take into account the surroundings	Surround player with obstacles and spawn boss	Boss will not attempt to attack player through walls and will adjust attacks	

			based on obstacles surrounding them and te player	
2	Boss bar behaves correctly	Spawn boss	While alive a giant healthbar will appear on the player's screen with the boss's name above it and will fill and deplete like the player's health bar	
3a	Verify boss spawn at end of every 10th wave	Check if boss is present for first 100 waves	Boss will spawn once the last enemy has been killed every 10th wave	
3b	Player's health is replenished to max after the boss is defeated	Kill boss while health is below max	Player's health will be set to the maximum	

Stage 9 (Magic – 1 week):

1. Create some spells for the player to use
2. Create/find some visuals and animations
3. Create/find model for scroll chest
4. Refactor code to scatter items across the map to include scroll chests
5. Create screen that pops up when player tries to open scroll chest displaying the puzzle to open the chest with a text field or multiple choice for the user to answer the puzzle
6. Create script to generate unique puzzles for all the scroll chests
7. Give player random scroll they haven't found after completing puzzle

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Player can only jump or walk whilst casting	Try and sprint, use potions and or weapons whilst casting	Nothing will happen when the player attempts to sprint but upon switching to their weapon or using a potion the casting will cancel	
1b	Player can only hold 2 spells	Pick up a spell whilst holding 2 spells	1 spell the player is carrying will be dropped	
1c	Player cannot recast spell after limited number of uses has expired	Attempt to use spell with set number of uses repeatedly	Scroll will become unusable if it has a set number of uses each wave and disappear if it has a set number of uses throughout the game	

2	Spell casting animation stops when spell is cancelled	Cast a spell and cancel it before it finishes	Animation will stop playing when casting is cancelled	
3	Verify spell chests are easily accessible	Run script to scatter items across the map	Spell chests will be scattered in places easy for the player to find and interact with	
4a	Text field only allows user to type numbers and or letters	Try and type a letter, number and symbol into the text field	Letter will be accepted on text field; number may be accepted if the puzzle requires it, and symbol will not be accepted	
4b	Player can only select one option during multiple choice	Click a second option after already selecting one	Selected option will be changed to the second option that was clicked	
4c	Screen no longer appears after chest has been opened	Attempt to interact with chest after opening it	Nothing will happen	
5a	Puzzles are solvable	Repeatedly generate puzzles	All problems will have a solution	
5b	Puzzles can be solved without the need of a calculator	Repeatedly generate puzzles	All problems will be solvable by hand/mentally	
6	Player does not receive the same scroll	Open scroll chests repeatedly	Player will receive a unique speel each time	

Stage 10 (Misc – 1 week):

1. Find/create model for mystery box
2. Add script to mystery box so player can redeem points for a random weapon
3. Add script so mystery box starts at random location and reappears after a set number of uses like the magical forge
4. Add beacon in the sky to mark the location of the mystery box and magical forge
5. Add talismans into game with various effects
6. Add script to bosses so they drop a random talisman after they are defeated
7. Add any missing animations or sound effects
8. Add random facts for the loading screen to cycle through
9. Add music for the game and custom music for the boss fights

Test No.	Description	Test Data	Expected Result	Actual Result
1	Player cannot re-redeem points for mystery box immediately after redeeming	Redeem points for mystery box repeatedly	Player will only be able to use mystery box again after random weapon appears from opening it	

2a	Mystery box reappears elsewhere after a set number of uses	Redeem points for mystery box repeatedly	Mystery box will appear elsewhere on the map after the player has opened it a set number of times	
2b	Verify mystery is easily accessible	Spawn mystery box	Mystery box will appear in places easy for the player to find and interact with	
3	Beacon does not appear inside buildings	Spawn mystery box and magical forge	Coloured beacon will appear directly above mystery box and magical forge that only appears outside e.g. on top of roof above mystery box/ magical forge	
4	Player cannot equip a second talisman	Try and equip talisman whilst one is already equipped	Equipped talisman will be dropped, and the other talisman will be equipped	
5	Talisman drops when boss is defeated	Kill boss	A random talisman will spawn where the boss was killed	
6	Test all added sounds and animations play correctly	Test abilities with added sounds and animations	All sounds and animations will trigger when an ability starts and stop when an ability cancels	
7	Fact displayed changes when background image changes	Initiate loading screen	Loading screen will cycle through a new fact and background image at set intervals	
8a	Boss music starts when boss spawns and ends when boss is killed	Spawn boss and kill it	Boss music will begin when it spawns and then transition to the normal music when it dies	
8b	More dramatic music plays when the player is low on health	Let player take damage until their health is low	Once the player's health is below a set threshold more dramatic music will begin playing and will transition back to the normal music when the player's health is above the threshold	
8c	More dramatic music does not play during boss fights	Let player take damage until their health is low while boss is alive	Boss music will continue playing	

Stage 11 (Map Generation – 2 weeks):

1. Create a world generator so a similar map will generate each time, but it is unique
2. Create a bigger variety of map types which they player can select

Test No.	Description	Test Data	Expected Result	Actual Result

Stage 12 (Extra Enemy Stuff – 2 weeks):

1. Create different stage for the bosses
2. Create more advanced ai for zombies so each behaves somewhat uniquely

Test No.	Description	Test Data	Expected Result	Actual Result

Beta Testing:

For my beta testing, I will give a variety of users an executable to run the game so that can be played on a variety of machine specifications. I gave the instructions on how to play the game along with a survey to fill in at the end. The survey will give my insights into the user needs and experience which will help me assess the trade-offs I made between usability and functionality of the game.

