Analysis:

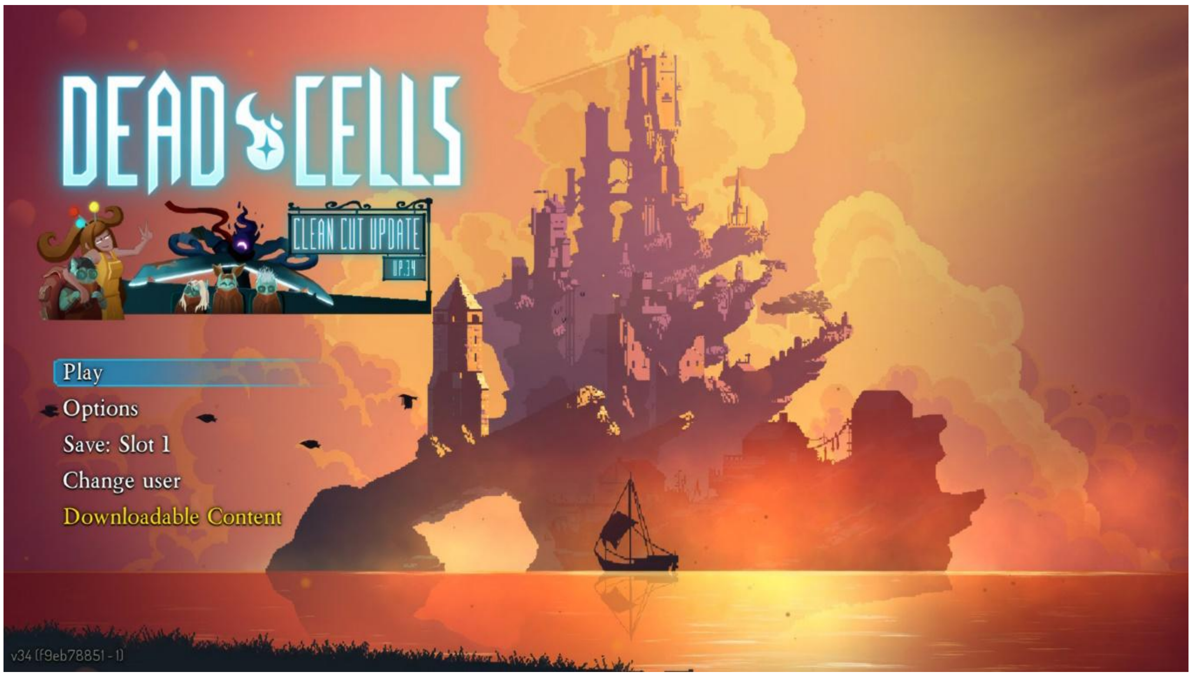
The solution

The idea I plan on working on is a 2D shooter game, which will focus on the reoccurring attempts by the player, and upon failure to beat the game will allow the player to upgrade their current items to have each run be easier than the last. It will allow the player to have a variety of different influences on their run and playstyle, including new weapons to change the moveset of the player, attachable items to the player to improve certain parts of the character, and permanent items which will carry across runs. The goal of the game would be to beat the last boss, and upon victory the player would be able to up the difficulty by sacrificing their progress. The challenge for the player so far would be basic enemies, platforming hazards and end-of-level enemy gauntlets or bosses serving as a challenge for the player to pass to the next area. I would like this game to be able to save and write the player's data on a file to be called back when they return to the game, as it would make the permanent upgrades truly permanent, as well as storing all of the items the player has unlocked so far. In addition, if I have the time after the main aspects of the game have been implemented, I could include a system where it adds certain modifiers to runs at the player's choice, like increased damage taken and dealt, or increased enemies.

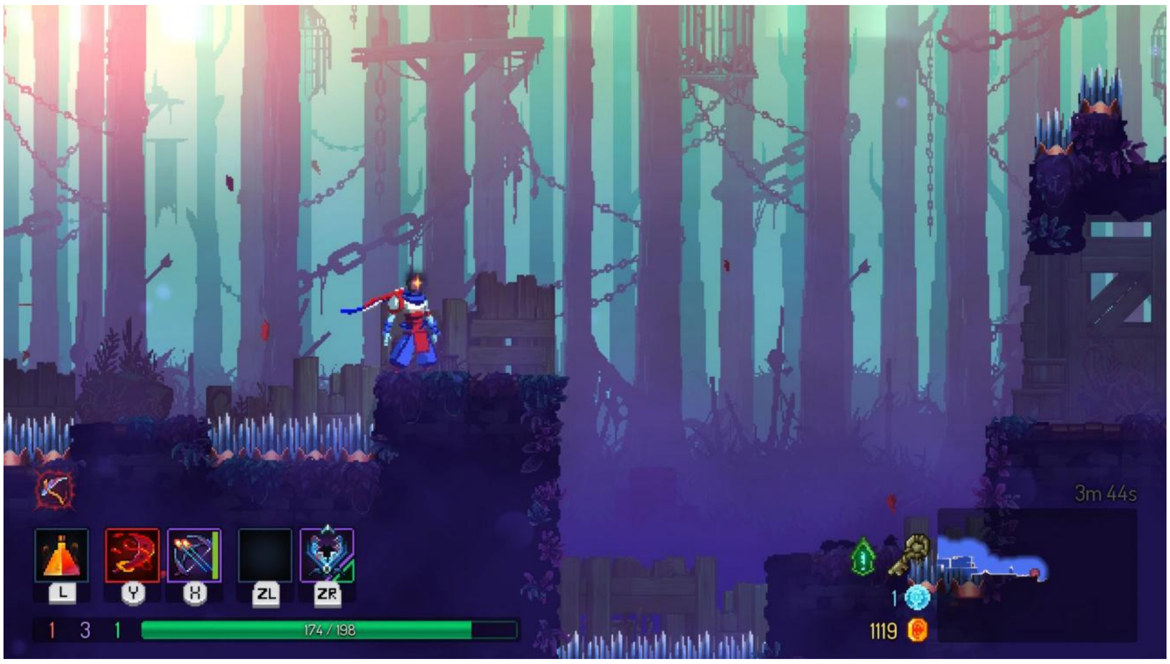
So far, this design has been a longstanding idea of mine, as I have always wanted to create a 2D permanent-death game. Other influences for this design have been other games I've played, like Dead Cells. It appealed to me as the game's high skill, high reward style, as well as the plethora of items made it very fun and engaging to play. There are many different things I will need to learn for this project, including how the target audience would receive a game modelled like my design, research on efficient and concise UI examples and additional features to modify subsequent runs.

The research

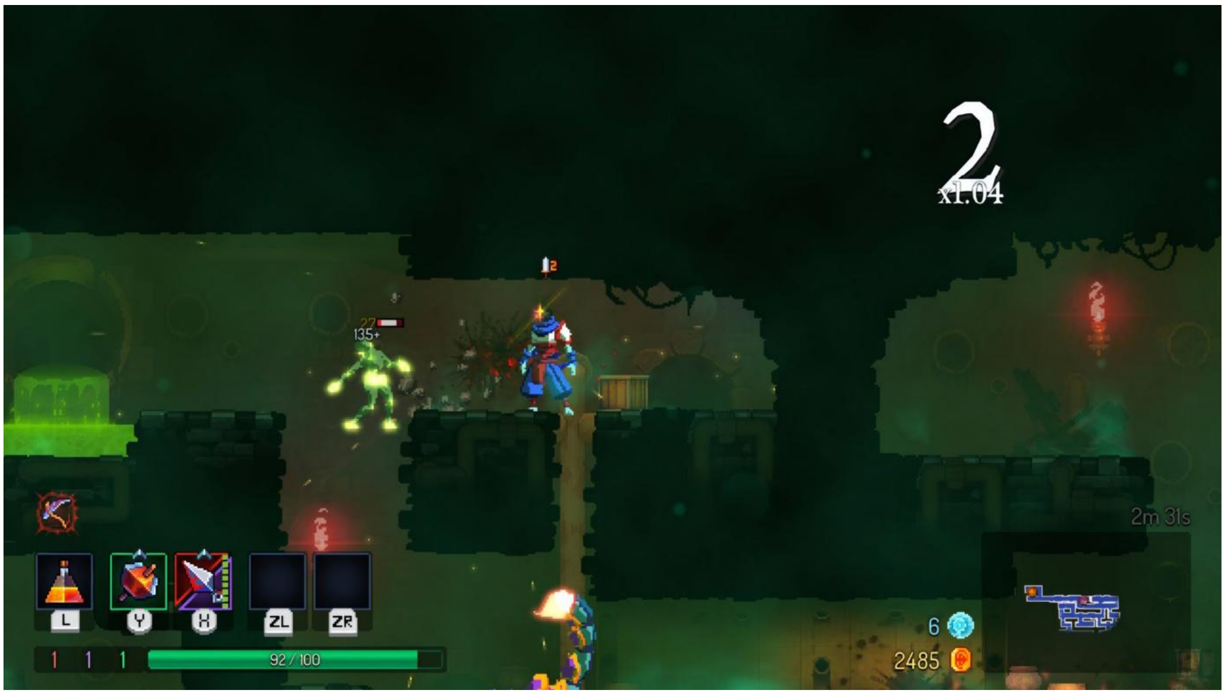
Menu Screen from Dead Cells – Link



UI from Dead Cells - Link



Dead Cells combat - Link

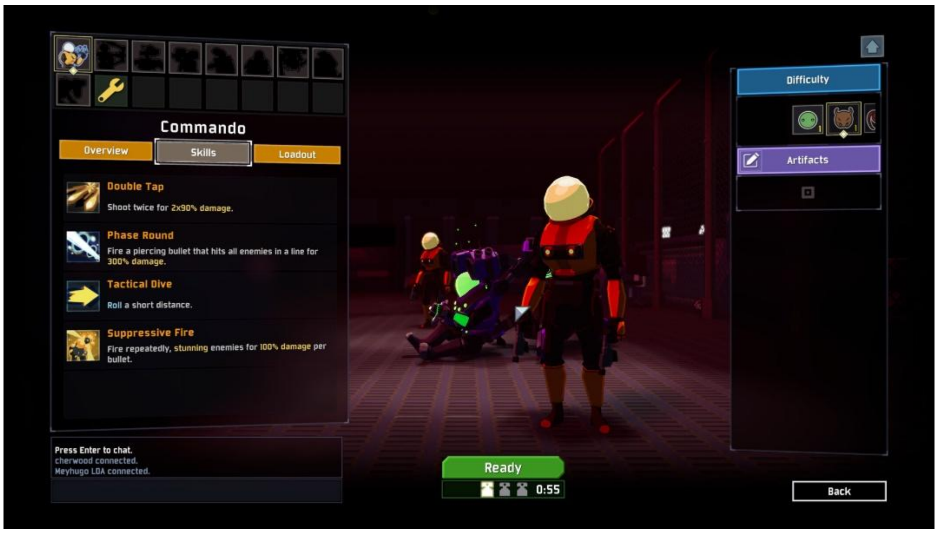


Dead Cells is a similar solution to my project. it features the permanent death mechanic, forcing the player to restart on death. It also provides a permanent currency to unlock new items for subsequent runs. The aesthetic of the game is grungy and bloody, showing that it is for a higher aged target audience. The main gameplay loop features the character finding weapons and defeating enemies to obtain more loot, as they try to find the exit to the level to progress to the next. This is also similar to my idea, but there are differences. The difficulty of the game does not scale with time, instead by level. The main incentive of each level is to find gear strong enough to carry forwards into the next level, whereas my idea would focus on the timing aspect. I plan on scalable difficulty with time, where if you take too long then the enemies get stronger. Overall, however, it has very similar aspects which are possible to implement into my project.

Things I would like to take forward/adapt into my project	**Things I would not like to take forward/adapt into my project**

- The controls on the UI are clean and readable, with the corresponding keys attached to the abilities, making the controls easier to learn - The icons are large enough for the player to know what they are, and small enough not to intrude on gameplay - The menu is clear and concise; it makes it less confusing to navigate and easy to access - The spikes seen in the 2nd screenshot show that there are platforming aspects in the game, something in plan on developing on in my project - The health bars above the enemies in the 3rd screenshot are short and easily interpreted, making the gameplay easier to be the main focus	- The map in the corner of the UI. I believe that a map would take away from the gameplay of my project, as instead the player should memorise a pre-built map - The health bar. I think that a bar of that length and style would also not fit into my project, instead I would use disappearing icons to represent lost health - The positioning. This is purely personal preference, but I think that the health in the top right would fit better, with the abilities in the bottom left, and the key binds on the right.

Risk of rain 2 character select screen – Link



Double Tap
Shoot twice for 2x90% damage.

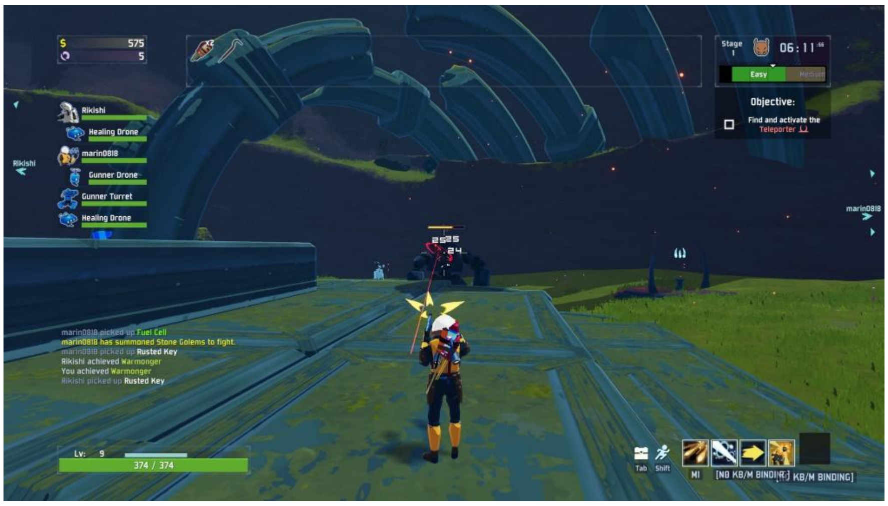
Phase Round
Fire a piercing bullet that hits all enemies in a line for 300% damage.

Tactical Dive
Roll a short distance.

Suppressive Fire
Fire repeatedly, stunning enemies for 100% damage per bullet.

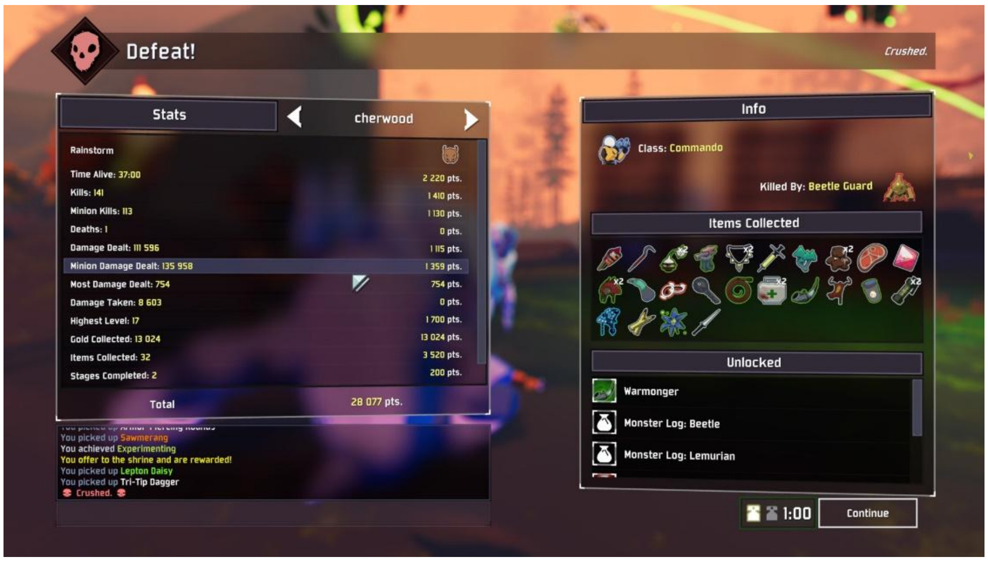
Press Enter to chat.
cherwood connected.
Meyhugo LDA connected.

Risk of Rain 2 UI – Link



marin0818 picked up Fuel Cell
marin0818 has summoned Stone Golems to fight.
marin0818 picked up Rusted Key
Rikishi achieved Warmonger
You achieved Warmonger
Rikishi picked up Rusted Key

Risk of Rain 2 defeat screen - Link



Rainstorm

Time Alive: 37:00	2 220 pts.
Kills: 141	1 410 pts.
Minion Kills: 113	1 130 pts.
Deaths: 1	0 pts.
Damage Dealt: 111 596	1 115 pts.
Minion Damage Dealt: 135 958	1 359 pts.
Most Damage Dealt: 754	754 pts.
Damage Taken: 8 603	0 pts.
Highest Level: 17	1 700 pts.
Gold Collected: 13 024	13 024 pts.
Items Collected: 32	3 520 pts.
Stages Completed: 2	200 pts.
Total	28 077 pts.

You picked up Sawmerang
You achieved Experimenting
You offer to the shrine and are rewarded!
You picked up Lepton Daisy
You picked up Tri-Tip Dagger
Crushed.

Class: Commando

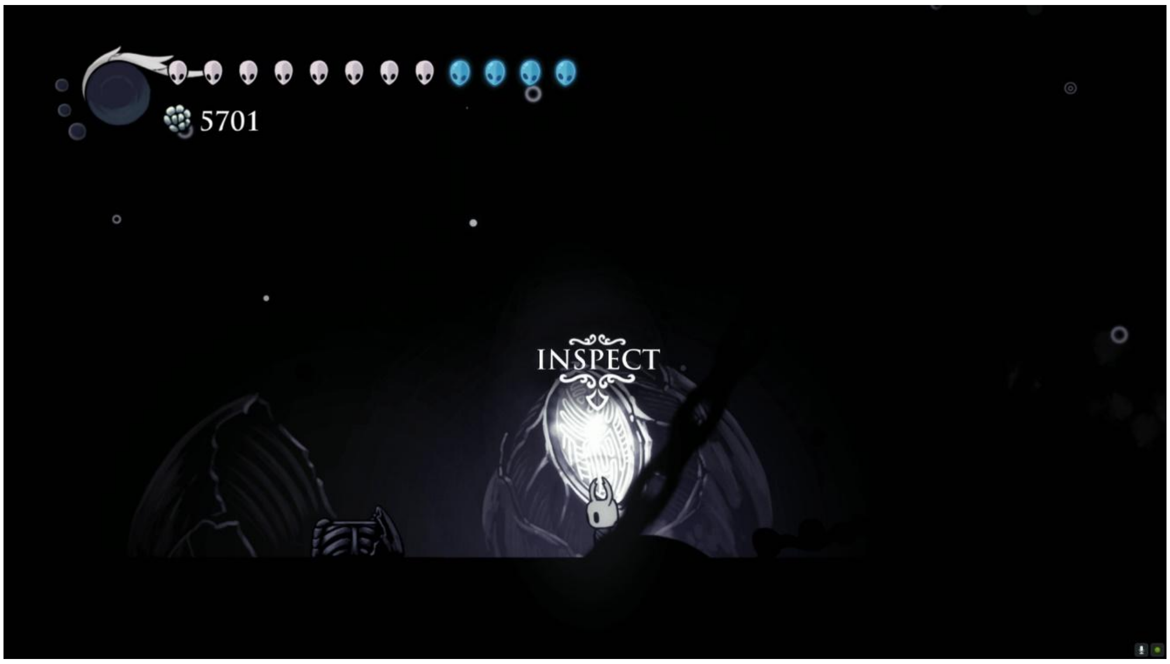
Killed By: Beetle Guard

Warmonger
Monster Log: Beetle
Monster Log: Lemurian

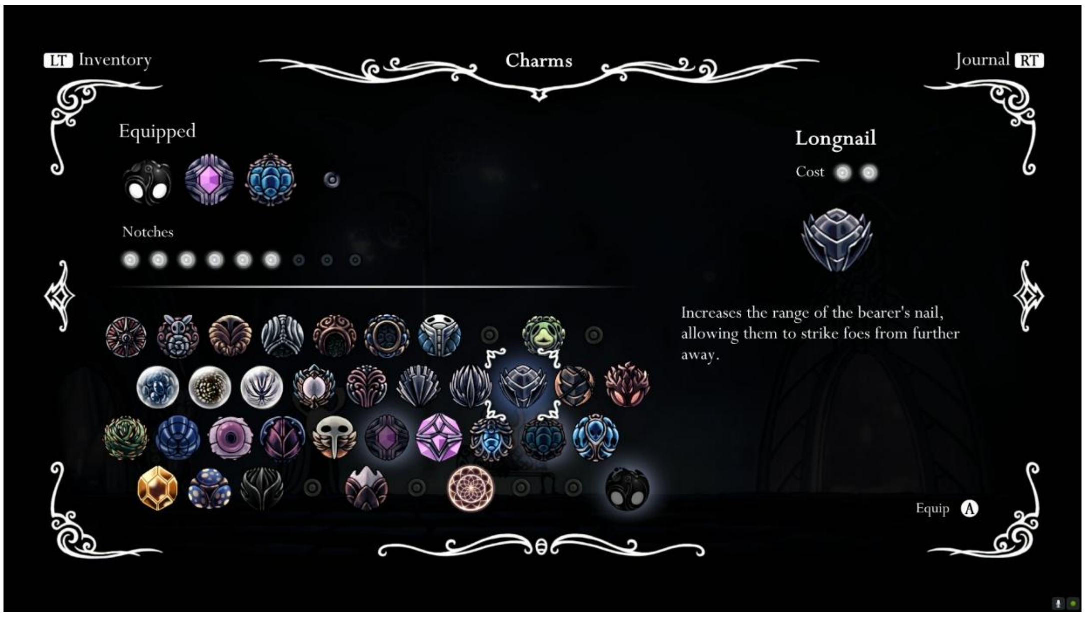
Risk of Rain 2 is a multiplayer shooter which has a heavy emphasis on difficulty scaling. The main mechanics of the game give you a plethora of characters with different abilities to choose from. These abilities, as well as the attached key binds are in the 2nd screenshot in the bottom right. I personally think that these positions are the best for each aspect of the UI. The main gameplay of Risk of Rain 2 is to find a teleporter to get you to the next level before the difficulty (The scrolling bar in the top right corner in the 2nd screenshot) becomes too high to beat. This is a good inspiration for the difficulty for my project, as it is entirely dependent on the player to choose their difficulty, provided they are fast enough. Another feature of Risk of Rain 2 are the items the characters can pick up. This is seen in the 3rd screenshot on the middle right, with each item giving the player a different bonus. The items system is a great feature for it to have, as the random number of items in Risk of Rain 2 give each run a uniquely different experience. I like the idea of an item system making the game completely random, so I will try and implement an item system. Lastly, Risk of Rain 2 also has a multiplayer system, where friends can join your run and fight with you. I do not think I will be able to add a multiplayer system, as it would make the timer-style difficulty scaling too easy to catch up with.