Survey Results:

1. How many bugs did you encounter?

More Details

0	5
1	5
2-3	4
4-5	0
6+	0

2. Describe the bug(s) in detail

9 Responses

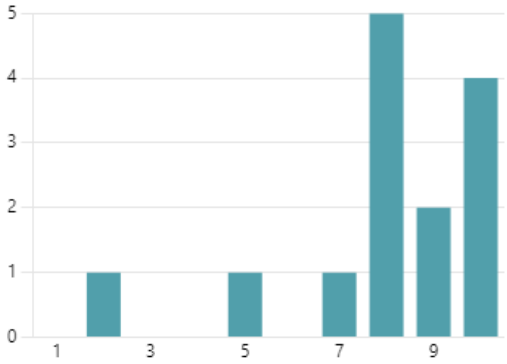
ID ↑	Name	Responses
1	anonymous	got stuck when walking
2	anonymous	Not sure if it's a bug, but the character model kind of blocks the camera when jumping, and looks quite odd if you look down.
3	anonymous	The equip was quite inconsistent, and I would often just be running into a weapon spamming e until I equipped it. The melee attack often wouldn't connect with an enemy when I was right in front of and colliding with them.
4	anonymous	when i spawned i couldnt move until i pressed esc
5	anonymous	Could not pick up the items Had to press esc to unfreeze when re-entering the game after returning to main menu
6	anonymous	THe gun barely worked I could fall of the edge of the map
7	anonymous	Falling off map Zombies stuttering a little bit
8	anonymous	escape bug
9	anonymous	After pressing return to main menu then start game again, game appears paused /frozen without any UI. Pressing escape fixes it again

Overall, it seems the functionality of the game worked for all users. However, some low and medium priority bugs were found and a couple of enhancements. This could have been improved I had done more through testing during my development.

3. How user-friendly was the UI (user interface)?

More Details Insights

8.00
Average Rating



An average rating of 8 clearly shows that the usability of the UI was easy to navigate and appealing to most users. However in hindsight I could have incorporated key elements available in most games like a HUD, settings menu, maps, etc… instead of solely focusing on combat.

4. You would describe the combat as…

More Details

Simple and straightforward to u...	11
Statisfying	5
Well-balanced (difficulty)	4

(Note: 1st option is Simple and straightforward to understand and there were 14 responses)

Almost all beta testers found the combat to be simple to use however only 1/3 of users found it to be satisfying or well-balanced. This is primarily due to the lack of visual feedback. Due to lack of time, I had intentionally made spawning of zombies and combat, simpler however that contributed to a less challenging game.

5. Do you typically play games on PC?

More Details

Yes	9
No	5

My user demographic was targeted was pc users and my beta testers represented a good sample size of both targeted and non-targeted users.

6. How would you rate the game's controls?

More Details

Exceptional	3
Good	3
Average	3
Poor	0
Very poor	0

(Note: This is the response from the 9 PC users)

As expected, the PC users found the controls reasonable and easy to use.

6. How would you rate the game's controls?

More Details

Exceptional	1
Good	2
Average	1
Poor	1
Very poor	0

(Note: This is the response from the 5 non-PC users)

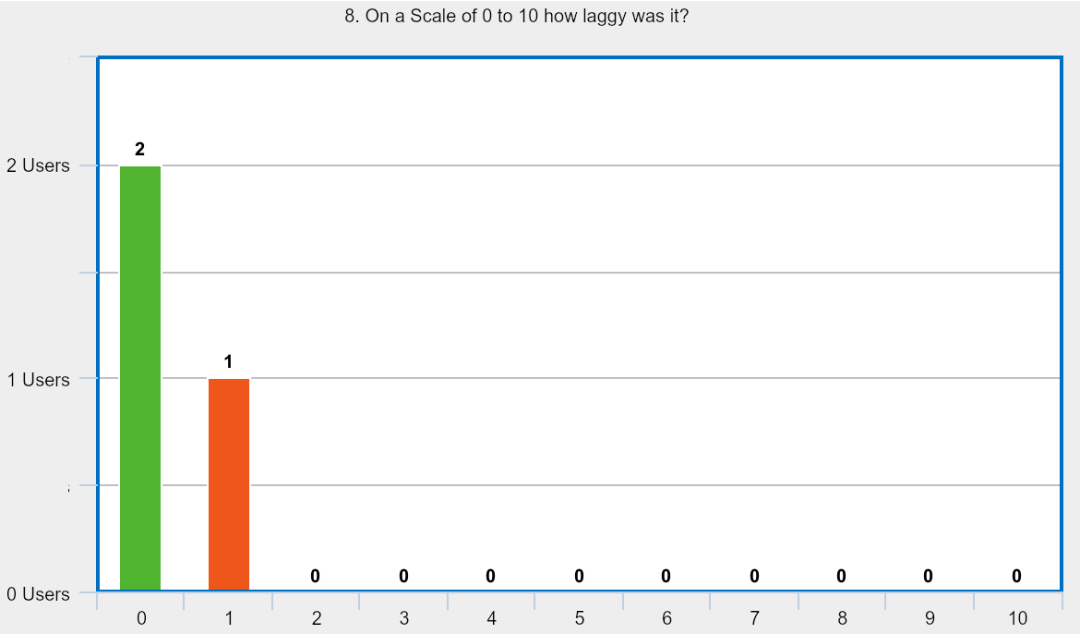
Even though one of the users found the controls poor, the range of responses indicates that the controls are effective even for those that don't regularly play PC games. To improve this I could have added a simple visual to display some essential controls so that users are reminded of what buttons to press.