Things I would like to take forward/adapt into my project	**Things I would not like to take forward/adapt into my project**
- The items. There are so many which add complete randomness to each run, adding replayability to the game even after finishing it - The positioning of the UI elements. As I mentioned previously, I think that parts of the UI should be in different positions, with Risk of Rain 2 having the best positions so far - The difficulty scaling. The timed aspect of the difficulty adds a sense of urgency to each level, adding to the incentive to get to the next level as fast as possible	- The characters. I think that different characters would make my project unnecessarily complex, as it would add too many different abilities to keep up with, and balancing each one to be of equal power would be impossible - The Multiplayer aspect. I think that adding a multiplayer ability to my project would also make it too complex, as there would only be a limited number of items for each level, it would make it unfair if one person got them all before the other

Hollow Knight UI - Link

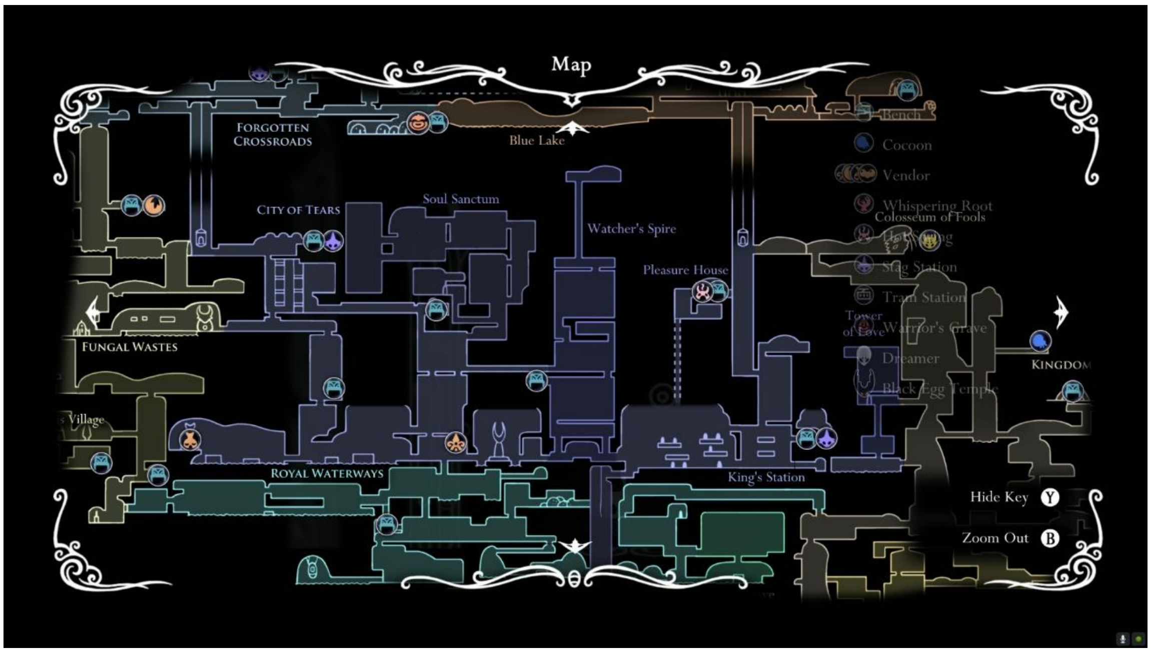


Hollow Knight equipment selection menu – [Link]



Increases the range of the bearer's nail, allowing them to strike foes from further away.

Hollow Knight map – [Link]



Hollow Knight is a 2D action-adventure platformer game focusing on the character's exploration of the map, as well as unlocking new abilities to access more parts of the map. The focus on unlocking new abilities will be key in my project, as new, permanent abilities will make new runs easier. Hollow Knight, first of all, does not have a permanent death mechanic. I will not be using it as inspiration. However, it does have an effect on death, where it limits resource gain until you reach your previous point of death. This could be implemented into my project, getting a past point of death and using it to gain more resources. Hollow Knight also has an equipment menu accessible from save spots. You only have a certain number of equipment slots, making you able only to attach certain amounts of equipment to the player. I will also not be implementing this style of equipment, as I don't think that having a limit to equipment will work in my game. Lastly, it has a large and unchanging map which the player can explore. This is a good idea, as I plan also on creating a large map to explore before death, where it will log all progress made on the map to carry over across runs.

Things I would like to take forward/adapt into my project	**Things I would not like to take forward/adapt into my project**
- The Map. I think that having a map which can carry across runs would be beneficial for players to not get lost. - The different equipment accessible to be found across the map. I think that having an exploration aspect for the project would be beneficial for players - The range of equipment. There is such a range of equipment to be accessed or used by the player, It makes each play unique to the style of the player, I think having variability for what you want the character to be is good game design practice	- The death mechanic. I think that despite having a marker for previous deaths would be cool, I think that losing all progress on death would be more aligned to the project - The equipment accessibility. I do not like having a limit to the amount of equipment you can put on, so I think that having an unlimited amount of equipment would be better - The difficulty being a consistent and not changing throughout the game. there are only two different difficulties, and the other just limits deaths to one. It does not change the enemies or their damage. This will not be the same in my project

The stakeholders

The target audience for this game are teenage boys who enjoy gaming, typically 15 – 18 years old. I chose this demographic because I know my friends of the same age have been playing and enjoying games of the same genre at the same age. Other reasons why this demographic does not focus on the younger people is that it will feature challenging level and enemy design, as well as difficult platforming sections. It also does not focus on the older people as people beyond college usually do not have the time or priorities to focus on gaming, making it less likely for them to play games. I could find out what the stakeholders want for games, which could help with my priorities and necessities for development. This will allow me to create my game according to the preferences to the stakeholders, therefore making responses to the game better. After development, I could ask the stakeholders what their opinions are on the near-final build during testing to see whether or not I have listened to or implemented their response, so I can then improve upon the final iteration with the target audience's preferences.

Hardware and software requirements

Hardware:

- A desktop computer or laptop computer. It will not be a mobile game
- A keyboard. It is required for inputs for the player to move and do certain actions
- A mouse. It is required for certain inputs for the player
- Computational power does not need to be particularly high but faster CPUs will result in higher performance
- Headphones or speakers may be required if the player wants to hear sound effects of music if added

Software:

- Any operating system which supports dotnet will be required
- Any estimate around 3-8 gigabytes of storage for the game

User requirements

According to a form with 18 responses, 15 of them said from their opinion the **most** important aspect of gaming. Many of them detailed more than one aspect in their response. This survey was sent to people around the age of users I predict to be using my project, so the responses should represent what the stakeholders want for the project.

- Three responses said they wanted a rewarding gameplay loop, meaning that they want a game with good replayability.
    - My idea of a rewarding gameplay loop is having unique experiences throughout subsequent runs, making each run uniquely fun. Despite this, there are too many different ways to make the game fun and enjoyable for repeated playthroughs. Many of these ways can be seen in different responses.
- Five responses said that the most important aspect of gaming is combat.
    - This is one of the ways of which gameplay can be both rewarding and provide replayability. Combat is a core aspect of the game I plan on making, so this is a good response as it shows that stakeholders also hold the same view on combat as I do. If the combat is well made, it makes the game replayable as people would enjoy playing the game to get better at the combat. On the other hand, there are many ways in which combat can be made, so having a satisfying combat could mean almost anything.
- One response said that games they prefer have no luck involved.
    - These two terms could mean many different things. The way I infer this idea is having no luck is having a consistent and unchangeable map and item drop chance. It could also mean that they want enemies which require you to learn their attacks without using luck to avoid or absorb damage. This is contrary in a few ways to my project, as I plan on having a random loot table and drop chance for those items. I also plan on having enemies and bosses randomised to ensure unique runs. I think a game where you have the same map and items for each new run gats boring as you learn the most optimal route, so I will not be implementing this.
- One response said variability.
    - This is the opposite to the response above, as variability makes games users are unable to predict, instead relying on luck and chance, as well as skill to play the game. again, this response could mean anything, so I infer it as uniqueness in subsequent runs. Variability could mean that all parts of the game change when the next run starts after the one before. It could also mean that loot could be a degree of quality higher than everything else you get, all up to

chance. This is the initial idea of my game, showing that stakeholders also hold some opinions which are similar to my ideas for the project.

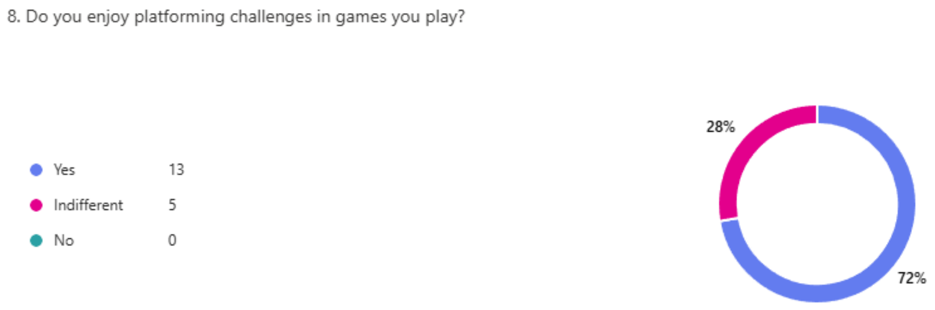
- Four responses said that they wanted smooth and fast movement.
    - Movement was a key part of the initial idea for the project, as both I and many stakeholders are fans of platformers and high-movement games. This was already a core component of the project to this only shows that stakeholders are also interested in my ideas
- Two responses said that they wanted a good story
    - While a story was not part of the initial plan for the project, as it is not required for the game to run or be played, a good story makes games interesting and engaging for people who enjoy story-related games. It shows how stakeholders would be more engaged in the game if it had a story, despite it not being a high priority

The follow up question to this one was asking the responders what the **least** important aspects of games they play were. Despite many of these responses overlapping with ones from the most important section, they represent a small minority of criticisms against those ideas. 13 responses from stakeholders within the target audience had detailed their opinion.

- 4 responses from stakeholders said that the story or plot was not important
    - I did not have a story planned for my game so far, and with a majority response saying that they do not want a story puts it very low on the list of priorities and requirements for my project
- Six responses said that they did not need good graphics or visuals to enjoy a game.
    - This was also not very important when I drafted my project, so this response from stakeholders shows that I do not need to worry about this unless if I have the time to add graphics, putting it lower the story on the list of priorities
- Two responses said that they did not care about weapons or upgrades.
    - This was an interesting response from stakeholders, as a majority from the list of most important parts was the combat and weapons. Despite being against these ideas, I will keep weapons and upgrades at the top of the list of priorities for development, as even though a ninth of my stakeholders are against it, a third said it was the most important part of gaming.
- One response was against microtransactions.
    - I had no plan nor idea to add microtransactions to my project, so this will not be a problem

I asked the stakeholders how important certain features I planned to integrate were to them in their opinion. These features are critical to my project, these responses show their opinion

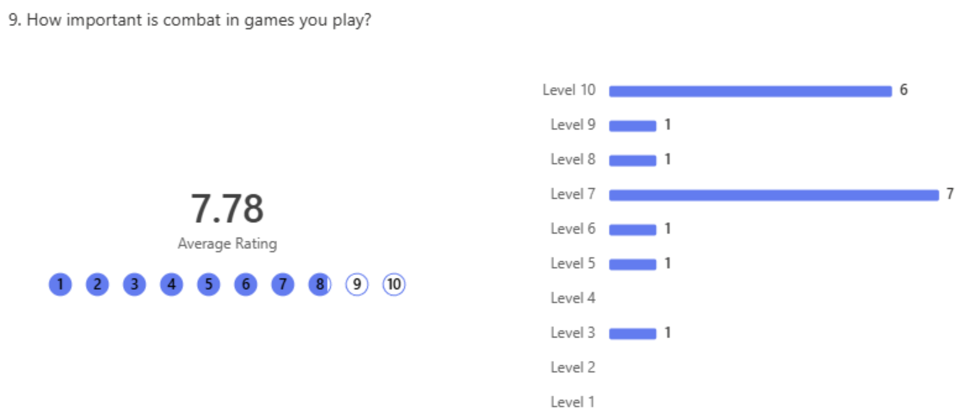
The first question asked about their opinion on platforming in games they play. Here are the responses:


8. Do you enjoy platforming challenges in games you play?

● Yes	13
● Indifferent	5
● No	0

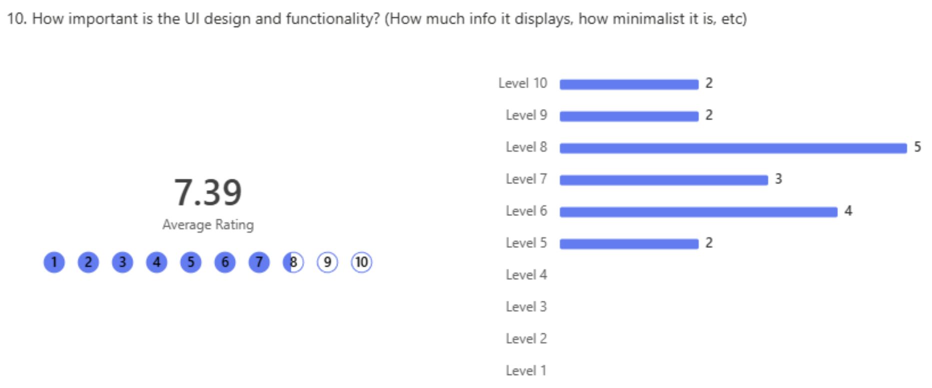
This shows me that platforming is an enjoyed part of gaming within the responses of the stakeholders. This is good, as it shows that my initial idea to add platforming challenges aligns with the views of most of the stakeholders. This means that it should be high on the list of priorities.

The next question I asked the responders was their opinion on the importance of combat in games they play. Here are the responses:



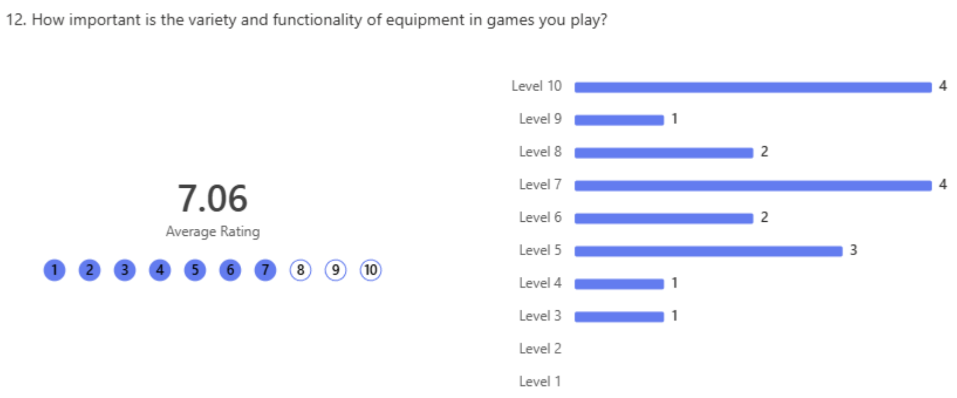
Of 18 responses, 6 saying that on a scale between one and ten it was the most important, and a further 7 on the scale saying that it was a seven show that combat is incredibly important to the stakeholders. This again aligns with my ideas for the project, as I plan on having combat being the main focus of the game.

Another question I asked was the importance of UI design in games. This was their response:



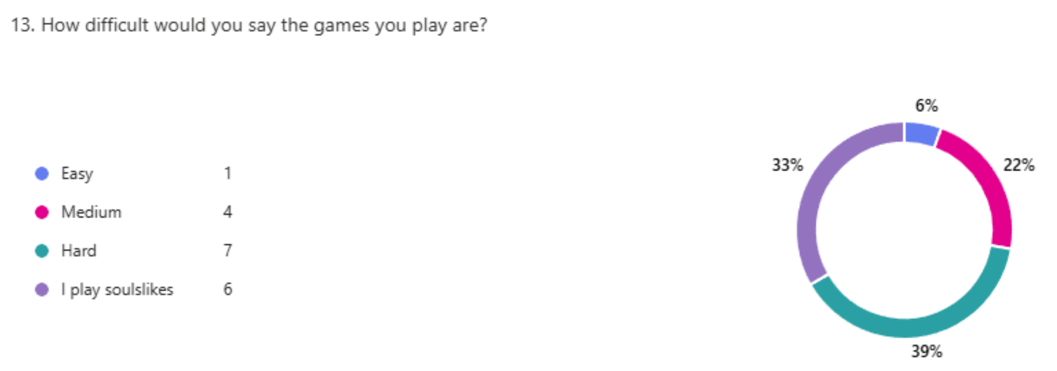
Of 18 responses an average of 7.39 out of 10 was given. This response surprised me, as UI design was not a very important part of gaming in my opinion. I planned to have the menu and UI be very simple and minimalist. Due to responses by my target audience, I will need to make the UI another requirement for the users.

One more question I asked was the importance of variety of equipment the player character can use in the game. This was the response:



This aligns with my original views for the project. despite the average being 7, just under a quarter viewed it as the most important part of games they play. This was already high on the list of user requirements, so this only confirms that most of the target audience agree that it is important.

One more question which was asked was the difficulty of games the target audience play. This was their response:



This chart shows that of 18 responses over 70% of my target audience play difficult games, showing that difficulty should be high on the user requirements. I already was planning on making the game difficulty fairly high, so this again confirms that my target audience would also enjoy games like that.

Computational methods

- There are parts which will have to happen with pipelining, as there are changes which can only be made after certain events have already happens, so there will be no relevant concurrent
- I plan on decomposing each section of the game which I can focus on, as it will make the development of the game a lot easier when there are problems to face in front of me instead of an entire game project.
- I also plan on using abstraction to remove all unnecessary details from the decomposed problems in order to more clearly see what I need to do for development. I work faster knowing what I need to do instead of assuming what I should do
- I also want to use visualisation in order to help summarise what I have decomposed in order to better focus on the more important tasks for the work. Again, this all makes it easier for me to work, as I know exactly what I need to do, easier still according to a diagram

Limitations

Some of the features listed below are not essential, but would like to be included in the game as it would look and feel a lot better if they were features. This is because there might simply not be enough time to learn and implement certain features. Others might be that I do not know how to implement them at all. Some of these include but are not limited to:

The title card: it is not essential to the game, yet it would look a lot better if it had an official title card with the name and emblem on it. While this may seem simple, designs are not, so time must be invested into it if I want to create a lasting and impressive title for the game

Item variation. Again, this goes down to the time it will take for me to implement all of the different items I want to add to the project. I also do not want the game to be filled with items which are bad, or too many for the game to not be enjoyable, as it would be too random. Therefore, I will try to limit the number of items the player can get in the game to prevent this

Settings menu: This may not be implemented as it seems too much of a problem for it to be created. Each setting must be a different thing to change within the player's code, adding only more problems to an already complex part of the main code. It is not a large priority due to this, and only if I have completed the player class fully will I implement settings for the player

Music and sound effects: I am not a sound designer nor produces, so both of these will be impossible for me to create. Instead, I will try to focus on the game itself being complete before I try to find suitable replacement audio for effects and music, each depending on certain prerequisites to play. I will bloat up the game if there are too many, so they become even less of a priority.

User requirements

Feature		Sub-feature	Explanation	Justification	Importance
Menu screen	1	Title card	It will display the name of the game in a large and readable font	It will show the user what game they are playing	Desirable
	2	Start button	It will boot the save file	It will allow the player to play the game	Essential
	3	Exit button	It will close the game	It will allow the player to exit the game through the menu screen instead of closing the application	Essential
	4	Save file button	It will make the player choose which save slot they want the game to run in	It will allow the player to choose which save they run, so sharing a computer doesn’t mean that progress for each player is lost	essential
	4	Background	It will show a relevant background to serve a scenery	It would make the game look more polished and immersive	Desirable
UI	1	Health bar	It will display the current and max health of the user	It allows the player to see what their health is and plan their next actions accordingly	Desirable
	2	Equipped weapon	It will display the icon of the current weapon of the player	It allows the player to see what weapon they have so they know for sure it is equipped	Non-essential
	3	Equipped items	It will display the current items the player has on their character	It allows the player to see how many items they have, and their effects if hovered over	Desirable
	4	Difficulty bar	It will display how far the difficulty has progressed over time	The player can infer from it how fast they need to move of to prevent getting out scaled, or how tough the enemies are	Essential
Gameplay	1	Avatar of the player	It will show the representation of the player in the game	It will allow the user to see where the character	Essential

				is and what actions they are doing	
	2	Items on the character	It will display the equipped items on the character, and will follow through with animations when doing actions	It will allow the user to see what weapon or item they have on the character, and lets them see their actions when the do the input	Essential
	3	Movement	It will move the character across the screen and change their position to where they are	It allows the player to move, instead of standing still all game	Essential
	4	Attacking	It will allow the player to damage and kill nearby enemies	It makes the game a combat experience rather than a platforming game, which is not what I want	Essential
	5	Enemies	It will move towards the player to attack and potentially kill the player if they do not avoid or kill it. On defeat will drop money	It will add challenge to the game. the difficulty of the enemies will be decided by the time spent in the run, adding more challenge	Essential
	6	Money	A collectable item for the player to acquire across the run, allowing them to interact with certain objects and NPCs	It adds the essence of randomisation to the game, as money can be spent to add modifiers or get better weapons or items	Essential
	7	Chests	A vessel to be unlocked by a certain amount of money to give the player a random item	It allows the player to upgrade their character, and with the right degree of luck they can find items which can make their run last a lot longer	Essential
	8	Shops	An NPC who will sell the player random weapons for money	It allows the player to find different move sets for weapons, with each doing different damage, allowing for a lot of playstyles and customisation	Essential
	9	Bosses	Stronger variants of enemies serving as the final fight for each level. On defeat will allow the payer to progress to the next level	It adds challenge to the game, as it will require the player to have found good items, weapons and to have the sufficient amount of skill to beat it. Difficulty scaling will also include bosses, so they	Essential

				will be difficult if left too late	
	10	Death screen	On player death, a screen showing the statistics of the run (time spent, items, weapon, money, difficulty, cause, and a sad face)	It will show the player what they died to, their items and other statistics which may be interesting to the player.	Essential
	11	Final boss	A unique enemy serving as the final challenge to the game. on defeat will allow the player to beat the game	It will serve as the final hurdle to the game, requiring items, weapons and skill to beat, it will be the final goal of the game and serve as the last challenge for the player	Essential
	12	Victory screen	On victory, a screen showing the statistics of the run (time spent, items, weapon, money, difficulty, and a happy face)	It will show the player the statistics of their successful run	Essential
	13	Pause screen	On input, will pause all actions in the game for the player until resumed	It will allow the player to edit settings midway through the game, to take a break from gameplay, or to exit the game	Essential
	14	Permanent money	On ending a run, it will award the player with a new permanent money equal to their distance in the run, allowing them to purchase things in the hub area before going on a new run	It will make subsequent runs easier, harder, or add new modifiers when those are purchased by this money	Desirable
	15	Hub area	A pre-run area to explore, with shops only taking permanent money, an expedition table to begin new runs and attachable modifiers to make new runs harder or more unique	New modifiers keeping the game random make the game more replayable and enjoyable, as well as new shops allowing for more options when using certain weapons, items and money	
Map	1	Exploration	While moving through new areas, it will automatically update the player map with new information	It allows the player to be exploratory throughout the game, and will make it easier for them to find their way across to places they haven’t visited.	Essential

	2	Map hotkey	On player input, it will open the map and pause the game while it is open. It will display all discovered information.	It will let the player be able to explore areas they haven't discovered yet, as well as show their position for them to not get lost	Essential
	3	Icons	Icons on the map serving as representatives of objects in the game, like the player, the exit, unopened chests, shops and secrets	It will allow the player to interpret the map and see what places or items they have missed	Essential
Save files	1	Beginning a new file	It will create a new save file for the user to use when playing the game to continue their progress when they come back to the game	It allows the player to access and continue playing with a save file so they can progress through the game while keeping permanent money and modifiers	Essential
	2	Saving	It will update the save file with data from when it was saved	It allows the player to manually save their progress from a point to continue from	Essential
	3	Auto-save	It will automatically save the game on ending a run to keep all changes and extra permanent money	It is a safety net for when players inevitably forget to save their progress manually	Essential
	4	Deleting a save	Will delete the save file from the files	Allowing the player to do this gives them freedom to save up spaces for new save slots	Desirable
	5	Save slots	Holds 3 different slots for save files	Allows people who share computers to have different data to other people so they can continue their own runs on their own progress and data	Essential

Design:

ESSENTIAL ITEMS TO THE PROJECT

STAGE 1 – THE PLAYER

I plan on making the player and their weapons. This is vital as it is what provides the gameplay for the user. The player is more important than the enemies as there is no game without the player. The weapons will come as well because they will be attached to the player for the user to use.