7. What spec is your computer?

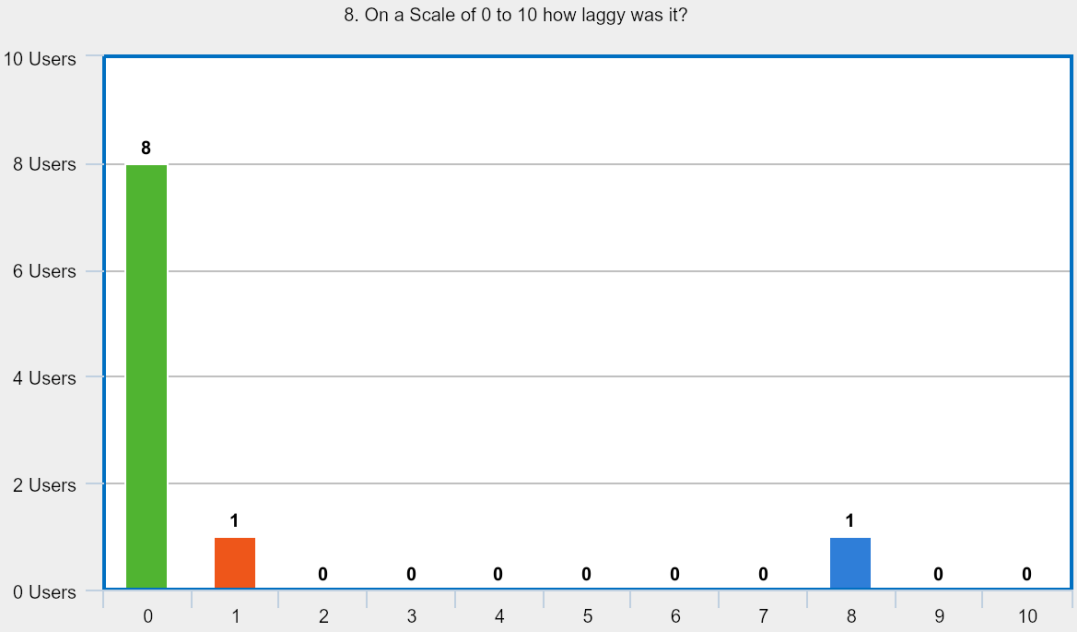
More Details Insights

High-end	3
Medium	10
Low-end	1

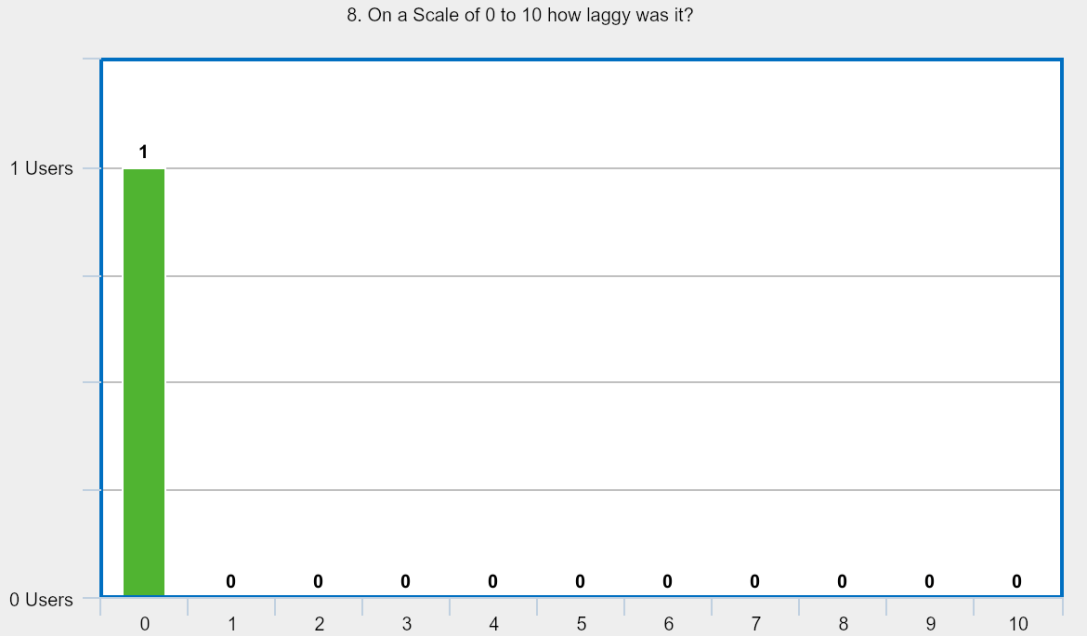
As shown most users were in the medium spec range and there wasn't much variety between the computer specs which skews the testing and does not accurately represent all user types. The graph below indicates that for all users (with 1 exception) the game performed smoothly with little to no lag.



(Note: This response is from the 3 High-end users)



(Note: This response is from the 10 medium users)

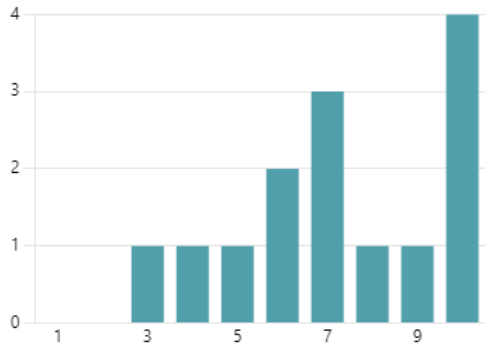


(Note: This response is from the 1 Low-end user)

9. How satisfied were you overall?

More Details Insights

7.29
Average Rating

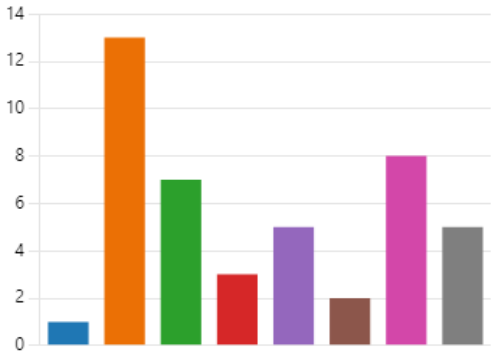


This a good score despite the game lacking many core features. However, the reviews are very mixed which indicates the games need more polishing to appeal to a wider audience.

10. How could the game have been improved? Tick all that apply

More Details

Nothing	1
A HUD for the player (healthbar,...	13
Tutorial for the controls	7
Settings Menu	3
Better Map	5
Better textures, models and ani...	2
Improved enemies and combat	8
Other	5



As shown in the graph above the player HUD is a critical which was identified as missing core need by most users. The responsiveness could be improved through visual feedback as indicated by 2 of the written comments below. To improve complexity and engagement, a larger variety could be to the enemies and combat feature. I prioritised the general mechanics of the combat over visuals which in hindsight is a good decision as reflected by the overall satisfaction rating however I could have provided some basic visuals as almost all users expected a HUD as a mandatory feature in the game.