Test No.	Test	Expected outcome
1	Movement – checking to see if input will make the character move	The player moves horizontally across the screen
2	Jumping – to check on input from spacebar if the player jumps	The player performs a jump
3	Dashing – to check on shift key input if the player performs a dash	The player performs a dash
4	Animation – to play animations when a player does something	The player performs an animation

STAGE 2 – SHOPS AND ITEMS

I plan on adding items after this, as the items cannot be attached if there is no player. I plan on designing at least 10 unique items adding new effects to the player, with more if I have time. I will have to modify the player data to suit the items, as well as adding an inventory, however both ae impossible again if there is no player.

It is after this I plan on adding shops and random chests to the background, so the player can access the items added in the last stage. Shops have to be added after items as the shops will not be able to sell if there is nothing there. The means to purchase them will be added after this, along with the enemies.

Test No.	Test	Expected outcome	Success?	Outcome
1	Adding random items to shop – to have the shop generate 3 different items to sell	It will have 3 items on display		
2	Generate random prices – it will assign a random price to each item	The items and weapons will have different prices		
3	Buying the item – to check on input if the player has enough money to buy the item	It will remove the money from the player's inventory		
3.1 – player class	Picking up an item – to check on collision and input if the player picks up an item	The player picks up the weapon and adds it to the inventory on input		
6	Giving the item to the player – it will remove	It will remove the item from the shop and add it		

	the item from the shop and add it to the player inventory	to the player inventory		

STAGE 3 – ENEMIES

I plan on adding enemy data after shops. There are many parts to the enemies. They will drop money for the player to buy things from shops, as you cannot access shops when broke. I also plan on adding at least 4 enemies, with more after initial development (Similar to items). The reason enemies are made after the shop and items is because I plan on having them have a random chance to drop an item (Impossible without items already existing) and for them to drop money on death (No purpose if shops do not exist).

Test No.	Test	Expected outcome	Success?	Outcome
1	Movement – allows the enemy to automatically move towards the player	It will detect the player and move towards them		
2	Attack – it will swing towards the player and try to hit when in range	It will attempt to attack the player		
3	Damage – it will deal damage to the player on contact. It will deal more damage if it was attacking	It will remove the health from the player and stun the player		
3.1 – player class	Death – on taking cumulative damage which reduces the player health to zero or less, it will play an animation and die	The player will die		
4	Taking damage – the player can damage the enemy by shooting it with their weapons	It will remove health from the enemy on hit		
4.1 – player class	Shooting – to check on mouse input if the player shoots if they have a weapon	The player shoots their weapon and creates a projectile if they have a weapon equipped		
5	Death – on HP being reduced to 0 or less, it will play an	It will die		

	animation and remove itself			

STAGE 4 – MENUS AND ACCESS TO DIFFERENT SCENES

I plan on creating the Menu next. The background of the menu can be left for later as it is not necessary. The buttons can take users to new menus, like save files (Will get to later) and the game itself. It also provides the buttons which can access different scenes.

I plan on adding a pause menu after this, as it will provide access to the main menu after this. This will sacrifice all progress made up to the point in the run, as you are leaving it. It is essential for the player to have, so it is after the player.

STAGE 5 – THE MAP

I plan on making the framework for the game map next, having a map is important to the project. The map access key bind will also be made here for the player to access the map. This can therefore make building the map, enemy placements and design choices be easier later on after it is done.

STAGE 6 – DEATH

I plan on adding the death mechanic and screen after enemies, which will provide the player access to the in-game menu on defeat (More on this later), as well as sense of danger when facing enemies. It will also loop the player back to the beginning of the game with no items and the starting weapon, to add challenge to the game. This is added after enemies as you cannot die without enemies.

<u>NON-ESSENTIAL ITEMS TO THE PROJECT</u>

STAGE 7 – BOSSES

I plan on adding bosses after enemies because a boss is in my opinion a larger and stronger enemy. With the framework for enemies, it will be a lot easier for me to add bosses. I plan on giving the bosses new attacks, increased health pools, and a challenge to the player. I also plan on adding a large amount of loot on defeat, as the player will have surpassed a challenge.

I then plan on adding a final boss to serve as the ending to the game. It is after bosses as they also provide framework for me to work on as an extension of them. This stage is not entirely necessary, as if I choose, I can make the game an infinite crawler, yet that is not my plan. It is quite late into the development plan, as this is the stag where I plan on making non-essential extensions to the game.

STAGE 8 – LEVELS AND MAP

Now I plan on adding levels to the game, to serve as a higher difficulty past a boss. This is after bosses because I plan on providing access to the next level after a boss is defeated. New levels will require new maps, so it is after the maps as well for them to provide the framework for a new level. It is also by this point I will consider adding new enemies to fit an aesthetic of a new level.

STAGE 9 – WIN

I plan now on making a victory screen, which will congratulate the player on beating the game. It will then unlock an additional difficulty scaling for the player to then access and use. This is after the final boss for the reason of it being required to be defeated in order to access this screen. The screen will provide access to the in-game menu providing access to upgrades (Will be added later) and the main menu.

STAGE 10 – MAP DESIGN + PLATFORMING

I then plan on adding challenges to the maps (Platforming) after this, as it is a secondary source of dying. It again is a secondary priority to me, as its not completely essential to having a complete game. I will have to modify the map data with this stage, as well as adding stage threats, like spikes or pits. This all requires a player, so it is after the player data and map data, but after the enemies as it is not as essential as that.

STAGE 11 – RUN-ALTERING AND PERSISTENT CURRENCY

It is after this I will add a permanent currency which will persist past player death. It will be earned upon defeating bosses, providing upgrades to the player which will affect all subsequent runs. This is a key component of any roguelite game. You will gain more the further you get into a run. This is added now because it is not only non-essential, but also you need to beat a boss for it to be gained.

STAGE 12 – VICTORY SCREEN

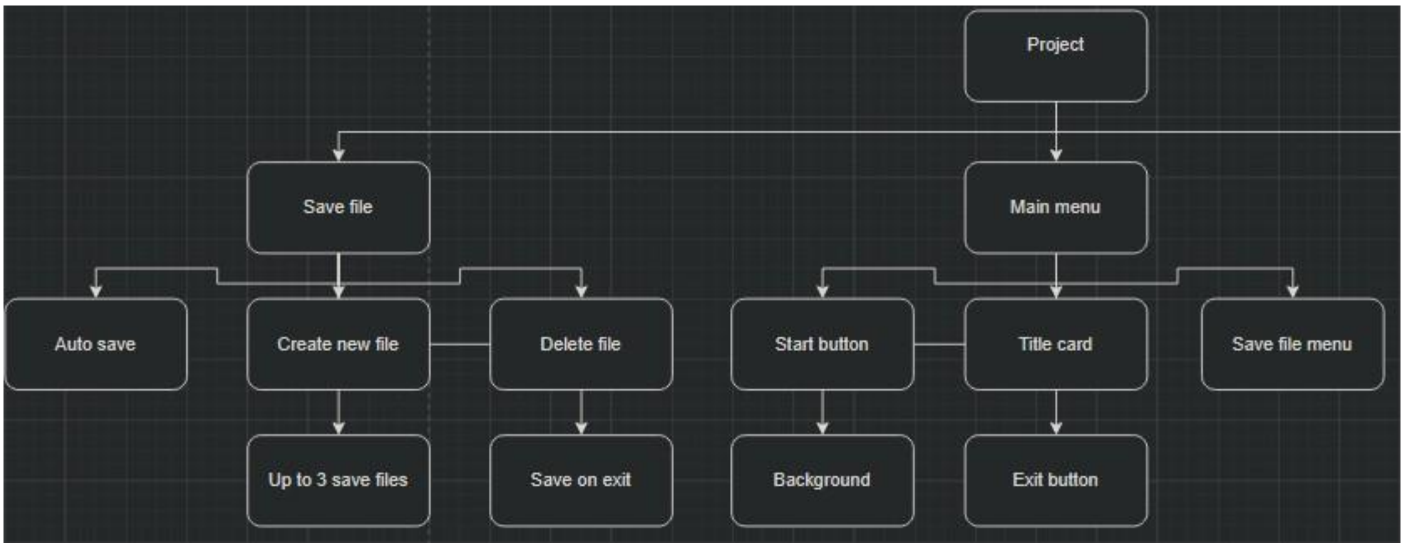
I will add the in-game menu after this, provided to the player after their death or victory. This will give them access to permanent upgrades or different characters (Added later). Permanent currency will be the way in which the player is upgraded in this screen, so it is after that stage.

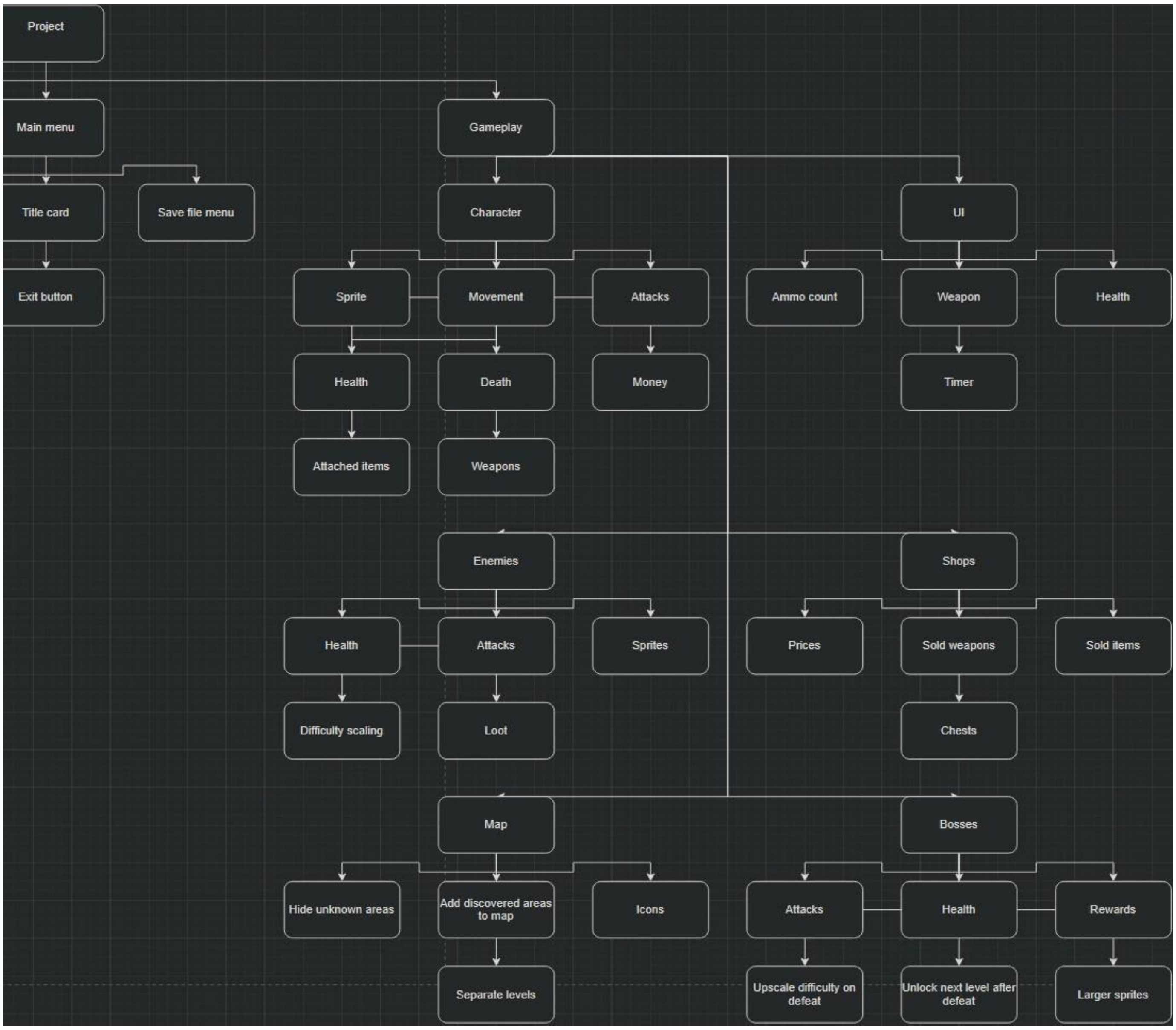
STAGE 13 – SAVE FILES

It is here I plan on making save files. This is largely uncharted territory for me, so I will need time to figure out how to create and modify files. However, the player shall provide a lot of access to the files, so it is after that. It is also an essential feature, however, it is after the point where players gain access to the permanent upgrade menu, so those can be saved too.

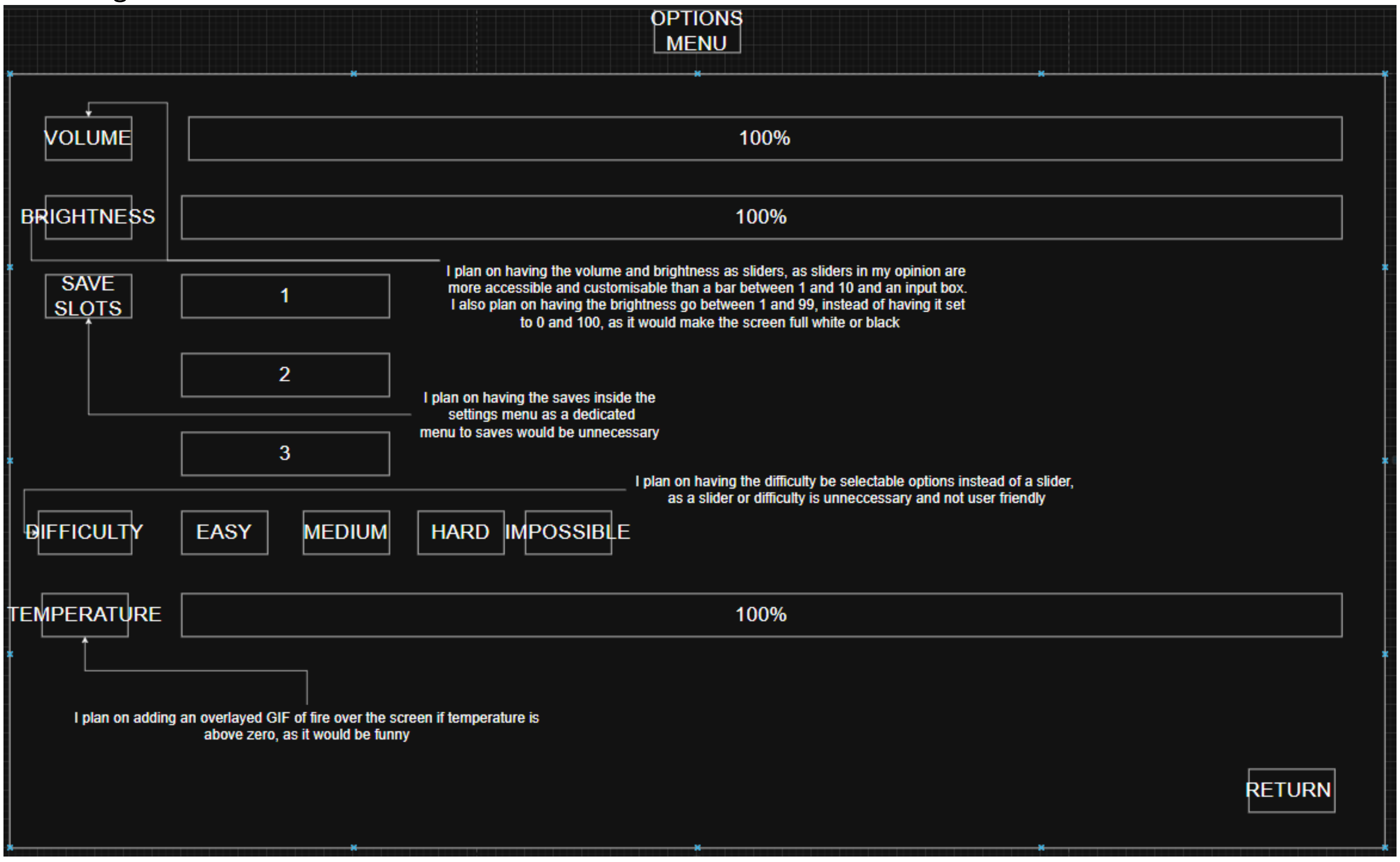
It is only after this that I will figure out how to add multiple save files for the user to use. It will give them access to 2 other files, so multiple users can save their progress on the same application. It is only after the save file part in the plan, so there will be no other plans for this section to give me time on how to figure it out.

FLOWCHART





GUI design



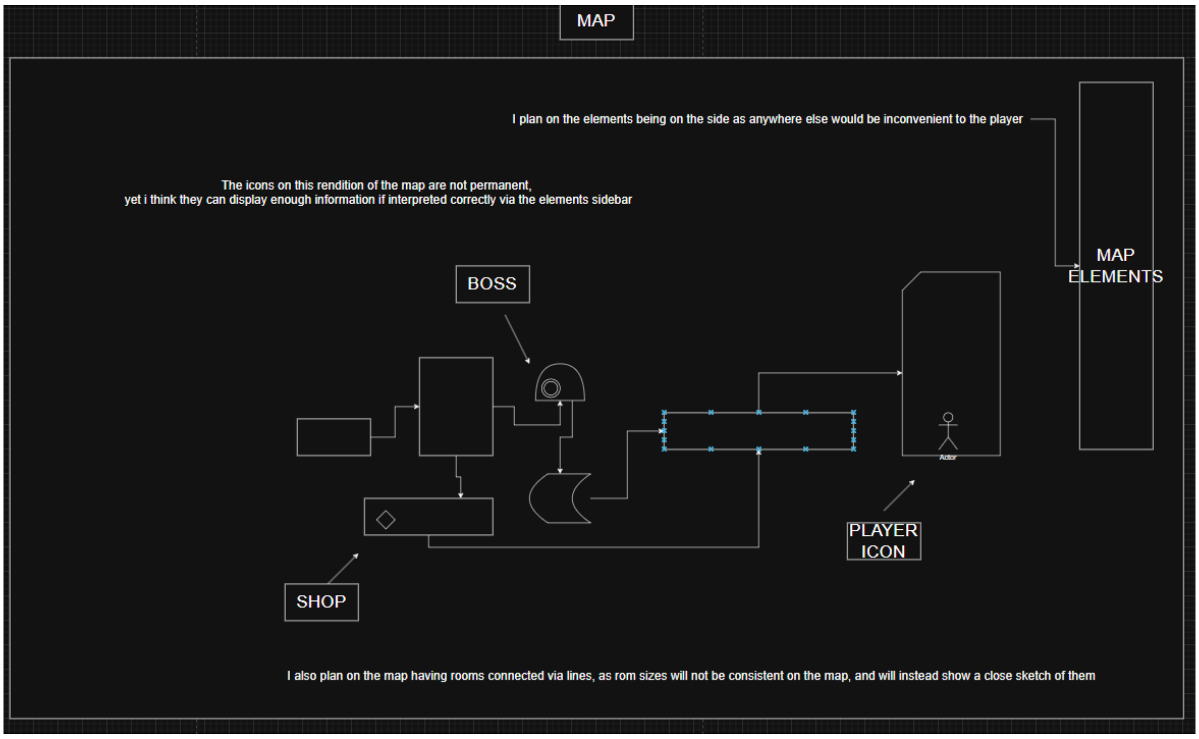
I plan on having the volume and brightness as sliders, as sliders in my opinion are more accessible and customisable than a bar between 1 and 10 and an input box. I also plan on having the brightness go between 1 and 99, instead of having it set to 0 and 100, as it would make the screen full white or black

I plan on having the saves inside the settings menu as a dedicated menu to saves would be unnecessary

I plan on having the difficulty be selectable options instead of a slider, as a slider or difficulty is unneccessary and not user friendly

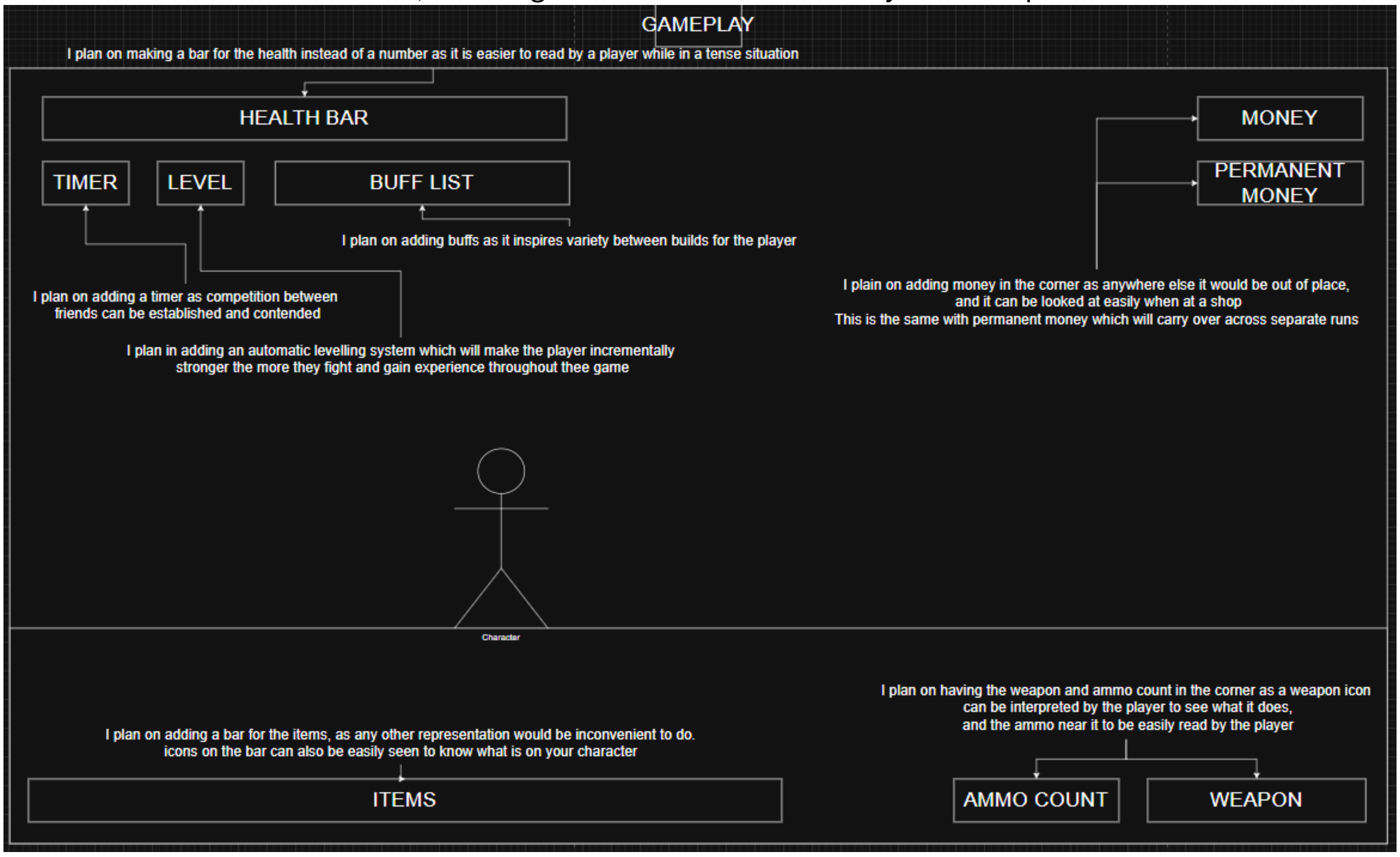
I plan on adding an overlayed GIF of fire over the screen if temperature is above zero, as it would be funny

The options menu will be designed in a user-friendly way, with large and clear icons for the player to read and interact with, allowing them to navigate the menu easily. It will also feature a few settings which will affect the game, like difficulty and volume. It will also feature a return button to leave this menu. I chose not to add a manual saving to the settings as it is common for players to forget to apply their changes to settings.



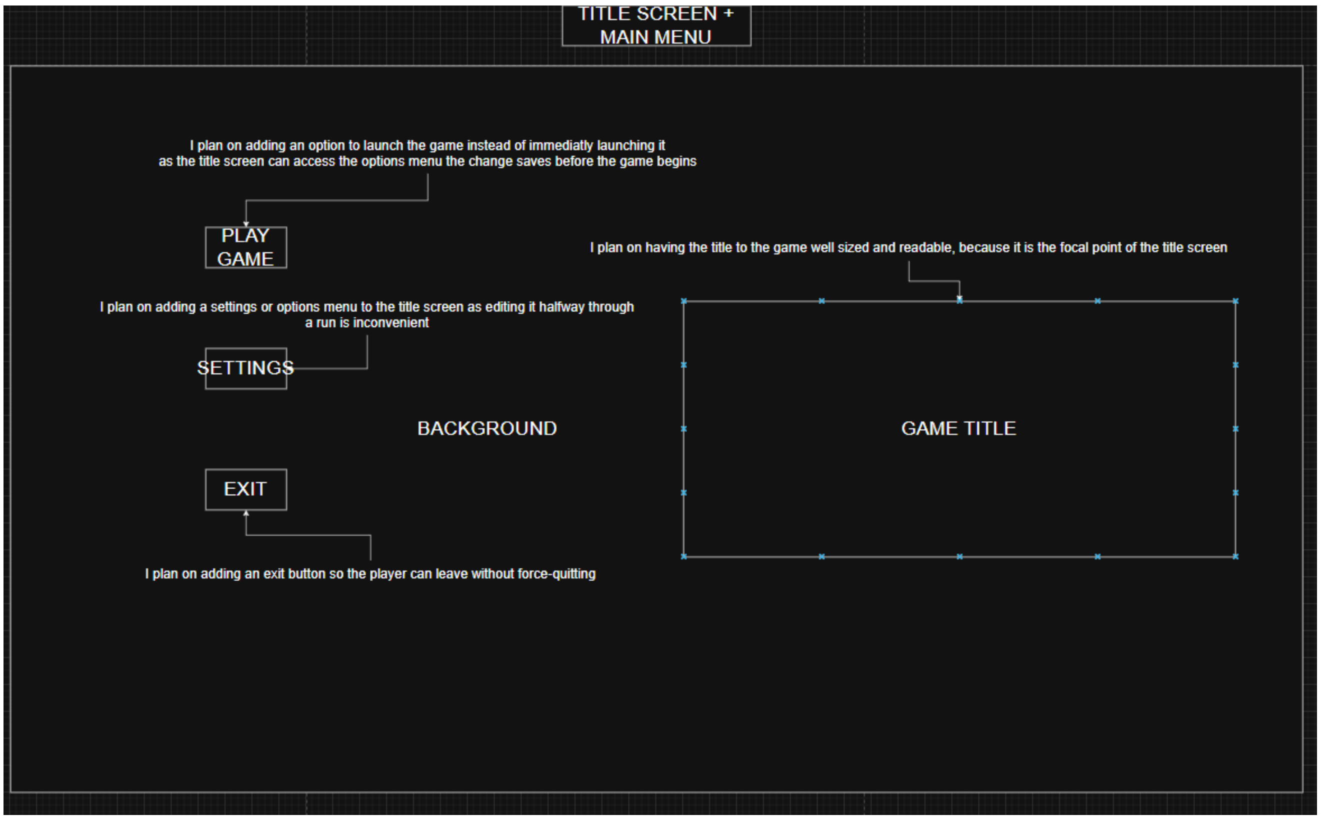
MAP
I plan on the elements being on the side as anywhere else would be inconvenient to the player
The icons on this rendition of the map are not permanent,
yet i think they can display enough information if interpreted correctly via the elements sidebar
MAP ELEMENTS
BOSS
PLAYER ICON
Actor
SHOP
I also plan on the map having rooms connected via lines, as rom sizes will not be consistent on the map, and will instead show a close sketch of them

The map will be accessed by a key bind of my choice, something suitable. It will also have a key for players to read and interpret, allowing them to interpret the map. Connections via arrows will show the connections between the rooms, which again adds to the accessibility of the map.



GAMEPLAY
I plan on making a bar for the health instead of a number as it is easier to read by a player while in a tense situation
HEALTH BAR
TIMER
LEVEL
BUFF LIST
I plan on adding buffs as it inspires variety between builds for the player
I plan on adding a timer as competition between friends can be established and contended
I plan in adding an automatic levelling system which will make the player incrementally stronger the more they fight and gain experience throughout thee game
MONEY
PERMANENT MONEY
I plain on adding money in the corner as anywhere else it would be out of place, and it can be looked at easily when at a shop
This is the same with permanent money which will carry over across separate runs
Character
I plan on having the weapon and ammo count in the corner as a weapon icon can be interpreted by the player to see what it does, and the ammo near it to be easily read by the player
I plan on adding a bar for the items, as any other representation would be inconvenient to do. icons on the bar can also be easily seen to know what is on your character
ITEMS
AMMO COUNT
WEAPON

The UI design I chose is similar to the projects which I chose to research. These elements show easy readability by the player, allowing quicker interpretations of the UI during combat or gameplay. Other elements are explained in the image, each showing a reason for my choice of placement.



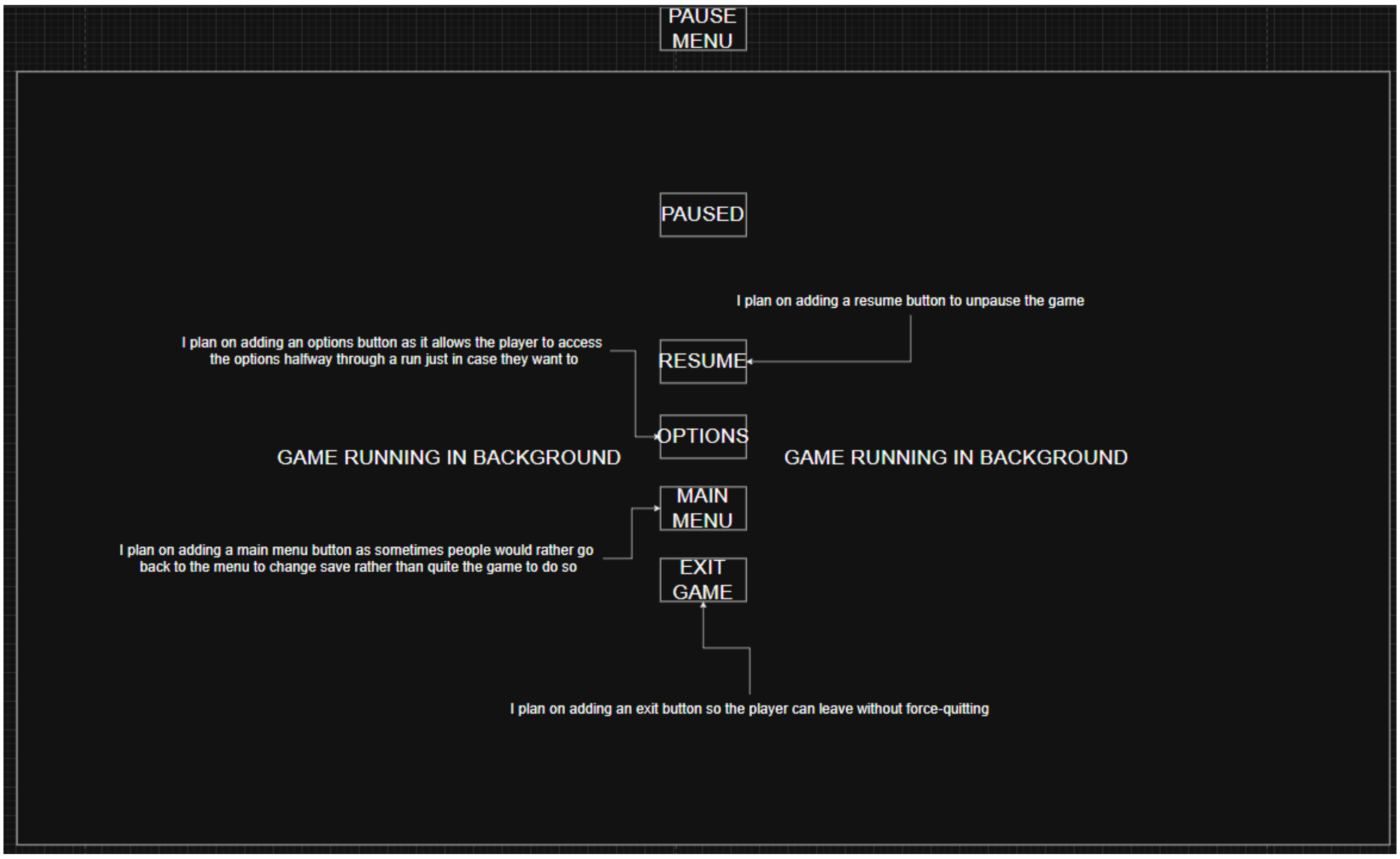
I plan on adding an option to launch the game instead of immediatly launching it as the title screen can access the options menu the change saves before the game begins

I plan on having the title to the game well sized and readable, because it is the focal point of the title screen

I plan on adding a settings or options menu to the title screen as editing it halfway through a run is inconvenient

I plan on adding an exit button so the player can leave without force-quitting

The title screen will include more decoration than this, yet it's functionality will remain the same. Each button has a purpose to fill, with no unnecessary buttons or any other way to lose accessibility for the player. It is simply designed, allowing for a faster development and easier access for testing



I plan on adding a resume button to unpause the game

I plan on adding an options button as it allows the player to access the options halfway through a run just in case they want to

I plan on adding a main menu button as sometimes people would rather go back to the menu to change save rather than quite the game to do so

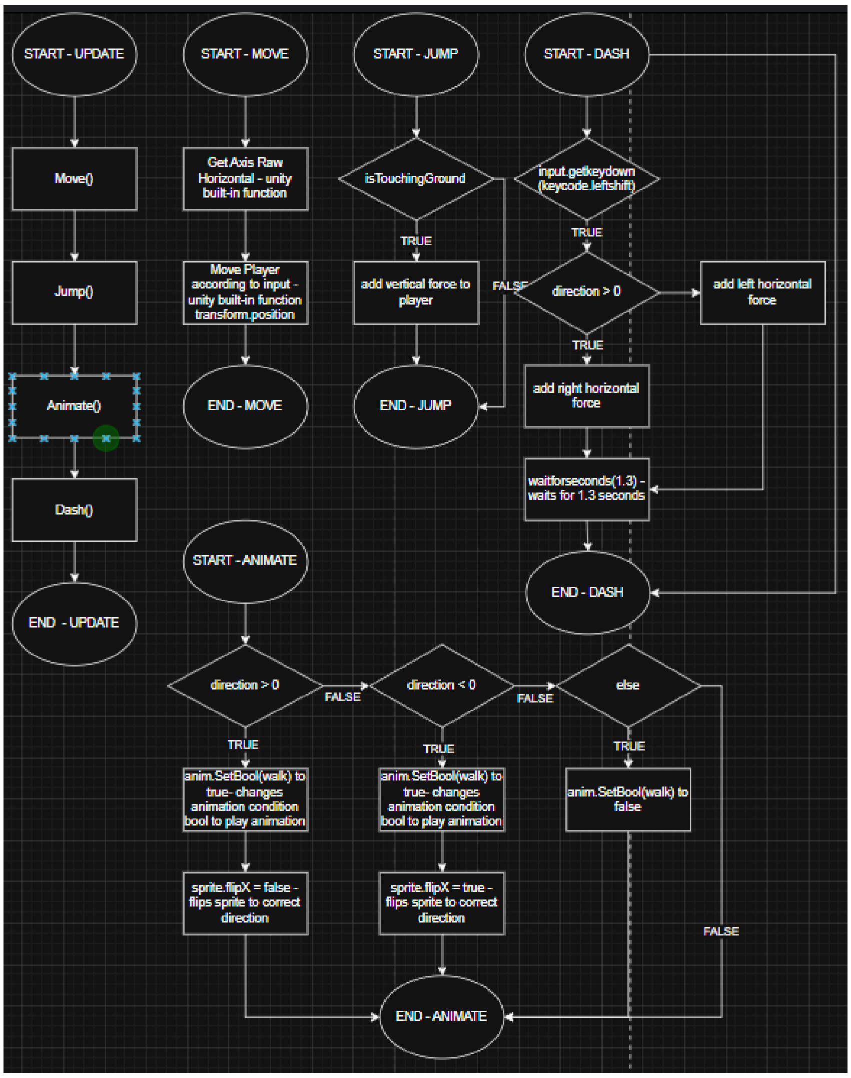
I plan on adding an exit button so the player can leave without force-quitting

The pause menu is similar to the main menu, again showing a simple design with minimal non-essential features. I chose to add an exit button as well as a main menu button to give the player more choice of how to exit, especially since you can change saves within the main menu options. It again is similar to the title screen with it's simplicity and efficiency, again allowing for a faster development and more efficient testing.

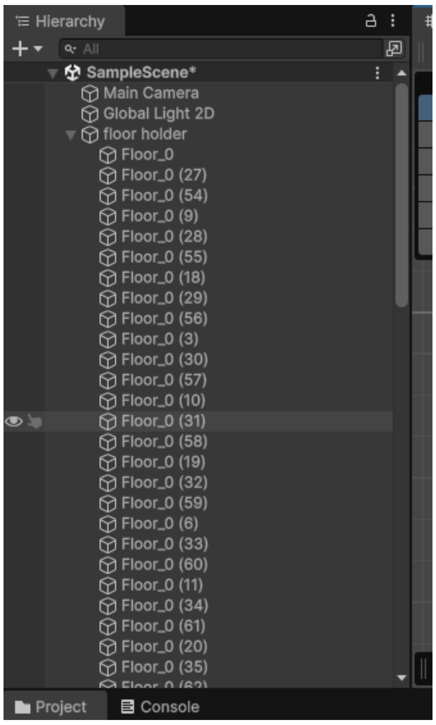
Development:

Player – diagrams:

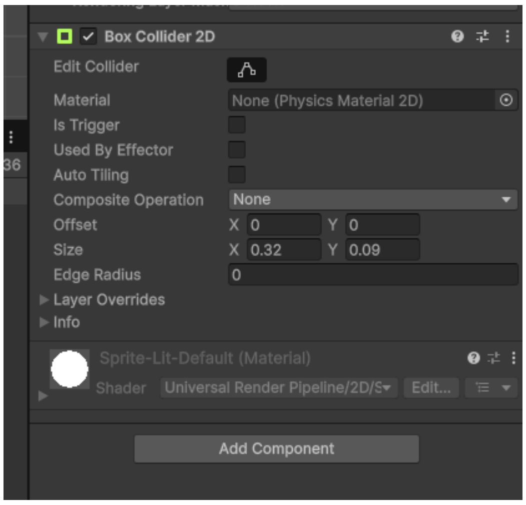
Player
-direction: string -move: float -jump: float -isTouchingFloor: Boolean -dash: string -inventory: string list -body: rigidbody2d -anim: animator -sprite: spriterender
-Update(): void -Shoot(): void -Jump(): void -Move(): void -Dash(): void -Animate(): void -PickUp(): void


START - UPDATE
Move()
Jump()
Animate()
Dash()
END - UPDATE
START - MOVE
Get Axis Raw Horizontal - unity built-in function
Move Player according to input - unity built-in function transform.position
END - MOVE
START - JUMP
isTouchingGround
TRUE
FALSE
add vertical force to player
END - JUMP
START - DASH
input.getkeydown (keycode.leftshift)
TRUE
direction > 0
TRUE
add right horizontal force
add left horizontal force
waitforseconds(1.3) - waits for 1.3 seconds
END - DASH
START - ANIMATE
direction > 0
FALSE
direction < 0
FALSE
else
TRUE
TRUE
TRUE
anim.SetBool(walk) to true- changes animation condition bool to play animation
anim.SetBool(walk) to true- changes animation condition bool to play animation
anim.SetBool(walk) to false
sprite.flipX = false - flips sprite to correct direction
sprite.flipX = true - flips sprite to correct direction
FALSE
END - ANIMATE

For the beginning of part 1 there are a few prerequisites to create


Hierarchy
All
SampleScene*
Main Camera
Global Light 2D
floor holder
Floor_0
Floor_0 (27)
Floor_0 (54)
Floor_0 (9)
Floor_0 (28)
Floor_0 (55)
Floor_0 (18)
Floor_0 (29)
Floor_0 (56)
Floor_0 (3)
Floor_0 (30)
Floor_0 (57)
Floor_0 (10)
Floor_0 (31)
Floor_0 (58)
Floor_0 (19)
Floor_0 (32)
Floor_0 (59)
Floor_0 (6)
Floor_0 (33)
Floor_0 (60)
Floor_0 (11)
Floor_0 (34)
Floor_0 (61)
Floor_0 (20)
Floor_0 (35)
Project
Console

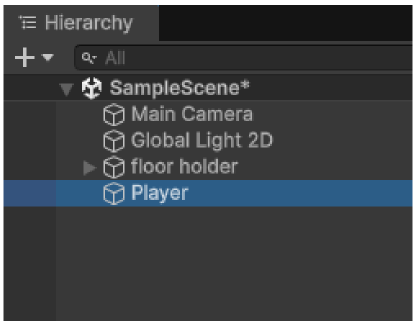
You need a ground for the player to stand on, so I have created a holder and duplicated floor sprites into it to create the floor


Box Collider 2D

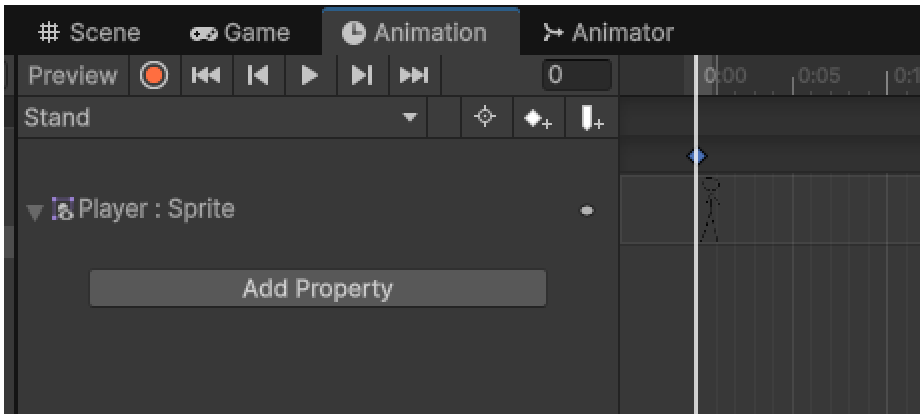
Edit Collider	
Material	None (Physics Material 2D)
Is Trigger	
Used By Effector	
Auto Tiling	
Composite Operation	None
Offset	X 0 Y 0
Size	X 0.32 Y 0.09
Edge Radius	0

Layer Overrides
Info
Sprite-Lit-Default (Material)
Shader Universal Render Pipeline/2D/S Edit...
Add Component

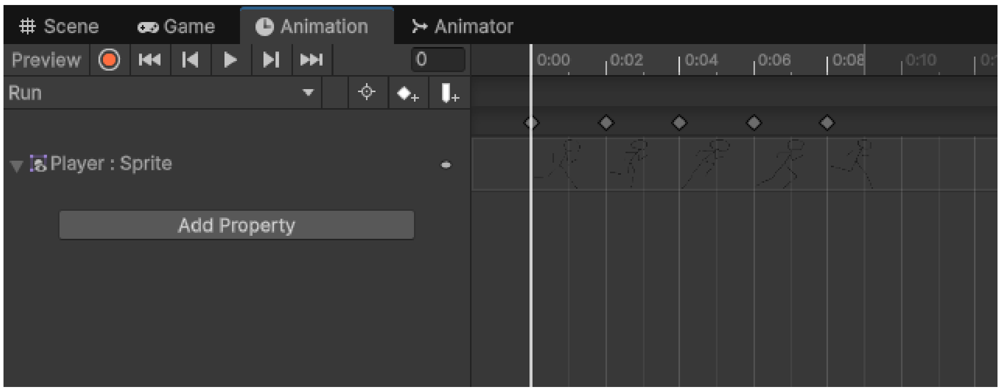
It is after this I add the component boxcollider2d to them to give them colliders or it will be unable to hold the player


Hierarchy
SampleScene*
Main Camera
Global Light 2D
floor holder
Player

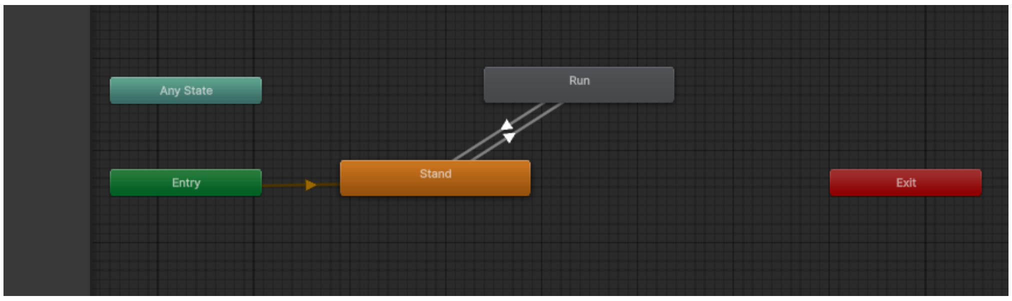
I then create the player object, this will be the main area which I add the scripts, components and animations to