11. If you selected "other" what features do you think could have improved the game?

5 Responses

ID ↑	Name	Responses
1	anonymous	Would be nice if the enemies actually dealt damage because otherwise it does feel a bit low-stakes.
2	anonymous	I think it would be really good if the player got more feedback from their actions. Although things were happening it often felt disconnected or unresponsive from my inputs. When clicking a lot of times with the gun, the enemy only dies every so often with no visual feedback. When attack I can't see the range of the attack as my arms fly above the camera. I think some kind of particles or graphics to show what my inputs are doing a little more clearly would help. Also because you might need this for the write up ;) It would be good if the player took damage to increase the stakes of the game.
3	anonymous	Kill counter
4	anonymous	Special and more sound effects
5	anonymous	make it obvious that bullets are shot / enemies are hit, enemy healthbars, enemy death animation

Evaluation:

1.Player Movement:

Description: WASD movement for the player and the ability to jump, sprint and dash

Importance: Essential

Score: 8/10 – Partially met

Location in project: Stage 1

Comments: For the player movement I managed to implement the most essential features: the WASD movement, sprint and jump ability. The WASD movement as intended moves the player forwards, left, backwards and right using the W, A, S and D keys respectively and normalises the speed of the player when moving diagonally. The sprint ability increases the players speed upon holding the ctrl key and can be used indefinitely. The jump ability allows the player to jump when grounded by pressing the spacebar and brings the player back down after doing so. Although, neither a roll nor dash ability have been implemented. They however were of minor importance and as shown in the beta testing results for Q6 users were generally satisfied with the controls and on Q11 there was no requests for improvements in the player's movement or any movement abilities.

Improvements: Only improvements could have been to add the choice between a dash or roll ability for the player. This would make it easier for the player to doge enemy attacks and manoeuvre around them.

2.Player Health:

Description: health mechanics for the player

Importance: Essential

Score: 0/10 – not met

Location in project: Stage 3

Comments: When trying to implement this feature I tried detecting collision between the player and objects in the "enemy" layer. Although even with the enemy clearly collided with the player the game wasn't detecting any collisions this left me with no clear way to damage the player and so due to a lack of time, I did not pursue any further methods of adding this feature.

3.Player Items:

Description: A variety of tools the player can use to survive and fight against enemies

Importance: Essential, with some desirables

Score: 7.5/10 – Partially met

Location in project: Stage 2

Comments: For the player items I managed to implement the core functionality for the player's melee and ranged Weapons. Melee weapons can either be one or two handed, one handed weapons can be dual wielded and two handed weapons can only be held one at a time. The player can initiate an attack by pressing the LMB which plays the equipped weapon(s) attack animation and fires a capsule cast to detect if any enemies were hit. Initially I had planned to add a throw and block ability, but these were not implemented. Ranged weapons are two handed and are comprised of a shoot and reload ability. The shoot ability utilises a raycast to detect enemies hit although during the beta testing I noticed that gun was inconsistent at hitting a target as pointed out by one of the beta testers in Q2. The reload ability "fills" the player's ranged weapon with the ammo the player currently holds. However, I did not manage to find a working method to delay the reload. This was a critical problem as without it makes the whole reload ability redundant (due to the reload now being instant). On reflection I should have addressed this problem before releasing the game for beta testing. The other important feedback from a number of beta testers was on the lack of visual response for player attacks with both melee and ranged weapons. Additional features like wonder weapons, magic, upgrades, talismans and flask were not implemented due to lack and dependency to other features that were not yet built e.g. flasks were useless when player health is stagnant and upgrades are not necessary when the enemies strength remains the same.

Improvements:

- I could have refined the hit detection for melee weapons – I could do this by instead directing the player's weapon towards the crosshair and check the collision of the object as opposed to using a capsule cast.
- Improve consistency of ranged weapon's shots – This could be done by using a capsule cast to give it's shots volume so it hits the target more frequently.
- Visual feedback – Add a hit indicator when the player successfully hits a target and more specifically for a ranged weapon a muzzle flash and decal on what was hit.

4.Player HUD:

Description: Status bar for player through visual cues

Importance: Essential

Score: 0/10 – Not met

Location in project: N/A

Comments: Again, due to lack of time I did not create a player HUD as I prioritised the functionality of the game over the visuals. However, in Q10 of my beta testing survey more than 90% of users felt that a player HUD would have improved the game. So, in retrospect it may have been better to pay more attention to visuals even if only adding very basic ones.

5.Player Perspective:

Description: Player will be in first person perspective

Importance: Essential

Score: 10/10 – Fully met

Location in project: Stage 1

Comments: When entering the game, the camera will follow the player displaying their POV. Additionally, when moving their mouse, it will rotate the player and camera in unison. The player perspective functions as expected however a beat tester reported a scenario where the camera was blocked when the player jumped looking downwards. But this issue is of minor importance and can be ignored.

Improvements: Adress the body blocking the camera when jumping. This could be resolved by exploring camera movement speed as I believe the issue is caused due to the player's jump speed exceeding that of the camera.

6.Enemies:

Description: NPC's that chase the player and start attacking them when within range

Importance: Essential, with some desirables

Score: 8/10 – Partially met

Location in project: Stage 3

Comments: The enemies spawn periodically from a set location in the game as due to time constraints I did not make a script to scatter them across the map. Once spawned enemies pathfind towards the player. When in range they begin attacking the player however as mentioned prior I could not find a suitable method to cause damage to the player. When an enemy has been hit by the player's weapon it will be dealt an appropriate amount damage depending on the weapon used. Overall, the most essential features were addressed so I thought that there was no need to build the desirable features. Although over 50% of the users felt that the game could have benefited from improved enemies and combat. In hindsight as most games offer this as baseline, I should have considered this as an essential part of my game.

Improvements:

- Spawning enemies at random locations – This would help spread out the enemies and keeps the user in a constant state of being hunted
- Different types of enemies - I would do this by creating attributes for speed, damage, health and possibly other characteristics. Furthermore, I would then randomise these attributes among enemies to enhance the variation providing the user with a more engaging experience.

7.Bosses:

Description: More challenging enemies that spawn every 10th wave

Importance: Essential, with one desirable

Score: 0/10 – Not met

Location in project: N/A

Comments: This feature was not delivered however as no users were requesting for a boss to be added. Therefore, this feature may have been non-essential.

8.Main Menu:

Description: A menu screen with a title, start game button, settings menu, loadout menu, exit game button overlayed on a background image.