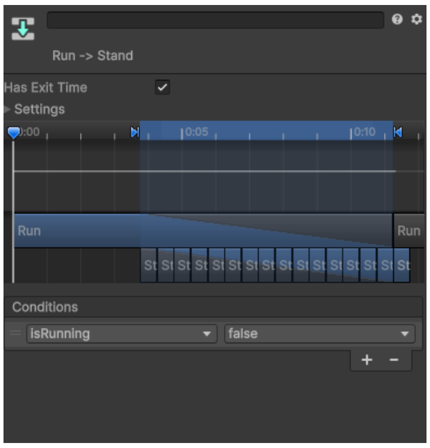
Another prerequisite is the animations. I will begin creating them now, starting with the standing animation



This is the process of creating the running animation. I separated them into one-second gaps to make it slower



This is the animation controller. I will add transitions between standing and running within the code, but there have to also be conditions within the controller to check through by the code


Run -> Stand

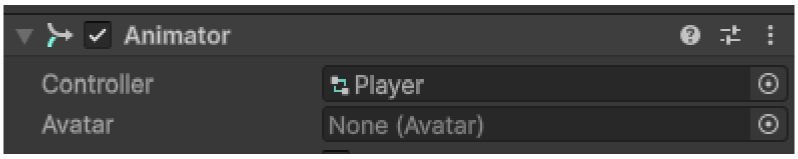
Has Exit Time ✓

Settings

Conditions

isRunning false

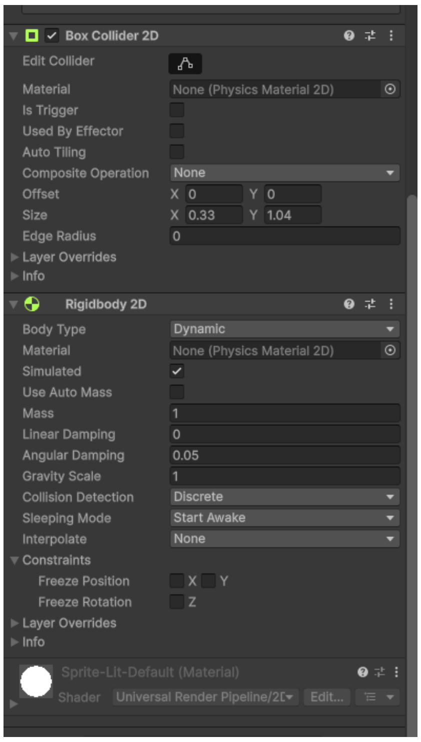
I have added the Boolean condition to the transition between running to standing. There is the opposite on the other transition


Animator

Controller Player

Avatar None (Avatar)

I then attach the controller to the animator in Player for the player to be able to play those animations


Box Collider 2D

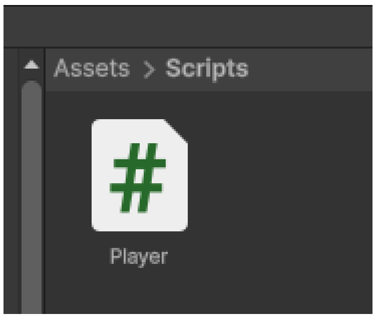
Edit Collider	
Material	None (Physics Material 2D)
Is Trigger	☐
Used By Effector	☐
Auto Tiling	☐
Composite Operation	None
Offset	X 0 Y 0
Size	X 0.33 Y 1.04
Edge Radius	0
Layer Overrides	
Info	

Rigidbody 2D

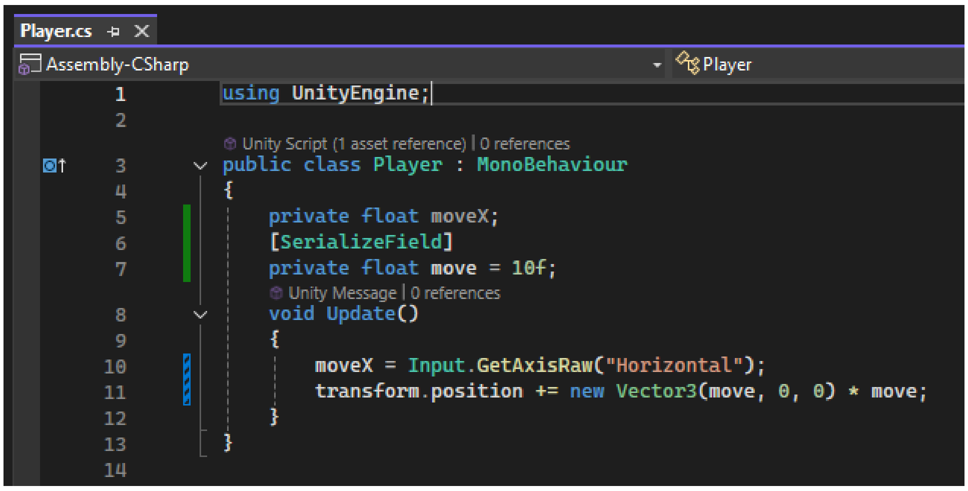
Body Type	Dynamic
Material	None (Physics Material 2D)
Simulated	✓
Use Auto Mass	☐
Mass	1
Linear Damping	0
Angular Damping	0.05
Gravity Scale	1
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None
Constraints	
Freeze Position	☐ X ☐ Y
Freeze Rotation	☐ Z
Layer Overrides	
Info	

Sprite-Lit-Default (Material)
Shader Universal Render Pipeline/2D Edit...

It is after this I add the other components to the player, making him able to fall and detect collision with the floor

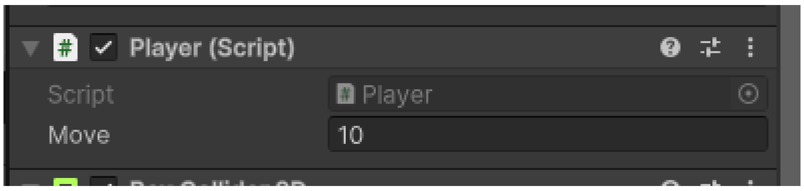


It is by this point I need to start creating the code for the player. I will start with the code for movement as it is the most basic part

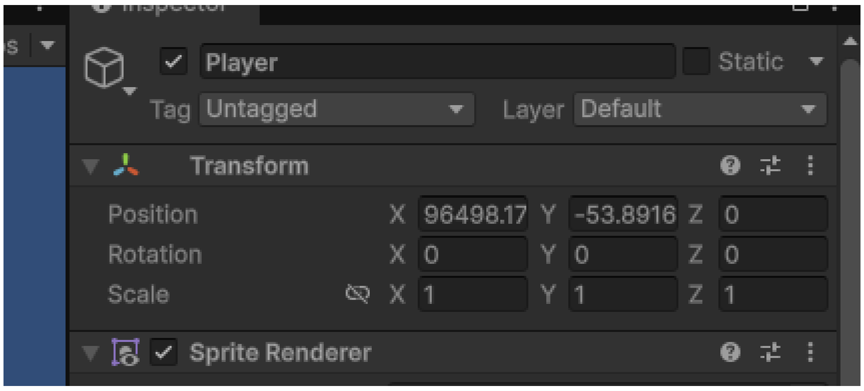


```csharp
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Player : MonoBehaviour
{
    private float moveX;
    [SerializeField]
    private float move = 10f;
    Unity Message | 0 references
    void Update()
    {
        moveX = Input.GetAxisRaw("Horizontal");
        transform.position += new Vector3(move, 0, 0) * move;
    }
}
```

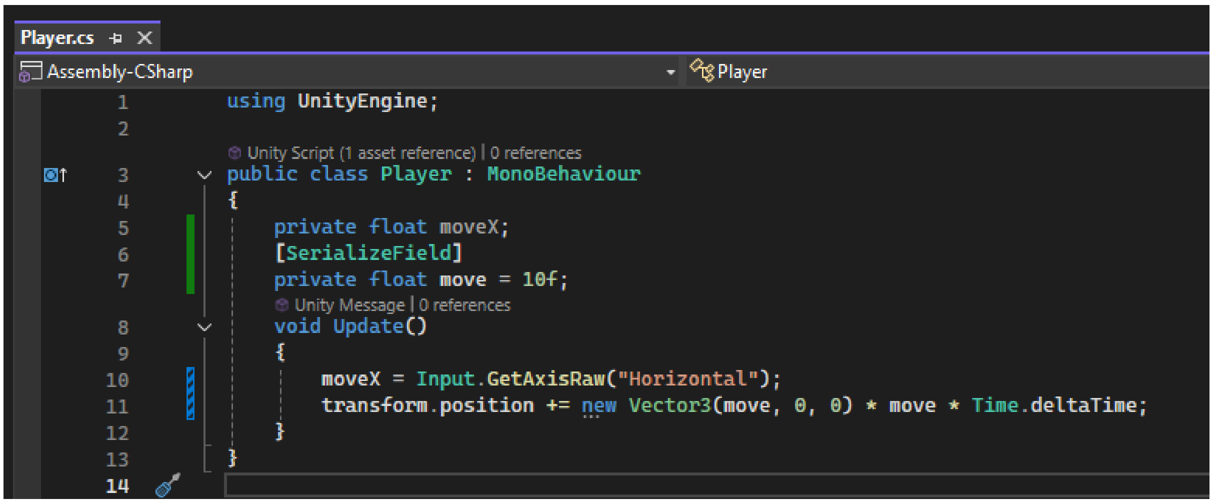
There were a few new additions to a newly created script, like the creation of two variables, one to hold input and the other the momentum. Those were both put into the update function which is called once a frame. Adding the vector to the position allows the player to move


Player (Script)
Script: Player
Move: 10

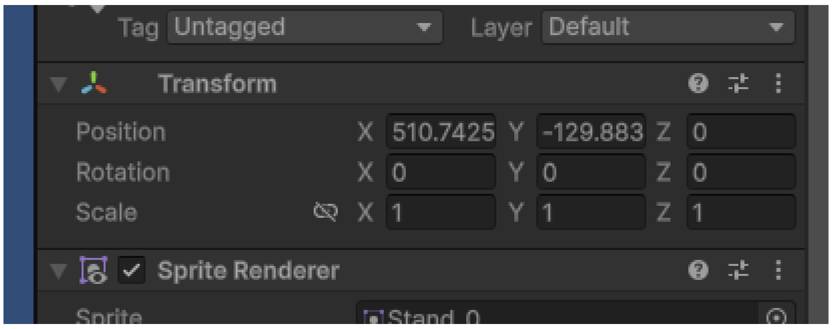
The script now is added to the player object. It can now move. However, this does not animate the player as changing the conditions have not been added to the code yet


Player
Tag: Untagged   Layer: Default
Transform
Position X 96498.17 Y -53.8916 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer

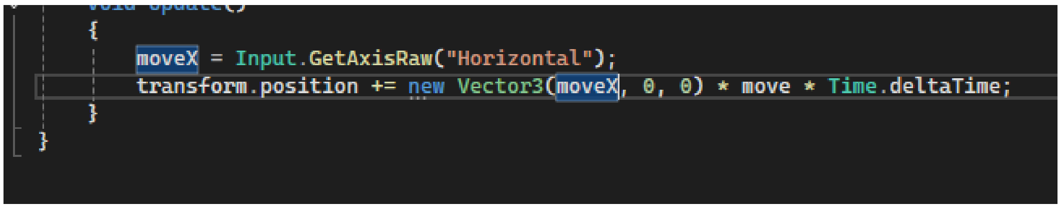
It is clear this script did not work


```
using UnityEngine;

public class Player : MonoBehaviour
{
    private float moveX;
    [SerializeField]
    private float move = 10f;
    void Update()
    {
        moveX = Input.GetAxisRaw("Horizontal");
        transform.position += new Vector3(move, 0, 0) * move * Time.deltaTime;
    }
}
```

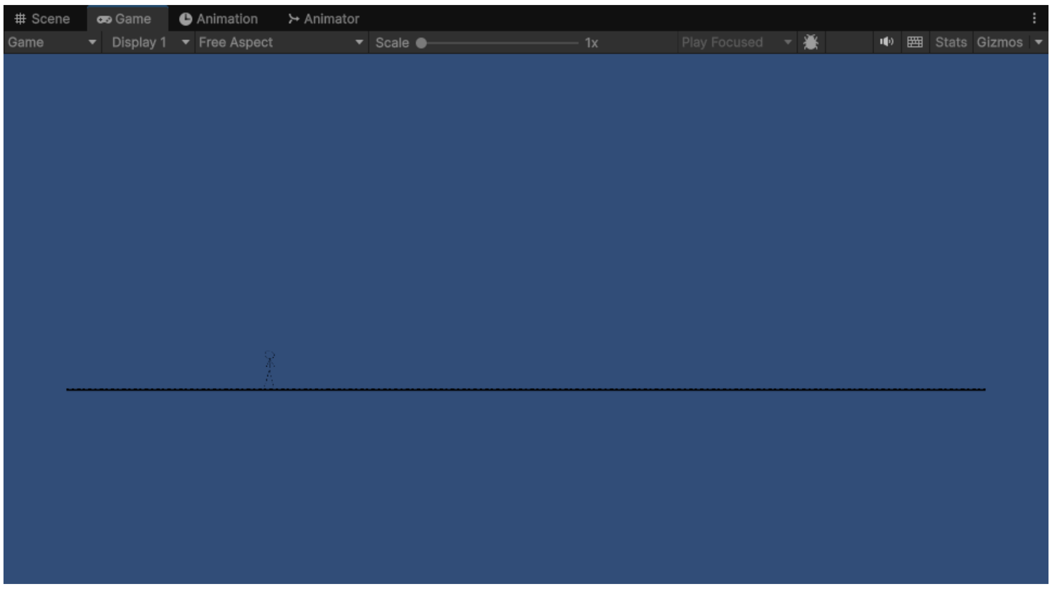
I will attempt to fix this by multiplying the vector by time as well, to slow it down


Tag: Untagged   Layer: Default
Transform
Position X 510.7425 Y -129.883 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer
Sprite: Stand_0

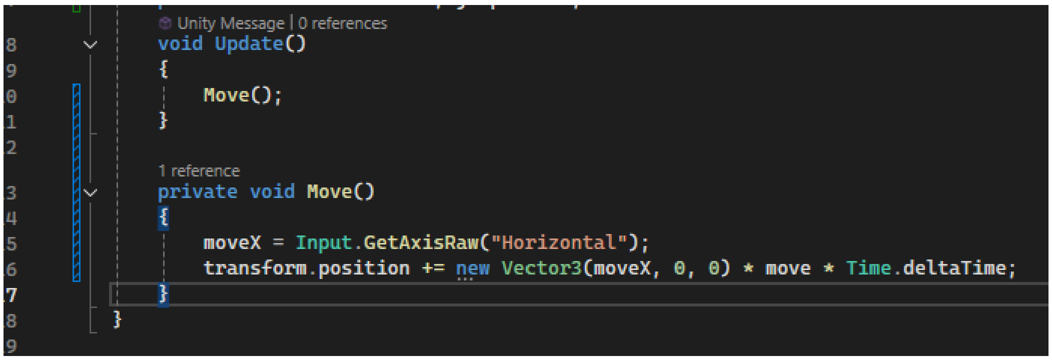
It is clear this change did not help


```
    {
        moveX = Input.GetAxisRaw("Horizontal");
        transform.position += new Vector3(moveX, 0, 0) * move * Time.deltaTime;
    }
}
```

I will try and fix it again by instead detecting if it has had an input instead of a constant input

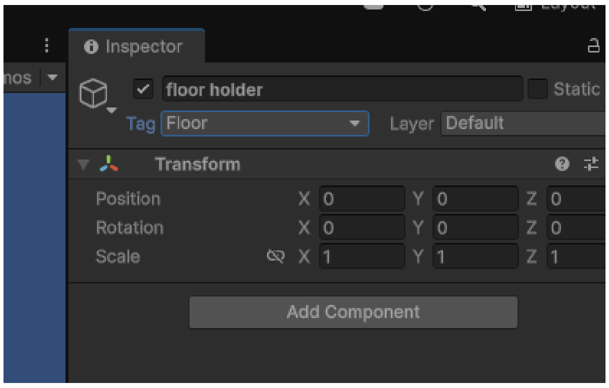


This change had worked, and the player can now move across the floor without resistance or change. I plan now on adding jumping to the player



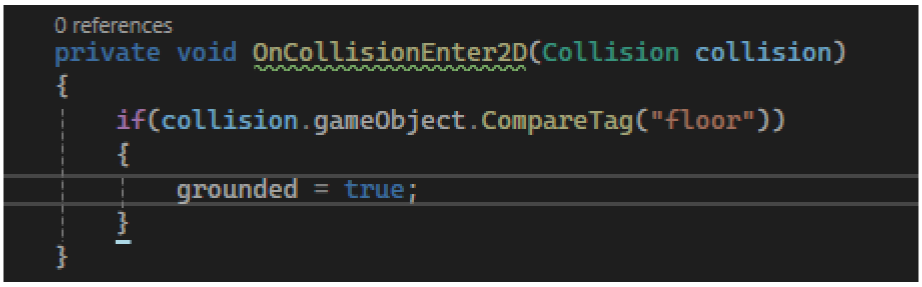
```
    Unity Message | 0 references
    void Update()
    {
        Move();
    }

    1 reference
    private void Move()
    {
        moveX = Input.GetAxisRaw("Horizontal");
        transform.position += new Vector3(moveX, 0, 0) * move * Time.deltaTime;
    }
}
```

It is also during this time I have changed the move code to a function for easier readability in Update



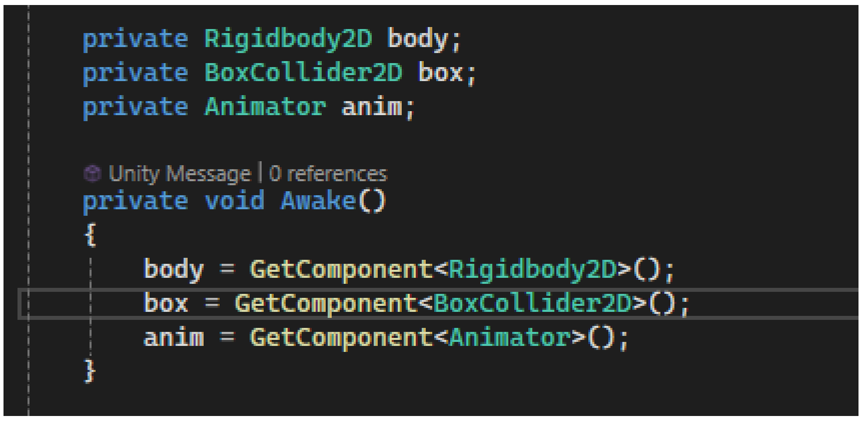
Inspector
floor holder
Static
Tag Floor
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Add Component

In preparation for jumping, it is required of me to add the floor tag to each floor object. This is so that the player can now compare the tag whenever it collides with an object



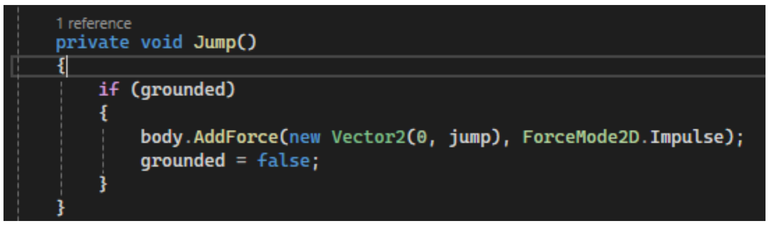
```
0 references
private void OnCollisionEnter2D(Collision collision)
{
    if(collision.gameObject.CompareTag("floor"))
    {
        grounded = true;
    }
}
```

This is the detection part to see if player as touched the floor. If they have, it assigns the new variable grounded to true. That will later be changed to false when the player jumps



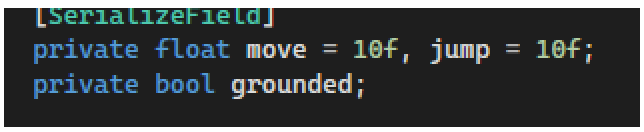
```
private Rigidbody2D body;
private BoxCollider2D box;
private Animator anim;

Unity Message | 0 references
private void Awake()
{
    body = GetComponent<Rigidbody2D>();
    box = GetComponent<BoxCollider2D>();
    anim = GetComponent<Animator>();
}
```

It is again during this time that I need to add references to each essential component to player in order to instantiate their effects on the player. This will be seen in a moment



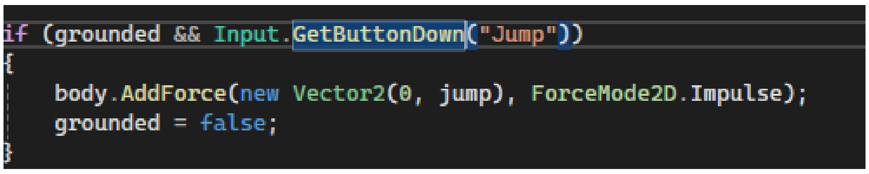
```
1 reference
private void Jump()
{
    if (grounded)
    {
        body.AddForce(new Vector2(0, jump), ForceMode2D.Impulse);
        grounded = false;
    }
}
```

The code for Jump adds upwards force to the player, propelling them upwards and creating the effect of jumping. The forcemode2d is just a part of addforce. It is after that it then sets grounded to false, making it so that the player cannot jump again until they are grounded again



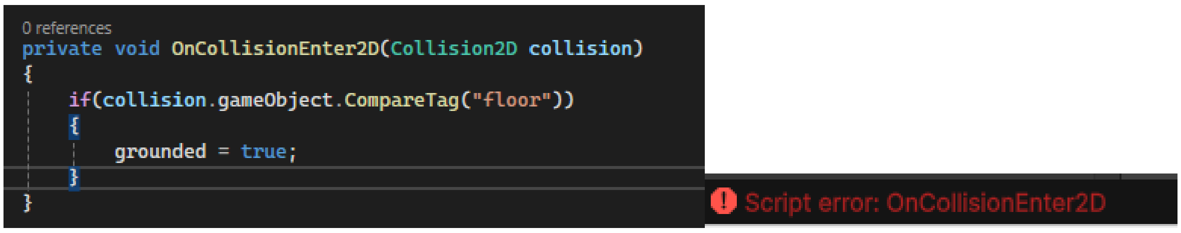
```
[SerializeField]
private float move = 10f, jump = 10f;
private bool grounded;
```

The newly created jump variable is under the serialisefield, allowing me to edit the value from unity



```
if (grounded && Input.GetButtonDown("Jump"))
{
    body.AddForce(new Vector2(0, jump), ForceMode2D.Impulse);
    grounded = false;
}
```

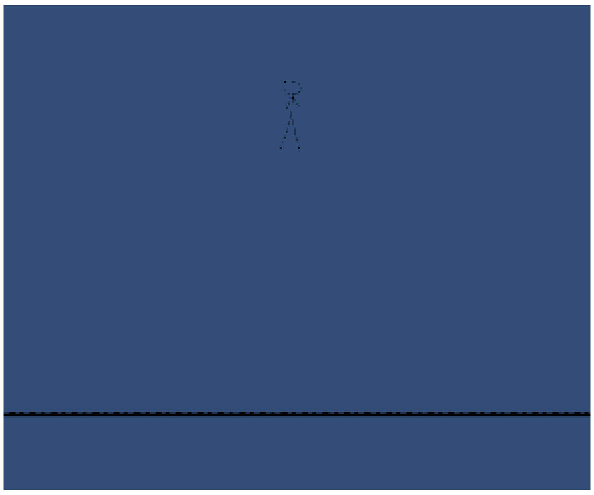
Despite this, I neglected to add the input to this, making it impossible until now to actually jump, since there was no detection