Importance: Essential with one desirable

Score: 7/10 – Partially met

Location in project: Stage 6

Comments: My main menu features a start and exit game button, title and a background graphic of the gameplay. This was clearly satisfactory for most users as reflected in Q3 where the user-friendliness obtained an average score of 8. Additionally, I had categorised settings menu as a core feature of the main menu but only a minor percentage of users felt that would have improved the game. Consequently, it implies that the absence of a settings menu in the game did not make an impact on the player's experience. Lastly the loadout menu had dependencies on other features which are yet to be implemented and it was considered non- essential so again it has little impact on the player's experience.

Improvements:

- Settings Menu – This could be done by adding a button to the main menu that directs the player to a new page on click to customise the games as per their preferences.

9.Pause Menu:

Description: A menu screen that is triggered by the esc key whilst the player is in the game

Importance: Essential

Score: 8/10 – Partially met

Location in project: Stage 6

Comments: My pause menu triggers upon the player the esc whilst in the game. Once activate it pauses the game, puts it out of focus and overlays the menu buttons. The user can the choose to resume, exit to main menu or exit the game. This functionality was fully met however there was a bug whereby when returning to the main menu and playing the game again, the game stays in the paused state until the user presses esc key. In Q2 of the beta testing survey this was the most prominent error observed. This could have been resolved with more thorough testing during development.

Improvements: Fix the esc bug – This could be resolved by adding a step to explicitly "resume" the game when returning to main menu.

10.Start of Game:

Description: The player has all the items they selected and spawn in the appropriate location when starting the game.

Importance: Essential

Score: 0/10 – Not met

Location in project: N/A

Comments: Due to the lack of proper map and loadout for the player not being fully implemented this feature could not be completed.

11.End of Game:

Description: A graphic the displays the users stats upon death and a return to main menu.

Importance: Essential

Score: 0/10 – Not met

Location in project: N/A

Comments: Due to the player being unable to take damage this feature could not be developed as it was impossible for the player to die.

12.Sound:

Description: Sound effects and background music

Importance: Essential

Score: 0/10 – Not met

Location in project: N/A

Comments: Again, due to time constraints this feature was not added as I considered the game's functionality to be of more importance. Although only 1 beta tester requested for sound effects in Q11, so my judgement was correct.

13.Points:

Description: A currency the player gains by defeating enemies

Importance: Essential

Score: 0/10 – Not met

Location in project: N/A

Comments: This feature is an in-game currency so without anything to spend it on it would add little value to the player experience.

14.Map:

Description: An interactive playable area filed with weapons and chests for the player to obtain items

Importance: Essential

Score: 1/10 – Not met

Location in project: N/A

Comments: A very basic terrain was added to provide baseline functionality in the game. However, it seems a 1/3 of users felt the map should have been better. So, in retrospect I could have further developed the map to provide a richer experience to the user.

Improvements:

- Scattering purchasable weapons – This provides the player with a way to obtain a variety of weapons and using points. Additionally, it helps the user to resupply their ammo for ranged weapons
- Mystery box – Gives user another way to spend points and earn surprise weapons
- Beacons – provides the user a way to find the mystery box

Code:

Player.cs:

```
using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UIElements;


public class Player : MonoBehaviour {


  [SerializeField] private CharacterController characterController;

  //movement

  [SerializeField] private float movSpeed = 7f;
```

```
    private float sprintMultiplier;
    //gravity
    [SerializeField] private float gravity = -9.8f;
    [SerializeField] private Transform groundCheck;
    private float groundDistance = 0.4f;
    [SerializeField] private LayerMask groundLayer;
    [SerializeField] private float jumpHeight=10f;
    Vector3 v;
    bool isGrounded;
    //interaction
    [SerializeField] private LayerMask interactLayer;
    //picking up weapon
    [SerializeField] WeaponList weaponList;
    private Transform weaponRef;
    [SerializeField] private Transform playerRightHand, playerLeftHand, floatingGun, playerCamera;
    private MeleeWeaponScriptObj rightHandSO, leftHandSO;
    private RangedWeaponSO rangedWeaponSO;
    private bool meleeEquipped;
    //animation state bools
    private bool isWalking, isSprinting, isJumping, isAttacking, rightHandWeapon, leftHandWeapon, rangedWeapon,
isOneHanded;
    [SerializeField] private Animator animator;
    //shooting
    [SerializeField] private LayerMask hittableLayer;
    float counter=0;
    int fps = 60;

    // Start is called before the first frame update
    void Start() {
        QualitySettings.vSyncCount = 0;
        Application.targetFrameRate = fps;
    }
```

```
// Update is called once per frame
void Update() {
    HandleMovement();
    if(Input.GetKey(KeyCode.E)) {
        HandleInteractions();
    }
    HandleControls();
}
private void HandleMovement() {
    //reseting all animation states
    isWalking = false;
    isSprinting = false;
    isJumping = false;
    //creates sphere at bottom of player to check if they have reached the ground
    isGrounded = Physics.CheckSphere(groundCheck.position,groundDistance,groundLayer);
    //sets velocity to -2 when player reaches the ground
    if(isGrounded && v.y<0) {
        v.y = -2f;
    }
    //reseting sprint multiplier
    sprintMultiplier = 1f;
    //gets movement vector for direction player is moving
    Vector3 movDir = getMovDir();
    //making isWalking true when player is moving
    if(movDir != Vector3.zero) {
        isWalking = true;
    }
    //adds sprint multiplier and activates isSprinting if sprintkey pressed and attack anim isn't playing
    if(Input.GetKey(KeyCode.LeftShift)&&!AttackAnimPlaying()) {
        sprintMultiplier = 1.5f;
        isSprinting = true;
```

```
    }
    //provides charcter controller with movement vector for player
    characterController.Move(movDir.normalized*movSpeed*sprintMultiplier*Time.deltaTime);

    //executes jump ability if player is on the ground
    if(isGrounded && Input.GetButtonDown("Jump")) {
      //sets velocity to initial velocity required to reach desired jump height
      v.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
      isJumping = true;
    }

    //adds gravity to player
    v.y += gravity * Time.deltaTime;
    characterController.Move(v * Time.deltaTime);
  }

  private void HandleInteractions() {
    float playerHeight = 3f;
    //creating 3D vector for direction player is looking/moving
    Vector3 movDir = getMovDir();

    weaponRef = null;
    //checking if object in the interact layer is within 2 units of the player in the direction they're looking
    if(Physics.CapsuleCast(transform.position + Vector3.up * playerHeight,transform.position,.5f,movDir,out RaycastHit
hitInfo,2f,interactLayer)) {
      weaponRef = hitInfo.transform;
      //checking if weapon is melee through "Melee" tag
      if(weaponRef.CompareTag("Melee")) {
        //checks if player has no weapon in right hand
        if(!rightHandWeapon) {
          //finding scriptable object for weapon
          rightHandSO = weaponList.FindMeleeSO(weaponRef.name);
```

```
        isOneHanded = rightHandSO.IsOneHanded();
        IntitialiseWeapon(weaponRef,playerRightHand);
        rightHandWeapon = true;
        Debug.Log("Right Weapon in hand = " + weaponRef.name);

        //sets leftHand bool to true if weapon is 2 handed
        if(!isOneHanded) {
            leftHandWeapon = true;
            Debug.Log("Left Weapon in hand = " + weaponRef.name);
        } if(rangedWeapon) {
            ToggleWeapon(floatingGun,0,false);
        }
        //checks if player only has weapon in right hand and picked object isn't the weapon in their right hand
    } else if(rightHandWeapon && !leftHandWeapon && weaponRef.parent != playerRightHand) {
        leftHandSO = weaponList.FindMeleeSO(weaponRef.name);
        //makes sure weapon isn't two handed
        if (leftHandSO.IsOneHanded()) {
            IntitialiseWeapon(weaponRef,playerLeftHand);
            leftHandWeapon = true;
            Debug.Log("Left Weapon in hand = " + weaponRef.name);
        } else {
            Debug.Log("Hands are full");
        }if(rangedWeapon) {
            ToggleWeapon(floatingGun,0,false);
            ToggleWeapon(playerRightHand,5,true);
        }
    }
} if(weaponRef.CompareTag("Ranged")) {
    if(!rangedWeapon) {
        Debug.Log("Picked up ranged Weapon");
        rangedWeaponSO = weaponList.FindRangedSO(weaponRef.name);
        rangedWeaponSO.IntialiseAmmo();
```

```
            IntitialiseWeapon(weaponRef,floatingGun,Quaternion.identity,Vector3.zero);
            rangedWeapon = true;
            if(IsMeleeEquipped()) {
               if(rightHandWeapon) {
                  ToggleWeapon(playerRightHand,5,false);
               }
               if(leftHandWeapon && isOneHanded) {
                  ToggleWeapon(playerLeftHand,5,false);
               }
            }
         }
      }
   }
}

private void HandleControls() {
   //resets isAttacking each update so anim doesn't play continuous
   isAttacking = false;
   //drops the weapon currently in the player's hand
   if(!AttackAnimPlaying()) {
      if(Input.GetKeyDown(KeyCode.Q)) {
         if(IsMeleeEquipped()) {
            if(leftHandWeapon) {
               leftHandWeapon = false;
               if(isOneHanded) {
                  DropWeapon(playerLeftHand,5);
                  Debug.Log("Left Weapon dropped");
               } else {
                  rightHandWeapon = false;
                  DropWeapon(playerRightHand,5);
                  Debug.Log("Right Weapon dropped");
               }
```

```
            } else if(rightHandWeapon) {
                rightHandWeapon = false;
                DropWeapon(playerRightHand,5);
                Debug.Log("Right Weapon dropped");
            }
        } else {
            if(rangedWeapon) {
                rangedWeapon = false;
                DropWeapon(floatingGun,0);
                Debug.Log("Gun dropped");
            }
        }
    } else if(Input.GetKeyDown(KeyCode.Mouse0) && (rightHandWeapon || rangedWeapon)) {
        isAttacking = true;
        //Debug.Log("Attack!");
        if(!IsMeleeEquipped()&&rangedWeapon&&counter<=0) {
            Debug.Log("Shot fired");
            rangedWeaponSO.Shoot(floatingGun.position,playerCamera.forward,hittableLayer);
            counter = rangedWeaponSO.GetShootDelay(); ;
        } else if (IsMeleeEquipped()) {
            rightHandSO.Attack(playerRightHand.GetChild(5).position,playerCamera.forward,hittableLayer);
            if (leftHandSO!=null) {
                leftHandSO.Attack(playerLeftHand.GetChild(5).position,playerCamera.forward,hittableLayer);
            }
        }
    } else if((Input.GetKeyDown(KeyCode.Alpha1) && !IsMeleeEquipped() && rightHandWeapon) ||
(Input.GetKeyDown(KeyCode.Alpha2)) && (!rightHandWeapon || IsMeleeEquipped())) {
        if(rightHandWeapon) {
            ToggleWeapon(playerRightHand,5);
        }
        if(leftHandWeapon && isOneHanded) {
            ToggleWeapon(playerLeftHand,5);
```

```
            }
            if(rangedWeapon) {
                ToggleWeapon(floatingGun,0);
            }
        } else if(Input.GetKeyDown(KeyCode.R)&&rangedWeapon&&!IsMeleeEquipped()) {
            rangedWeaponSO.Reload();
        }
    } if(counter > 0) {
        counter -= Time.deltaTime;
    }

}
void OnCollisionStay(Collision collision) {
    Debug.Log(collision.gameObject.layer.ToString());
}

//animation bool return functions
public bool IsWalking { get { return isWalking; } }
public bool IsSprinting { get { return isSprinting; } }
public bool IsJumping { get { return isJumping; } }
public bool IsAttacking { get { return isAttacking; } }
public bool RightHandWeapon { get { return rightHandWeapon; } }
public bool LeftHandWeapon { get { return leftHandWeapon; } }
public bool IsMeleeEquipped() {
    if(rightHandWeapon) {
        return playerRightHand.GetChild(5).gameObject.activeSelf;
    } else {
        return false;
    }
}
public bool IsOneHanded { get { return isOneHanded; } }
//general functions
```

```
    private void IntitialiseWeapon(Transform weapon, Transform parent, Quaternion rotation, Vector3 pos) {
        weapon.GetComponent<Rigidbody>().isKinematic = true;
        weapon.parent = parent;
        weapon.localRotation = rotation;
        weapon.localPosition = pos;
    }
    private void IntitialiseWeapon(Transform weapon, Transform parent) {
        IntitialiseWeapon(weapon, parent,Quaternion.Euler(new Vector3(15f,-15f)),new Vector3(-0.0696f,-0.0002f,-
0.0032f));
    }
    private void DropWeapon(Transform parent,int index) {
        Transform weapon = parent.GetChild(index);
        weapon.GetComponent<Rigidbody>().isKinematic = false;
        weapon.parent = null;
    }
    private void ToggleWeapon(Transform parent,int index, bool state) {
        GameObject weapon = parent.GetChild(index).gameObject;
        weapon.SetActive(state);
    }
    private void ToggleWeapon(Transform parent,int index) {
        GameObject weapon = parent.GetChild(index).gameObject;
        weapon.SetActive(!weapon.activeSelf);
    }
    //returns movement vector relative to the player's position based on their input
    private Vector3 getMovDir() {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");
        return (transform.right * x + transform.forward * z);
    }

    private bool AttackAnimPlaying() {
```

```
      if(animator.GetCurrentAnimatorStateInfo(1).IsName("Melee Attack
A1")||animator.GetCurrentAnimatorStateInfo(2).IsName("Melee Attack A1")) {
         return true;
      } else {
         return false;
      }
   }
}
```