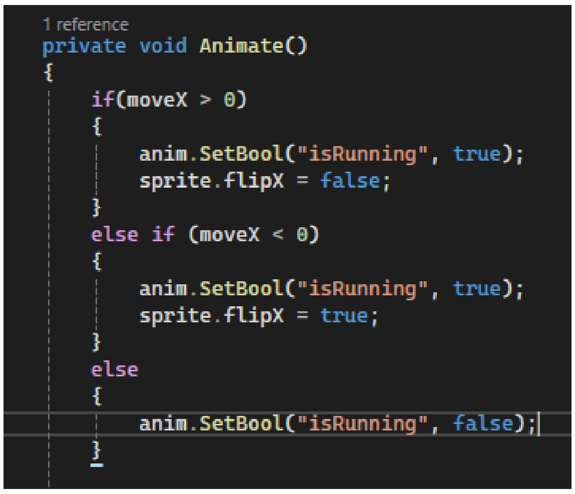
```
0 references
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.CompareTag("floor"))
    {
        grounded = true;
    }
}
```

```
Script error: OnCollisionEnter2D
```

There was also the fact that oncollisionenter2d was checking for a 3d collision, therefore would not assign grounded to true ever until now

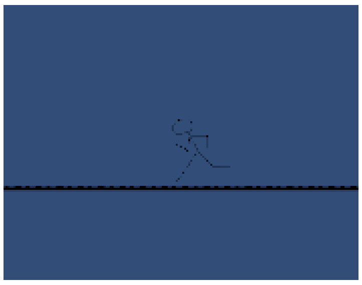


The player can now jump. It is now I plan on adding animations

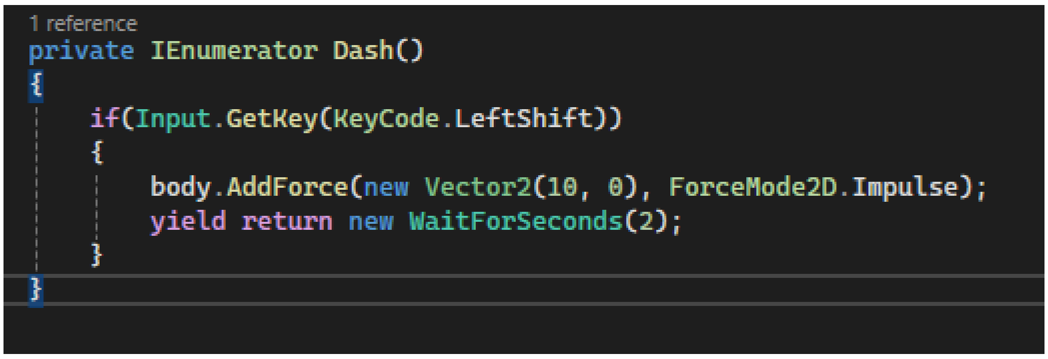


```
1 reference
private void Animate()
{
    if(moveX > 0)
    {
        anim.SetBool("isRunning", true);
        sprite.flipX = false;
    }
    else if (moveX < 0)
    {
        anim.SetBool("isRunning", true);
        sprite.flipX = true;
    }
    else
    {
        anim.SetBool("isRunning", false);
    }
```

With the animator, player can either be running left, right or standing still. These are the three conditions shown in the code. flipX reverses the sprite so it will look like the sprite is running in the right direction and not running backwards



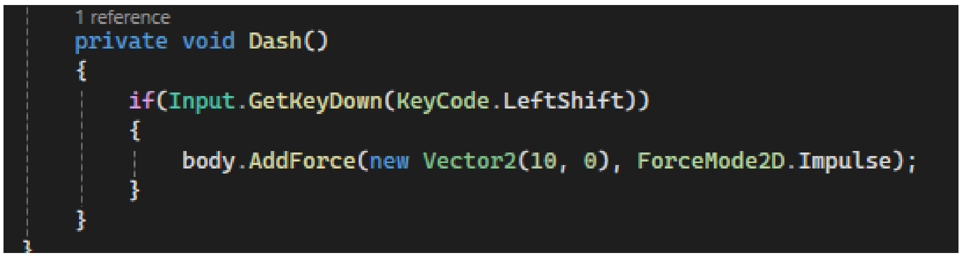
The player is now animated. I plan now on adding dashing



```
1 reference
private IEnumerator Dash()
{
    if(Input.GetKey(KeyCode.LeftShift))
    {
        body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
        yield return new WaitForSeconds(2);
    }
}
```

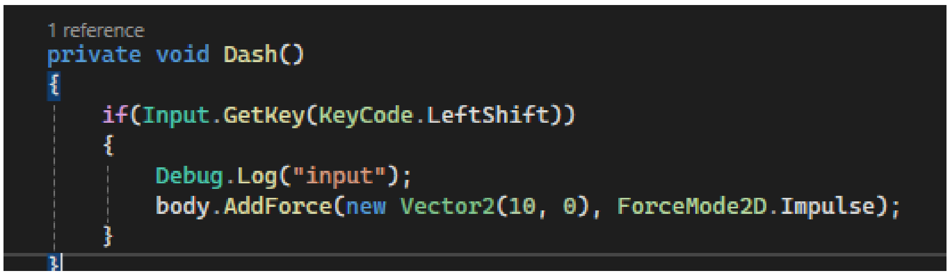
The dash code is an IEnumerator as it uses a return statement from waitforseconds, as it needs a cooldown. It also adds force to the player to make them move instantly and fast on input

- It did not work



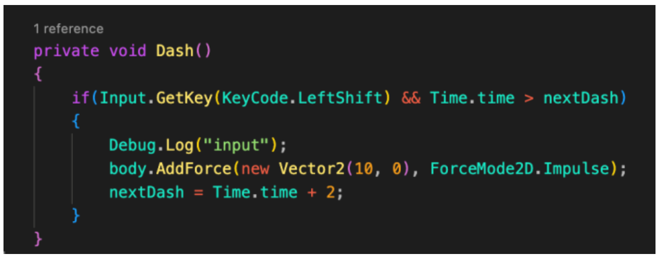
```
1 reference
private void Dash()
{
    if(Input.GetKeyDown(KeyCode.LeftShift))
    {
        body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
    }
}
```

I attempted to fix it, it only dashed once despite input being detected on subsequent attempts



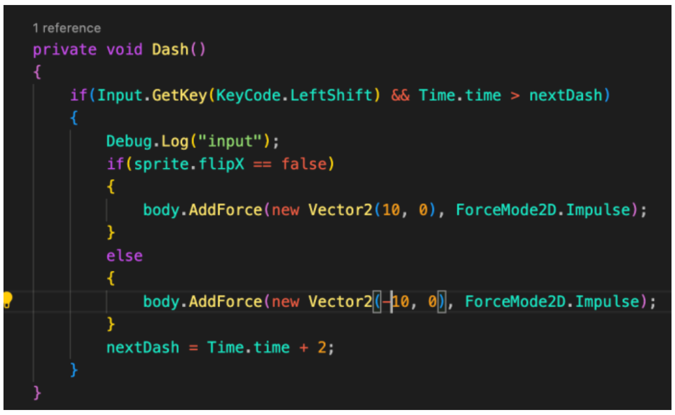
```
1 reference
private void Dash()
{
    if(Input.GetKey(KeyCode.LeftShift))
    {
        Debug.Log("input");
        body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
    }
}
```

It instead turns out that getkeydown only detects one input, if you instead use getkey it will detect all inputs



```
1 reference
private void Dash()
{
    if(Input.GetKey(KeyCode.LeftShift) && Time.time > nextDash)
    {
        Debug.Log("input");
        body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
        nextDash = Time.time + 2;
    }
}
```

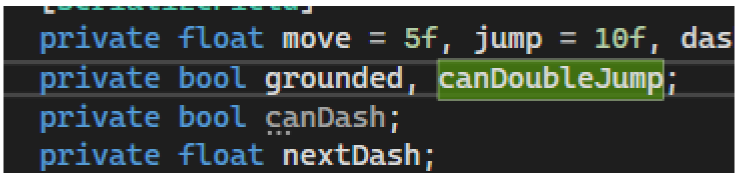
Adding a cooldown to dashing allows the player not to constantly accelerate when holding shift



```
1 reference
private void Dash()
{
    if(Input.GetKey(KeyCode.LeftShift) && Time.time > nextDash)
    {
        Debug.Log("input");
        if(sprite.flipX == false)
        {
            body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
        }
        else
        {
            body.AddForce(new Vector2(-10, 0), ForceMode2D.Impulse);
        }
        nextDash = Time.time + 2;
    }
}
```

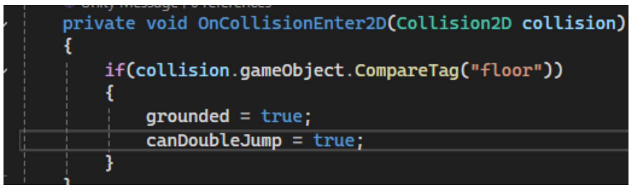
This iteration allows the player to dash in the direction they face, improving on the existing code

- Cooldown works



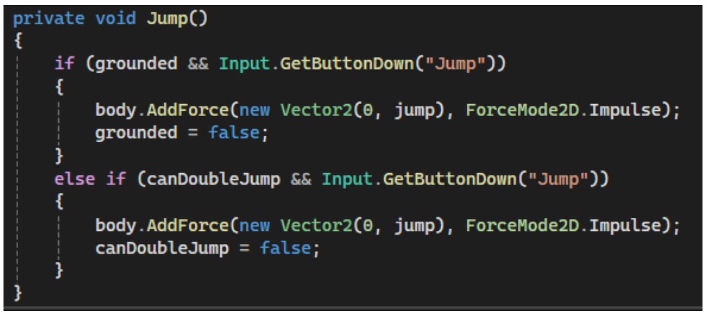
```
private float move = 5f, jump = 10f, das
private bool grounded, canDoubleJump;
private bool canDash;
private float nextDash;
```

Additionally I have decided to add a double-jump to the player for fun



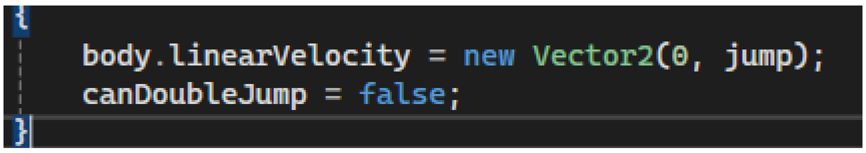
```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.CompareTag("floor"))
    {
        grounded = true;
        canDoubleJump = true;
    }
```

I added the check to recharge it when you touch the ground for it's cooldown



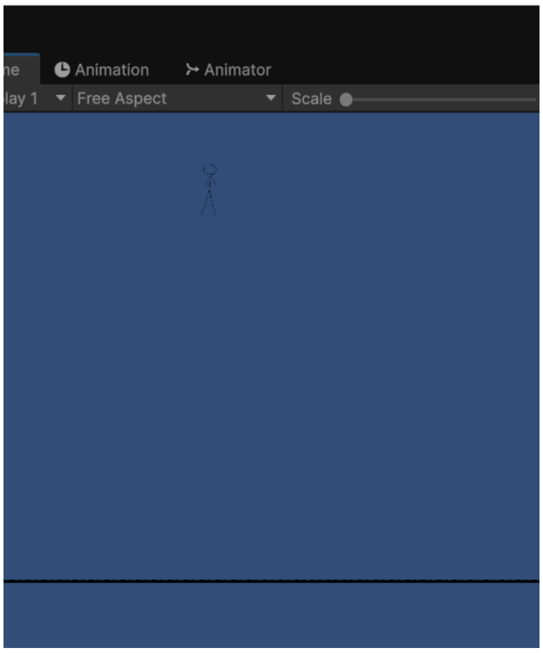
```
private void Jump()
{
    if (grounded && Input.GetButtonDown("Jump"))
    {
        body.AddForce(new Vector2(0, jump), ForceMode2D.Impulse);
        grounded = false;
    }
    else if (canDoubleJump && Input.GetButtonDown("Jump"))
    {
        body.AddForce(new Vector2(0, jump), ForceMode2D.Impulse);
        canDoubleJump = false;
    }
}
```

It is after this I add the additional statement to the jumping code to check if it has double-jumped or not, and only if it has already jumped once



```
{
    body.linearVelocity = new Vector2(0, jump);
    canDoubleJump = false;
}
```

Instead of this, I decided to use linear velocity as it cancels out all vertical movement. If It were not this, then if the player were moving downwards at the same rate of the jump force, the vertical movement would cancel out and not perform a jump



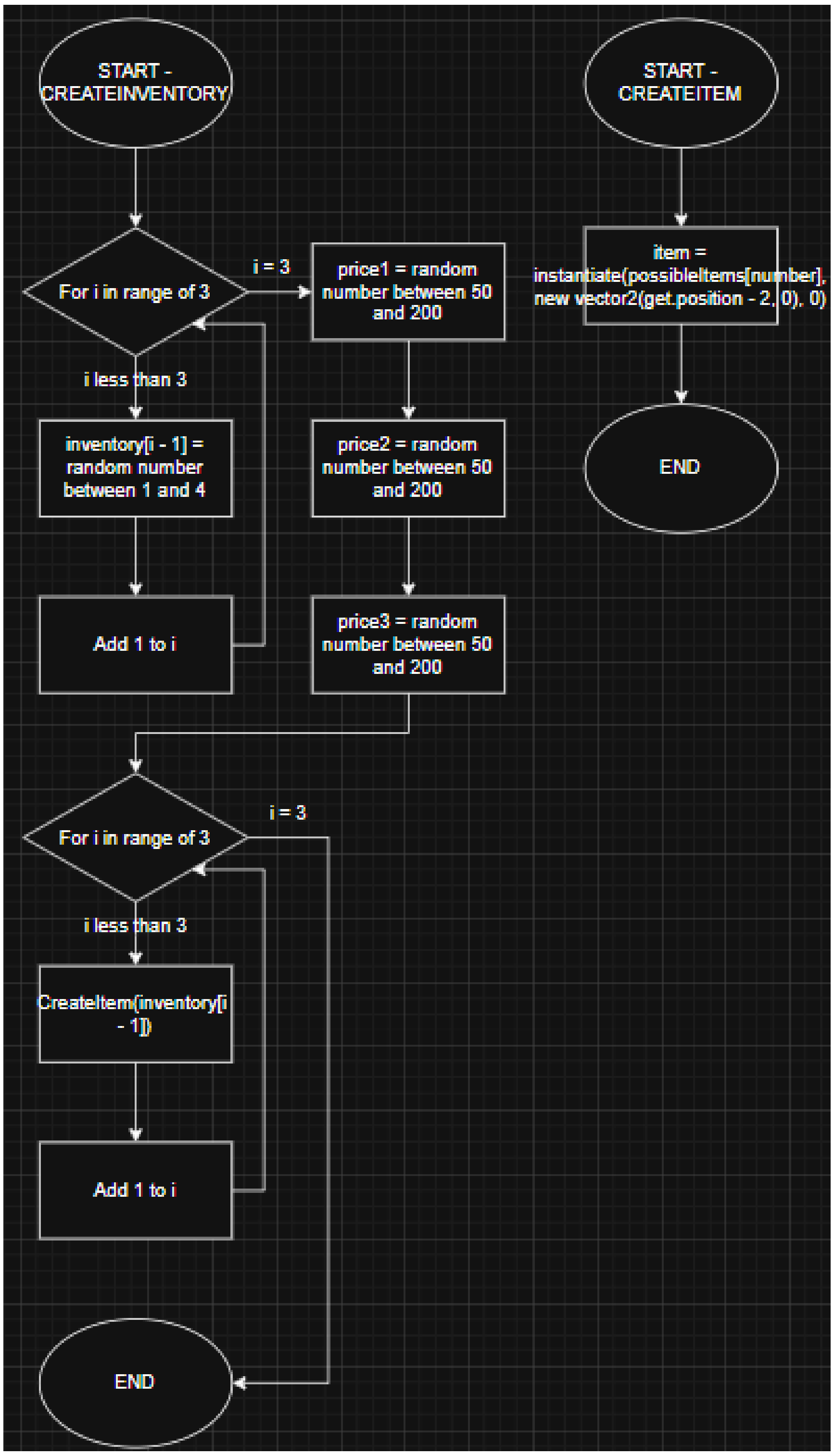
Double jumping now works – the player cannot usually jump this high if not for a double jump

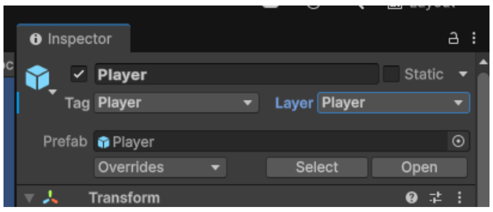
Test No.	Test	Expected outcome	Success?	Outcome	Screenshots
1	Movement – checking to see if input will make the character move	The player moves horizontally across the screen	YES	The player can slide across the floor with gravity	
2	Jumping – to check on input from spacebar if the player jumps	The player performs a jump	YES	The player can now move upwards after being connected with the floor	
3	Dashing – to check on shift key input if the player performs a dash	The player performs a dash	YES	The player can increase their horizontal movement temporarily	
4	Animation – to play animations when a player does something	The player performs an animation	YES	The player can play their animations on certain conditions	

Part one complete

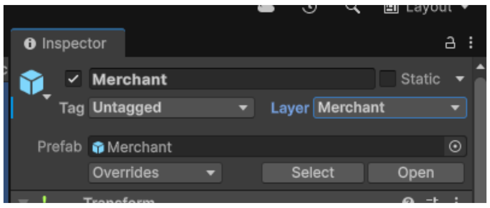
Part 2 – shops and items

Merchant
-Inventory: array -possibleItems: array -itemRef: GameObject -item: GameObject -index: int -prices: array
-Start: void -MakeItem: void

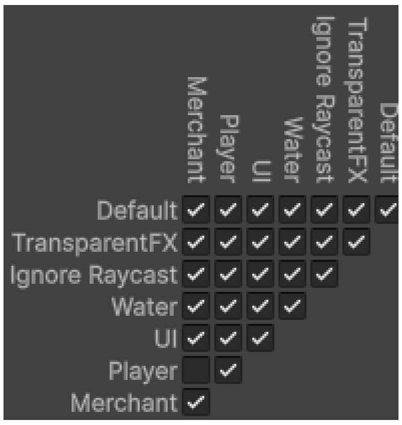

START - CREATEINVENTORY
For i in range of 3
i = 3
price1 = random number between 50 and 200
i less than 3
inventory[i - 1] = random number between 1 and 4
price2 = random number between 50 and 200
Add 1 to i
price3 = random number between 50 and 200
For i in range of 3
i = 3
i less than 3
CreateItem(inventory[i - 1])
Add 1 to i
END
START - CREATEITEM
item = instantiate(possibleItems[number], new vector2(get.position - 2, 0), 0)
END



Player — Static
Tag Player — Layer Player
Prefab Player
Overrides — Select — Open
Transform

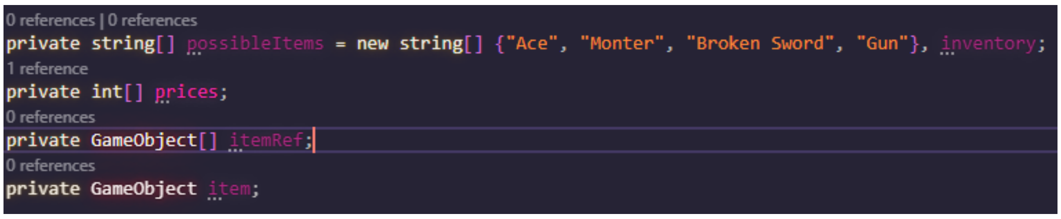


Merchant — Static
Tag Untagged — Layer Merchant
Prefab Merchant
Overrides — Select — Open



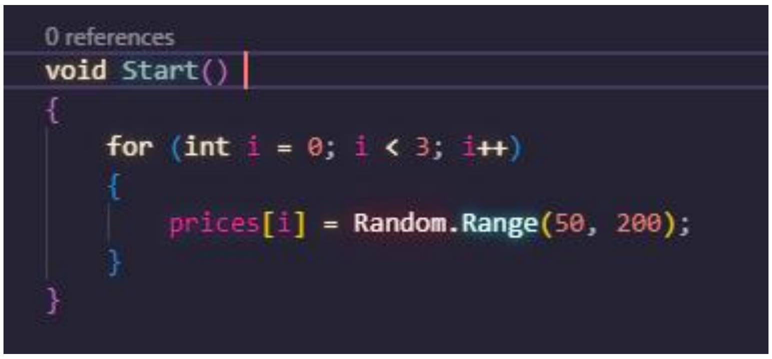
	Merchant	Player	UI	Water	Ignore Raycast	TransparentFX	Default
Default	✓	✓	✓	✓	✓	✓	✓
TransparentFX	✓	✓	✓	✓	✓	✓	
Ignore Raycast	✓	✓	✓	✓	✓		
Water	✓	✓	✓	✓			
UI	✓	✓	✓				
Player		✓					
Merchant	✓						

To begin with this I removed the collision between the player and the merchant, as the merchant is going to be in the background and I do not want the player to be running into them



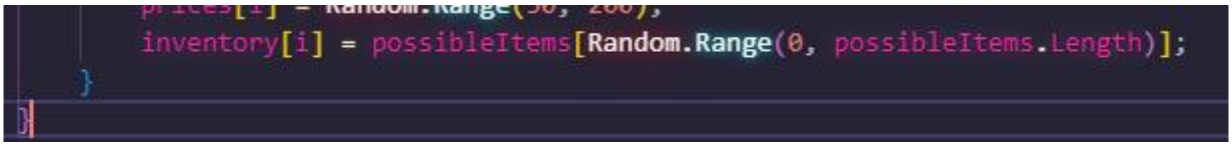
```
0 references | 0 references
private string[] possibleItems = new string[] {"Ace", "Monter", "Broken Sword", "Gun"}, inventory;
1 reference
private int[] prices;
0 references
private GameObject[] itemRef;
0 references
private GameObject item;
```

It is now I add the variables. These variables will soon be added to but for now they are the only necessary components so far



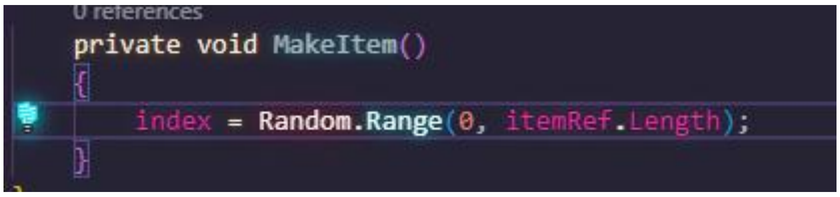
```
0 references
void Start()
{
    for (int i = 0; i < 3; i++)
    {
        prices[i] = Random.Range(50, 200);
    }
}
```