MouseLook.cs:

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

public class MouseLook : MonoBehaviour {
   [SerializeField] private float mouseSensitivity = 100f;
   [SerializeField] private Transform player;
   [SerializeField] private float rotateSpeed = 10f;
    private float xRotation = 0f;
   void Start() {
      Cursor.lockState = CursorLockMode.Locked;
   }
   void Update() {
      float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
      float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

      xRotation -= mouseY;
      xRotation = Mathf.Clamp(xRotation,-90f,90f);

      transform.localRotation = Quaternion.Lerp(transform.localRotation,Quaternion.Euler(xRotation,0f,0f),rotateSpeed
* Time.deltaTime);
      player.Rotate(Vector3.up * mouseX);
```

```
    }
}
```

PlayerAnimator.cs:

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

public class PlayerAnimator : MonoBehaviour {
    [SerializeField] private Player player;
    private Animator animator;
    private void Awake() {
        animator = GetComponent<Animator>();
    }
    private void Update() {
        animator.SetBool("isWalking",player.IsWalking);
        animator.SetBool("isSprinting",player.IsSprinting);
        animator.SetBool("isJumping",player.IsJumping);
        animator.SetBool("isAttacking",player.IsAttacking);
        animator.SetBool("rightHandWeapon",player.RightHandWeapon);
        animator.SetBool("leftHandWeapon",player.LeftHandWeapon);
        animator.SetBool("isMeleeEquipped",player.IsMeleeEquipped());
        animator.SetBool("isOneHanded",player.IsOneHanded);
    }
}
```

MeleeWeaponScriptObj.cs:

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

[CreateAssetMenu()]
```

```
public class MeleeWeaponScriptObj:ScriptableObject {
    [SerializeField] private Transform prefab;
    [SerializeField] private string objectName;
    [SerializeField] private bool isOneHanded;
    [SerializeField] private int damage;

    public Transform GetPrefab() {
        return prefab;
    }
    public string GetObjectName() {
        return objectName;
    }
    public bool IsOneHanded() {
        return isOneHanded;
    }
    public void Attack(Vector3 pos,Vector3 dirVector,LayerMask hittableLayerMask) {
        if(Physics.CapsuleCast(pos,pos,0.5f,dirVector,out RaycastHit hitInfo,2,hittableLayerMask)) {
            if(hitInfo.transform.gameObject.layer == 8) {
                hitInfo.transform.root.gameObject.GetComponent<Zombie>().Damage(damage);
                Debug.Log("Enenemy Hit!");
            }
        }
    }

}
```

RangedWeaponSO.cs:

```
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using Unity.VisualScripting;
using UnityEngine;
```

```
[CreateAssetMenu()]
public class RangedWeaponSO:ScriptableObject {
    [SerializeField] private Transform prefab;
    [SerializeField] private string objectName;
    private int ammo,ammoInGun;
    [SerializeField] private int damage, startingAmmo, maxAmmo, maxAmmoInGun, maxShootingRange;
    [SerializeField] private float shootDelay, reloadTime;

    //get functions
    public Transform GetPrefab() {
        return prefab;
    }
    public string GetObjectName() {
        return objectName;
    }
    public int GetMaxAmmoInGun() {
        return maxAmmoInGun;
    }
    public float GetShootDelay() {
        return shootDelay;
    }
    //ammo functions
    public void IntialiseAmmo() {
        ammo = startingAmmo;
        ammoInGun = maxAmmoInGun;
    }
    public void Shoot(Vector3 pos, Vector3 dirVector, LayerMask hittableLayerMask) {
        if(ammoInGun == 0) {
            Reload();
        } else {
            ammoInGun -= 1;
```

```
        if(Physics.Raycast(pos,dirVector,out RaycastHit hitInfo,maxShootingRange,hittableLayerMask)) {
            //checking if object hit is in the enemy layer
            //Debug.Log(hitInfo.transform.gameObject.layer.ToString());
            if(hitInfo.transform.gameObject.layer==8) {
                Debug.Log("Enemy hit!");
                //hitInfo.transform.root.gameObject.SetActive(false);
                hitInfo.transform.root.gameObject.GetComponent<Zombie>().Damage(damage);
            } else {
                Debug.Log("Something got hit");
            }
        } else {
            Debug.Log("You can't shoot");
        }
    }
}
public void Reload() {
    float i = 0;
    Debug.Log("Reloading...");
    //reload delay
    //yield return new WaitForSeconds(reloadTime);
    Debug.Log("Before Reloading - Ammo: " + ammo + " Ammo in gun: " + ammoInGun);
    //checking if gun is already fully loaded and player has ammo to reload
    if(ammoInGun != maxAmmoInGun && ammo != 0) {
        //subtracting the amount that needs to be loaded from ammo
        ammo -= (maxAmmoInGun - ammoInGun);
        //checking if gun can be fully loaded
        if(ammo >= 0) {
            ammoInGun = maxAmmoInGun;
        }
        //if not partially loading gun with what is remaining
        else {
            ammoInGun = maxAmmoInGun + ammo;
```

```
            ammo = 0;
        }
    }
    Debug.Log("After Reloading - Ammo: " + ammo + " Ammo in gun: " + ammoInGun);
  }

}
```

WeaponList.cs:

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

public class WeaponList : MonoBehaviour {
  List<MeleeWeaponScriptObj> meleeWeapons = new List<MeleeWeaponScriptObj>();
  [SerializeField] private MeleeWeaponScriptObj baseballBat, axe;

  List<RangedWeaponSO> rangedWeapons = new List<RangedWeaponSO>();
  [SerializeField] private RangedWeaponSO revolver;

  void Start() {
    //adding scribtable objects to list for melee weapon
    meleeWeapons.Add(baseballBat);
    meleeWeapons.Add(axe);
    //same for ranged weapons
    rangedWeapons.Add(revolver);
  }
  //finds scriptable object with weapon's name
  public MeleeWeaponScriptObj FindMeleeSO(string objectName) {
    for(int i = 0; i < meleeWeapons.Count; i++) {
      if(string.Equals(meleeWeapons[i].GetObjectName(),objectName)) {
        Debug.Log(meleeWeapons[i].GetObjectName());
```

```
            return meleeWeapons[i];
        }
    }
    return null;
  }
  public RangedWeaponSO FindRangedSO(string objectName) {
    for(int i = 0; i < rangedWeapons.Count; i++) {
        if(string.Equals(rangedWeapons[i].GetObjectName(),objectName)) {
            Debug.Log(rangedWeapons[i].GetObjectName());
            return rangedWeapons[i];
        }
    }
    return null;
  }
}
```

Zombie.cs:

```
using UnityEngine;
using UnityEngine.AI;

public class Zombie : MonoBehaviour {
  private NavMeshAgent agent;
  private Transform player;
  [SerializeField] private LayerMask playerLayer;
  private float attackRange = 2;
  private bool isWalking, isSprinting, isAttacking;
  private int health;
  void Start() {
    health = 75;
    this.gameObject.SetActive(true);
    player = GameObject.Find("Player").transform;
    agent = GetComponent<NavMeshAgent>();
```

```
}
void Update() {
    isAttacking = false;
    HandleMovement();
    if(Physics.CheckSphere(transform.position,attackRange,playerLayer)) {
        AttackPlayer();
    }
}

private void HandleMovement() {
    isWalking = true;
    agent.SetDestination(player.position);

}
private void AttackPlayer() {
    transform.LookAt(player.position);
    isAttacking=true;
}
private void OnCollisionEnter(Collision collision) {
    Debug.Log("test");
}
void OnCollisionStay(Collision collision) {
    Debug.Log(collision.gameObject.layer.ToString());
    if(collision.gameObject.layer == playerLayer) {
        Debug.Log("Hitting player");
        //player.damage(1);
    }
}

//anim get functions
public bool IsWalking { get { return isWalking; } }
public bool IsSprinting { get { return isSprinting; } }
```

```
    public bool IsAttacking { get { return isAttacking; } }
    //
    public void Damage(int dmg) {
        if((health-=dmg)>0) {
            health-=dmg;
        } else {
            health = 0;
            Destroy(this.gameObject);
        }
        Debug.Log("Zombie health now: " + health.ToString());
    }
}
```

ZombieAnimator.cs:

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

public class ZombieAnimator : MonoBehaviour {
    [SerializeField] private Zombie zombie;
    private Animator animator;
    private void Awake() {
        animator = GetComponent<Animator>();
    }
    private void Update() {
        animator.SetBool("isWalking",zombie.IsWalking);
        animator.SetBool("isSprinting",zombie.IsSprinting);
        animator.SetBool("isAttacking",zombie.IsAttacking);
    }
}
```

ZombieSpawner.cs:

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEditor;
using UnityEngine;

public class ZombieSpawner : MonoBehaviour
{
    [SerializeField] private GameObject zombie;
    [SerializeField] private Transform zombieSpawn;
    private float cooldown=7,i;
    void Start()
    {
        i = 0f;
    }

    // Update is called once per frame
    void Update() {
        if(i >= cooldown) {
            Debug.Log("Spawn");
            Instantiate(zombie,zombieSpawn.position,Quaternion.identity);
            i = 0f;
        } else {
            i += Time.deltaTime;
        }
    }
}
```

MainMenu.cs:

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

```
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public void StartGame() {
        SceneManager.LoadScene("GameScene");
    }
    public void QuitGame() {
        Application.Quit();
    }
}
```

PauseMenu.cs:

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

public class PauseMenu : MonoBehaviour {
    public static bool isPaused=false;
    [SerializeField] private GameObject crosshair;
    [SerializeField] private GameObject PauseMenuUI;

    void Start() {
        PauseMenuUI.gameObject.SetActive(false);
    }
    void Update() {
        if(Input.GetKeyUp(KeyCode.Escape)) {
            if(isPaused) {
                ResumeGame();
            } else {
                PauseGame();
```

```
        }
      }
    }

    public void ResumeGame() {
      isPaused = false;
      PauseMenuUI.gameObject.SetActive(false);
      crosshair.SetActive(true);
      Cursor.lockState = CursorLockMode.Locked;
      Time.timeScale = 1f;
    }
    public void ExitToMenu() {
      SceneManager.LoadScene("MainMenuScene");
    }
    public void ExitGame() {
      Application.Quit();
    }

    public void PauseGame() {
      isPaused=true;
      PauseMenuUI.SetActive(true);
      crosshair.SetActive(false);
      Cursor.lockState = CursorLockMode.None;
      Time.timeScale = 0f;
    }
}
```