I then added random prices to the three values. I decided on the range 50 – 200 as the amount of money I plan on having the enemies drop is around this value



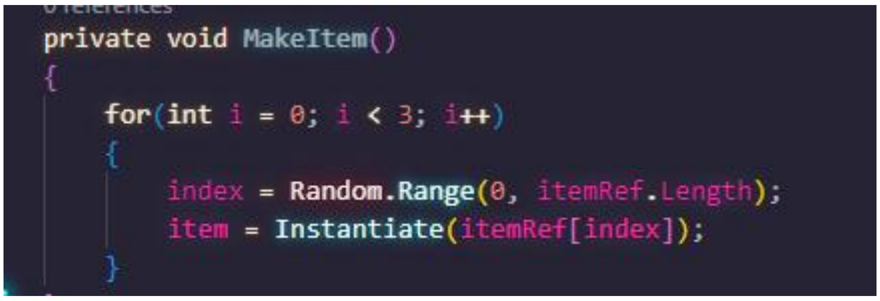
```
        inventory[i] = possibleItems[Random.Range(0, possibleItems.Length)];
    }
}
```

After this I add the inventory of the merchant. It will get three random items as it is in the same loop as prices. It also gets the items from the possible list for ease of access and simpler code (I would have assigned each number to an item and made a function for taking a random item and going through the statements if not for this code)



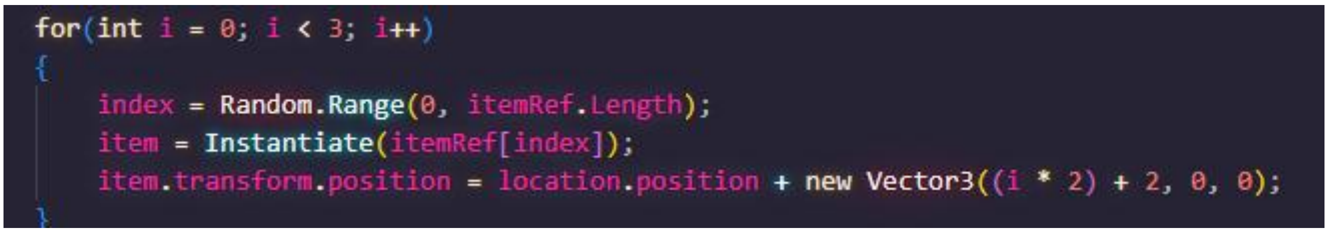
```
0 references
private void MakeItem()
{
    index = Random.Range(0, itemRef.Length);
}
```

Now I add the randomization not the items when they are being made. I also in this part create the MakeItem function. The MakeItem function will be responsible for creating the items for the merchant



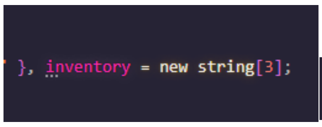
```
private void MakeItem()
{
    for(int i = 0; i < 3; i++)
    {
        index = Random.Range(0, itemRef.Length);
        item = Instantiate(itemRef[index]);
    }
```

Now it has the code to create an object. This is put in this function MakeItem as it makes the item

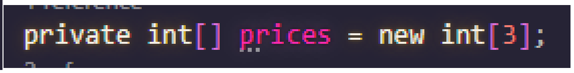


```
for(int i = 0; i < 3; i++)
{
    index = Random.Range(0, itemRef.Length);
    item = Instantiate(itemRef[index]);
    item.transform.position = location.position + new Vector3((i * 2) + 2, 0, 0);
}
```

I now add the code to move the spawned object to the side of the merchant. I put this in for the players to be able to see the merchant before they buy their items

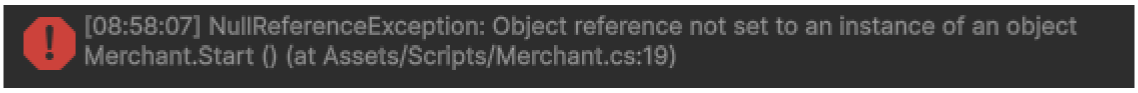


```
}, inventory = new string[3];
```



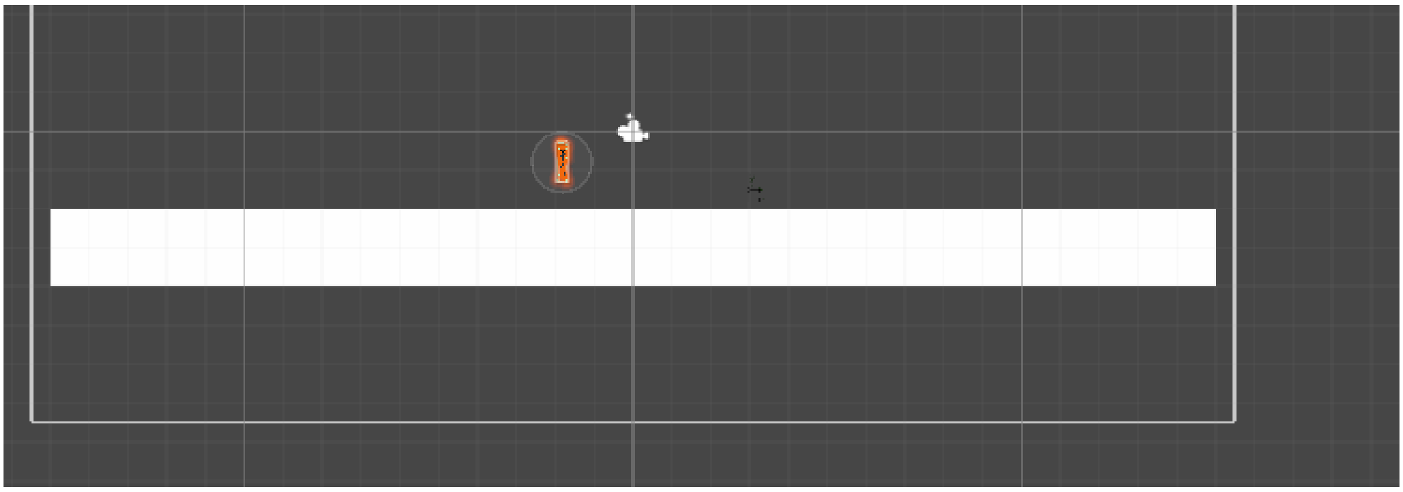
```
private int[] prices = new int[3];
```

This addition to the code fixes the error:

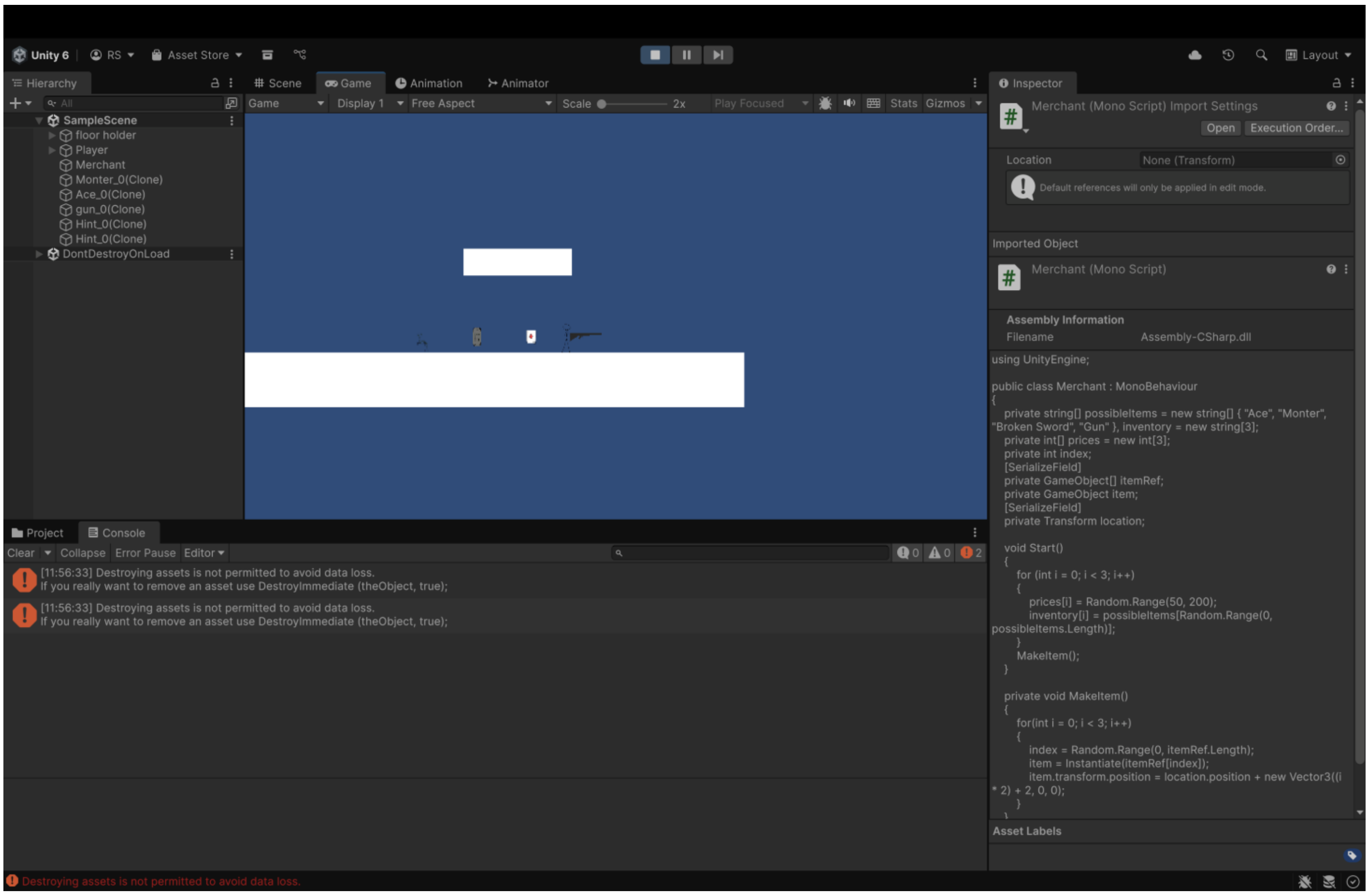


```
[08:58:07] NullReferenceException: Object reference not set to an instance of an object
Merchant.Start () (at Assets/Scripts/Merchant.cs:19)
```

This is because they do not have a set length yet, making the code not run and stop before the part where it creates the objects, nor did it assign the prices or inventory positions



I also revamped the platform as I was not satisfied with the other one. This was because on some tests the player would fall through the old platform and stop the test



Location: None (Transform)

Default references will only be applied in edit mode.

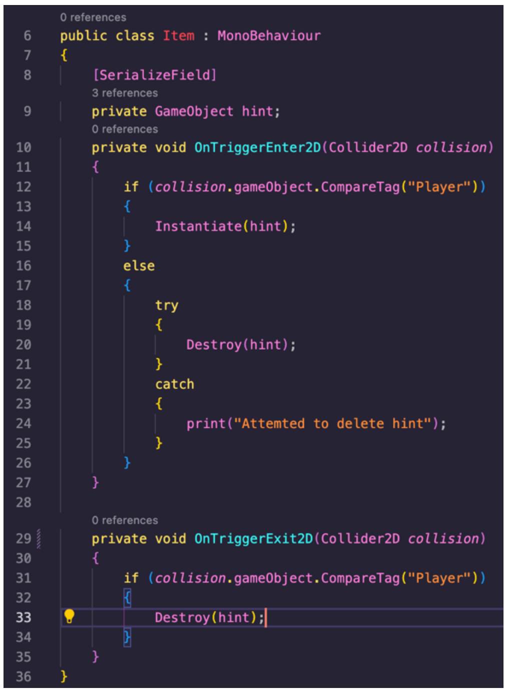
Assembly Information
Filename: Assembly-CSharp.dll

```
using UnityEngine;

public class Merchant : MonoBehaviour
{
    private string[] possibleItems = new string[] { "Ace", "Monter",
"Broken Sword", "Gun" }, inventory = new string[3];
    private int[] prices = new int[3];
    private int index;
    [SerializeField]
    private GameObject[] itemRef;
    private GameObject item;
    [SerializeField]
    private Transform location;

    void Start()
    {
        for (int i = 0; i < 3; i++)
        {
            prices[i] = Random.Range(50, 200);
            inventory[i] = possibleItems[Random.Range(0,
possibleItems.Length)];
        }
        MakeItem();
    }

    private void MakeItem()
    {
        for(int i = 0; i < 3; i++)
        {
            index = Random.Range(0, itemRef.Length);
            item = Instantiate(itemRef[index]);
            item.transform.position = location.position + new Vector3((i
* 2) + 2, 0, 0);
        }
```

```
[11:56:33] Destroying assets is not permitted to avoid data loss.
If you really want to remove an asset use DestroyImmediate (theObject, true);
[11:56:33] Destroying assets is not permitted to avoid data loss.
If you really want to remove an asset use DestroyImmediate (theObject, true);
```

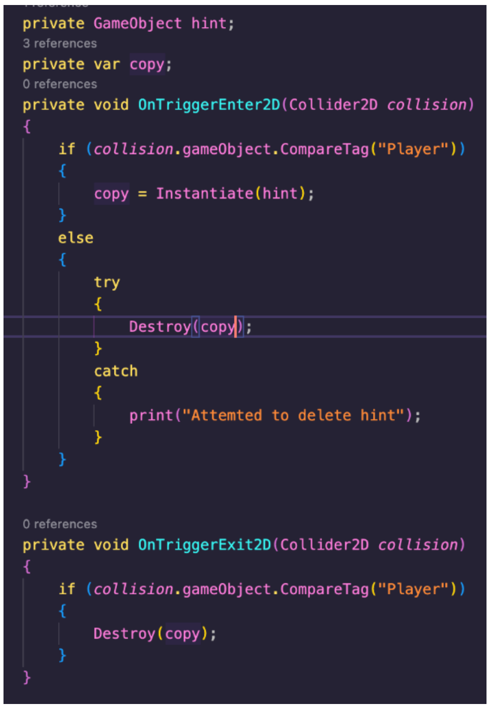
```
Destroying assets is not permitted to avoid data loss.
```

On adding collision detection to the items, I decided to add a hint on what to input when the player is colliding with the item. When the player does so, I wanted a hint to appear saying that the player must press “E” to pick up the item. This is because running into the item in a shop and automatically buying it just because you ran past is not good design for my game, as money will be limited



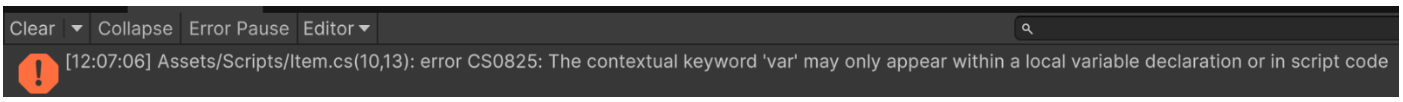
```
0 references
public class Item : MonoBehaviour
{
    [SerializeField]
    3 references
    private GameObject hint;
    0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            Instantiate(hint);
        }
        else
        {
            try
            {
                Destroy(hint);
            }
            catch
            {
                print("Attemted to delete hint");
            }
        }
    }

    0 references
    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            Destroy(hint);
        }
    }
}
```

This was my brief attempt at creating this. I decided to make it an object to instantiate, and when the player stops colliding with the object the hint will disappear, like a proximity trigger. I designed it this way because I thought it would be the easiest way to implement my idea. This instead caused errors to pop up, seen in the previous screenshot, showing that this did not work



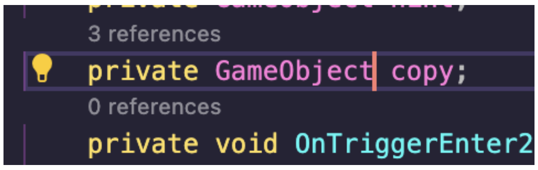
```
private GameObject hint;
3 references
private var copy;
0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Player"))
    {
        copy = Instantiate(hint);
    }
    else
    {
        try
        {
            Destroy(copy);
        }
        catch
        {
            print("Attemted to delete hint");
        }
    }
}

0 references
private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Player"))
    {
        Destroy(copy);
    }
}
```

Upon further research it seems that I was trying to destroy the prefab of the hint, not the instantiated object. This could be the cause for the error



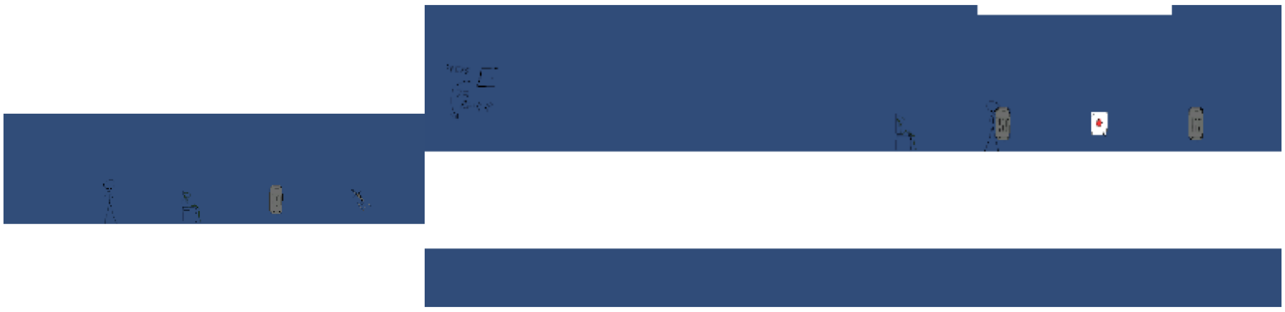
```
[12:07:06] Assets/Scripts/Item.cs(10,13): error CS0825: The contextual keyword 'var' may only appear within a local variable declaration or in script code
```

It turns out that having var as the type for the copy holder causes an error as var can only be used to define local variables. I do have another way to try and work past this

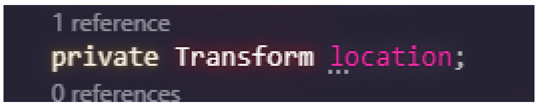


```
3 references
private GameObject copy;
0 references
private void OnTriggerEnter2
```

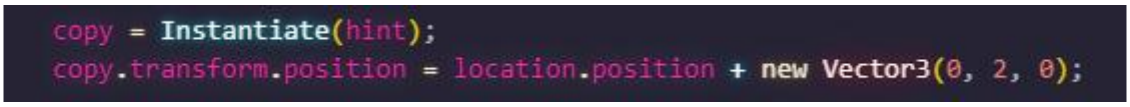
This idea makes copy a gameobject, allowing me to, instead of using a var and the code falling apart, have it enter as a gameobject, which is likely to fix this error. I believe I used var wrong



As it can be seen in the two screenshots, it shows one where the player is not colliding with the items and it does not exist, an the other where it does. This has fixed the problem

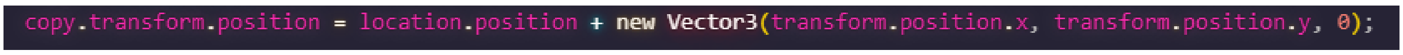


```
1 reference
private Transform location;
0 references
```



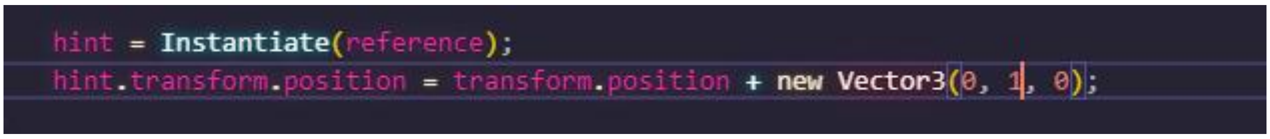
```
copy = Instantiate(hint);
copy.transform.position = location.position + new Vector3(0, 2, 0);
```

For this hint, I needed to make the hint appear above the items to show that you are able to pick them up



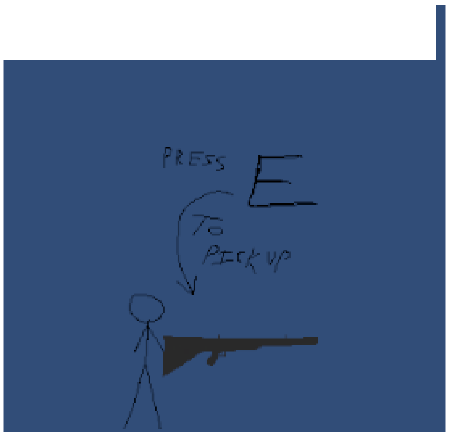
```
copy.transform.position = location.position + new Vector3(transform.position.x, transform.position.y, 0);
```

Instead of working as intended, it stayed in the same position. This code will attempt to fix it



```
hint = Instantiate(reference);
hint.transform.position = transform.position + new Vector3(0, 1, 0);
```

My code, instead of moving the hint, it was trying to return the value of the location, causing an error. It works now



The location now hovers above the item

```
GameObject player = GameObject.Find("Player");
PlayerScript script = player.GetComponent<PlayerScript>();
```

I needed to get a reference to both the player and the script in the player. The player gameobject finds the player object in the scene, and the script component gets a reference to the script on player. This allows me to access all functions in that script

```
script.addToInventory(name);
```

This code is an example of this, by calling the script from item it can add it’s name to the player inventory

```
4 references | 2 references
private int canDoubleJump, maxDoubleJump = 1;
```

In player, I decided to rework the existing double jumping script. This is because I plan on having an item give you an additional double jump.

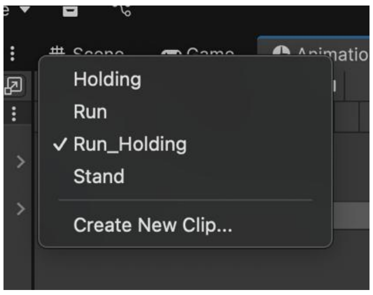
```
else if (canDoubleJump > 0 && Input.GetButtonDown("Jump"))
{
    body.linearVelocity = new Vector2(0, jump);
    canDoubleJump -= 1;
}
```

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.CompareTag("floor"))
    {
        grounded = true;
        canDoubleJump = maxDoubleJump;
    }
}
```

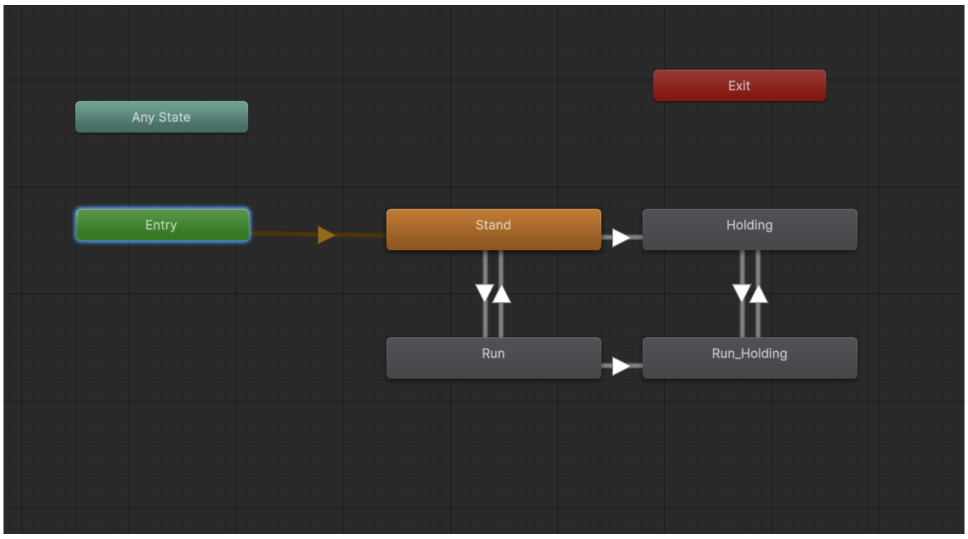
This can be seen in the reworked code. Instead of setting it to false it instead takes one away until you are out of jumps. It then resets on the next contact with the floor. This raises an interesting problem, you only get the next jump when you next collide with the floor, and not if you are already on the floor. This was fixed by the following:

```
else if(item == "Monter_0(Clone)")
{
    maxDoubleJump += 1;
    canDoubleJump += 1;
}
```

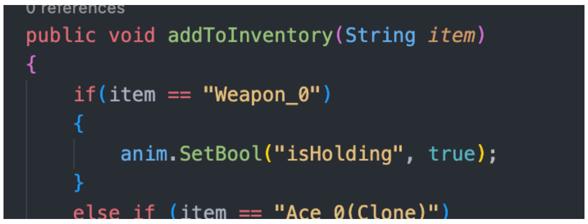
On picking up the item it gives the player both an additional max jump but another jump before, allowing them to get their jump immediately



It was also this time I realised that I had not made the attacking code for the player. First of all, I needed to create animations for the player holding weapons, so I made holding as the standing animation, and run_holding as the run animation when holding a weapon

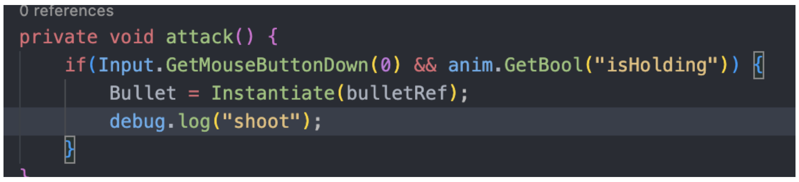


I added the new animations to the controller, as once you pick up the gun you cannot drop it, therefore there is no need for backwards connections between holding the gun and not. The transitions between standing and running when holding are the same as stand and run



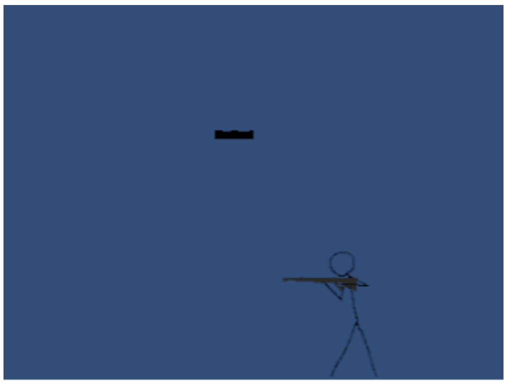
```
0 references
public void addToInventory(String item)
{
    if(item == "Weapon_0")
    {
        anim.SetBool("isHolding", true);
    }
    else if (item == "Ace 0(Clone)")
```

I then added the check in addToInventory to detect whether or not the player has picked up the gun. If they had, then it sets the check in the animation controller to true to transition between not holding to holding

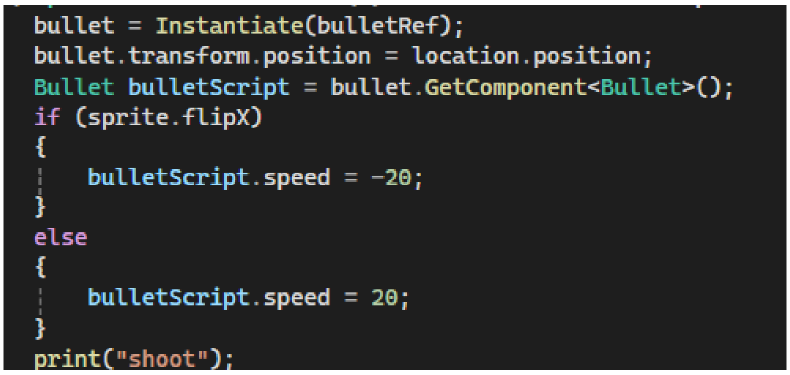


```
0 references
private void attack() {
    if(Input.GetMouseButtonDown(0) && anim.GetBool("isHolding")) {
        Bullet = Instantiate(bulletRef);
        debug.log("shoot");
    }
}
```

Lastly, I needed to test whether or not I was able to fire the weapon. It would instantiate a bullet to fire on mouse input. I will later need to add speed and location to the bullet on spawn for it to behave like a bullet

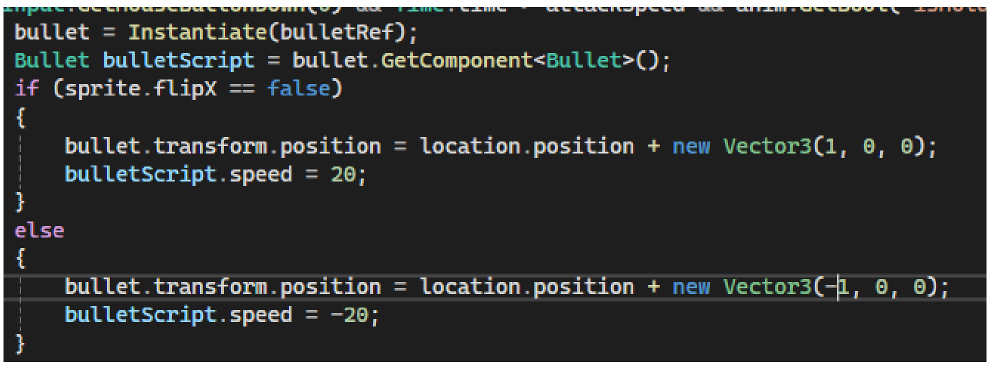


The code now creates the bullet. I now need to make the bullet not only spawn at the player location but also have it move



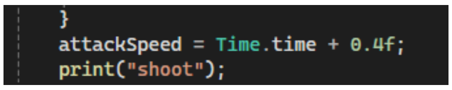
```
bullet = Instantiate(bulletRef);
bullet.transform.position = location.position;
Bullet bulletScript = bullet.GetComponent<Bullet>();
if (sprite.flipX)
{
    bulletScript.speed = -20;
}
else
{
    bulletScript.speed = 20;
}
print("shoot");
```

First of all, I added the bullet moving to the location of the player. I then got a reference to the script in bullet. This lets me access all public variables in the script, which let me change the speed of the bullet to 20. I then checked to see if the player sprite was flipped or not. If it was, then it would make the bullet move to the right, and if not, it would move to the left.



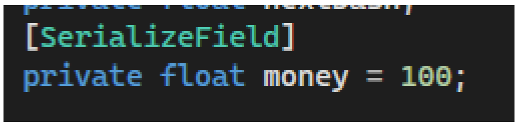
```
bullet = Instantiate(bulletRef);
Bullet bulletScript = bullet.GetComponent<Bullet>();
if (sprite.flipX == false)
{
    bullet.transform.position = location.position + new Vector3(1, 0, 0);
    bulletScript.speed = 20;
}
else
{
    bullet.transform.position = location.position + new Vector3(-1, 0, 0);
    bulletScript.speed = -20;
}
```

I changed the code to create the bullet on either side of the player in order for the bullet not to push the player when it is created



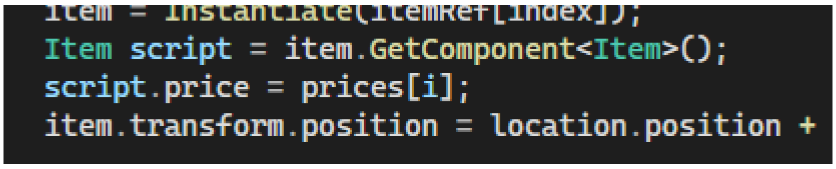
```
    }
    attackSpeed = Time.time + 0.4f;
    print("shoot");
```

I then added a cooldown to shooting in order to keep it fair and for players not to rely on spam-clicking



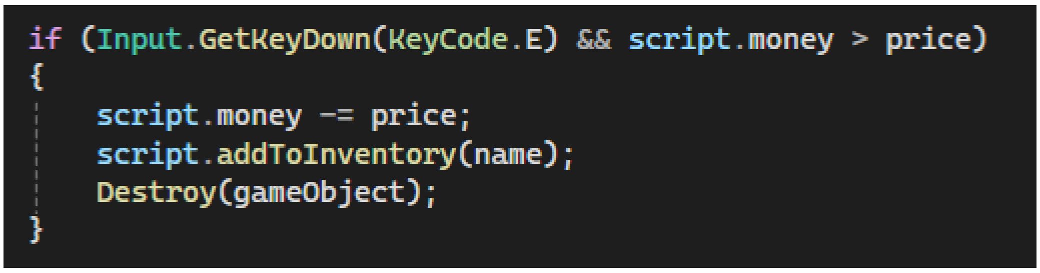
```
[SerializeField]
private float money = 100;
```

After this it is now time to work on prices for the items.



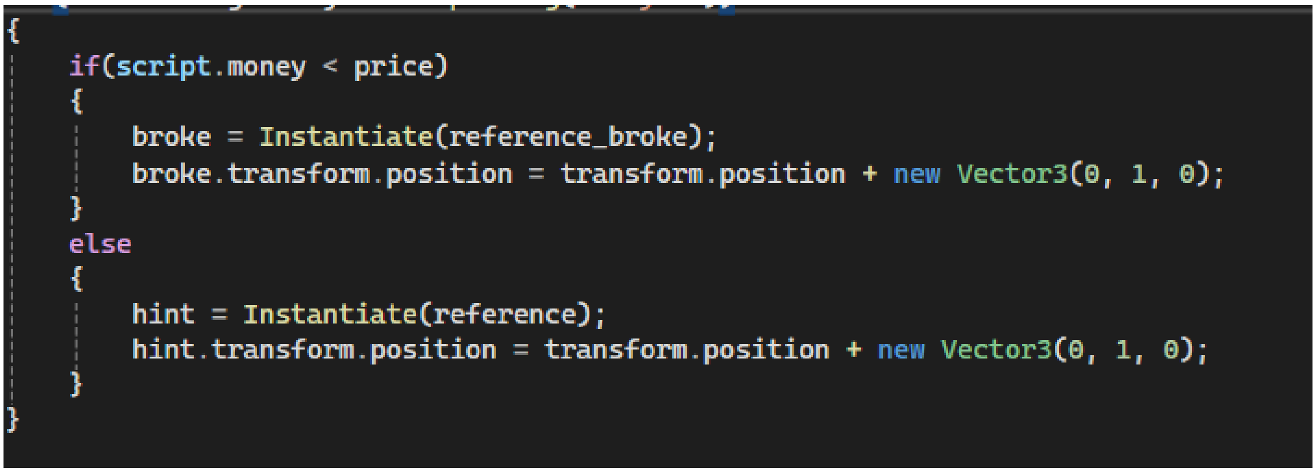
```
item = Instantiate(itemRef[index]);
Item script = item.GetComponent<Item>();
script.price = prices[i];
item.transform.position = location.position +
```

Inside the merchant code I needed a way to use the array of prices I would assign the objects. I needed to get a reference to the script and put it in that way



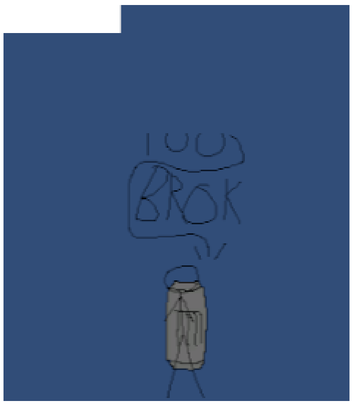
```
if (Input.GetKeyDown(KeyCode.E) && script.money > price)
{
    script.money -= price;
    script.addToInventory(name);
    Destroy(gameObject);
}
```

Additionally I added the check to see if the player has enough money to get the item. If they do, it takes the money away from the player, it adds it to the inventory, then destroys the object



```
{
    if(script.money < price)
    {
        broke = Instantiate(reference_broke);
        broke.transform.position = transform.position + new Vector3(0, 1, 0);
    }
    else
    {
        hint = Instantiate(reference);
        hint.transform.position = transform.position + new Vector3(0, 1, 0);
    }
}
```

In this part I implement a way to show the player whether or not they are able to buy an item. If they do not have enough money, then it will display this:



The sprite itself is cut off but it conveys the idea

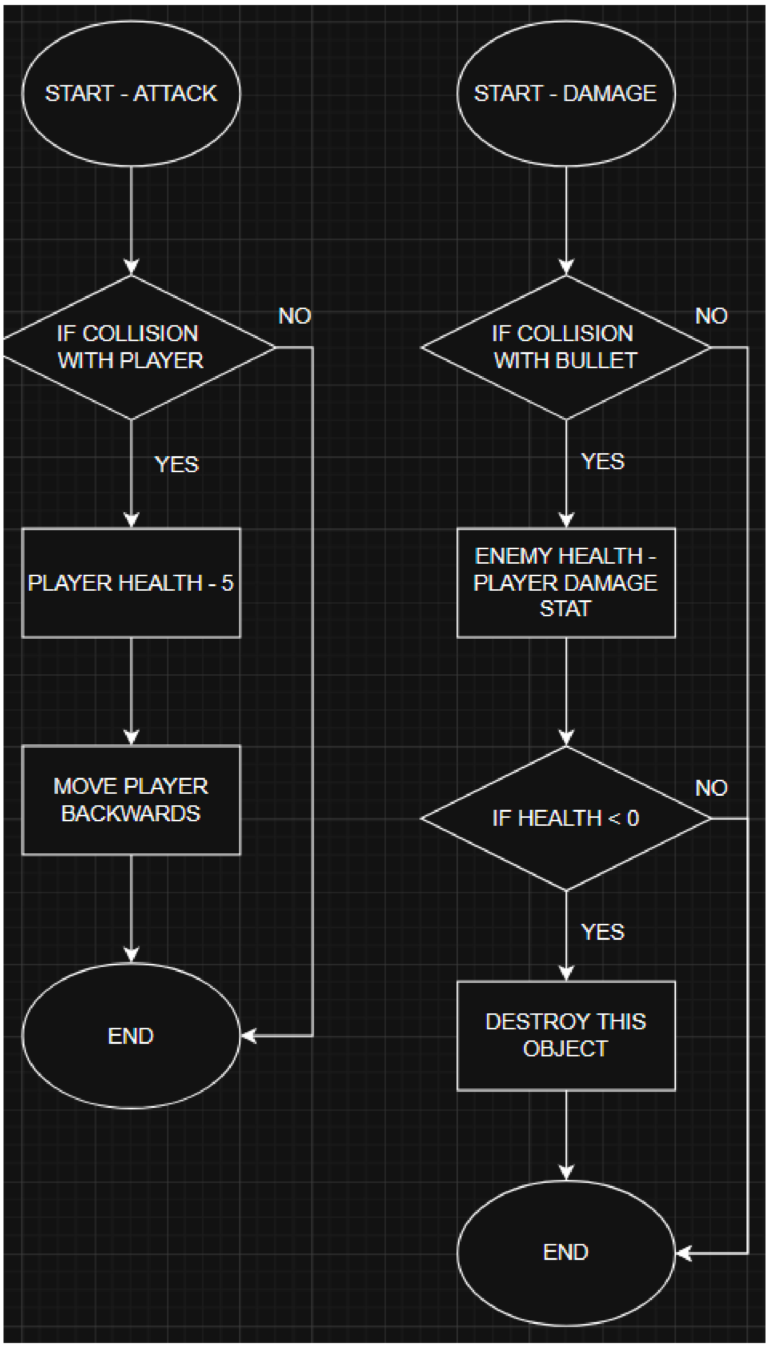
Test No.	Test	Expected outcome	Success?	Outcome
1	Adding random items to shop – to have the shop generate 3 different items to sell	It will have 3 items on display	YES	It generates 3 random items
2	Generate random prices – it will assign a random price to each item	The items and weapons will have different prices	YES	It successfully generates random prices for the items

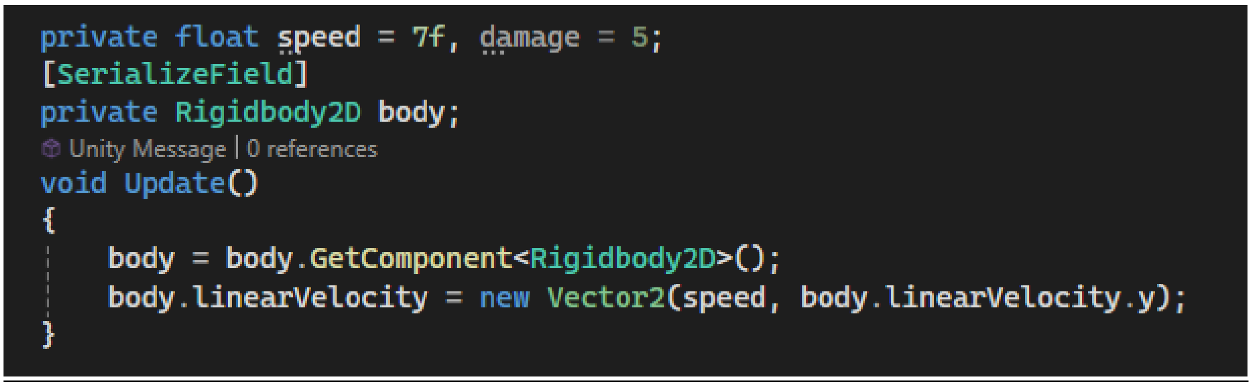
3	Buying the item – to check on input if the player has enough money to buy the item	It will remove the money from the player's inventory	YES	It successfully removes the money from the player
4	Picking up an item – to check on collision and input if the player picks up an item	The player picks up the weapon and adds it to the inventory on input	YES	It successfully removes the item from the game and adds the effect into the player
5	Giving the item to the player – it will remove the item from the shop and add it to the player inventory	It will remove the item from the shop and add it to the player inventory	YES	It adds the effect of the item onto the player

Part two complete

Part 3 – enemies

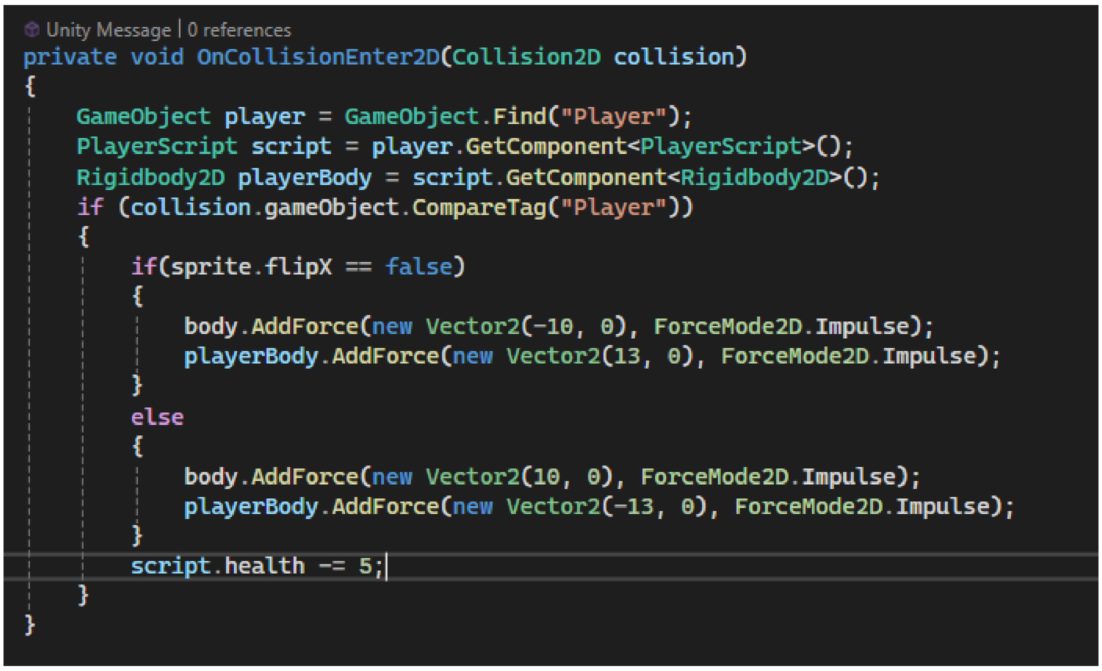
Enemy
-health: float -damage: float -speed: float
-Attack: void -TakeDamage: void



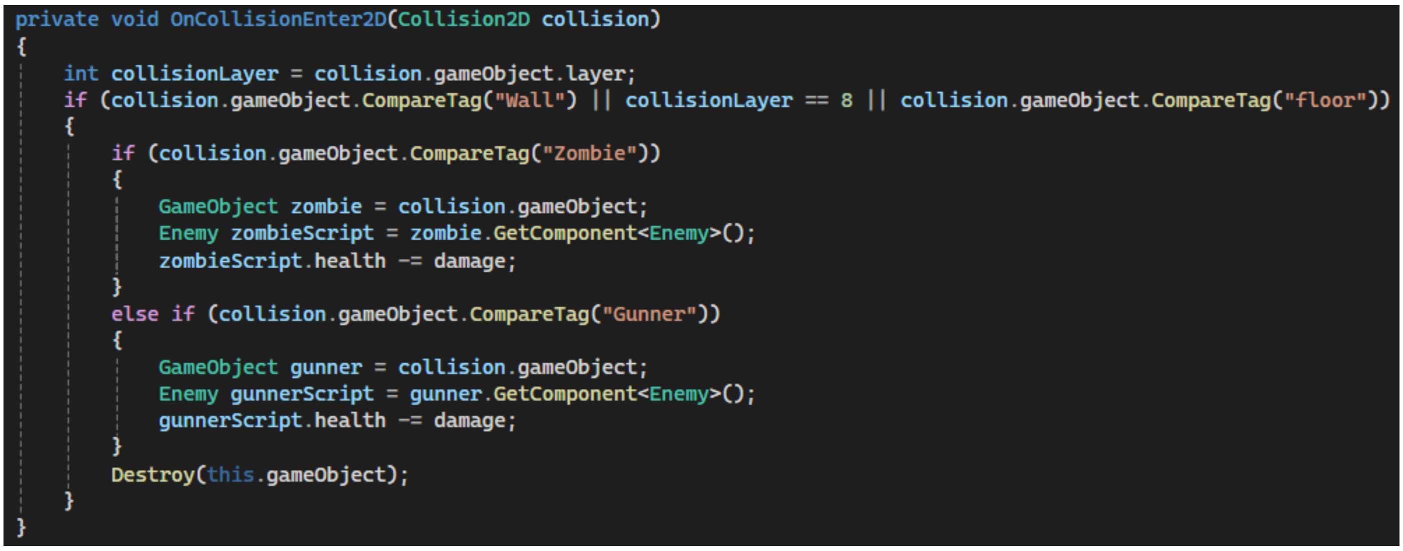


```
private float speed = 7f, damage = 5;
[SerializeField]
private Rigidbody2D body;
Unity Message | 0 references
void Update()
{
    body = body.GetComponent<Rigidbody2D>();
    body.linearVelocity = new Vector2(speed, body.linearVelocity.y);
}
```

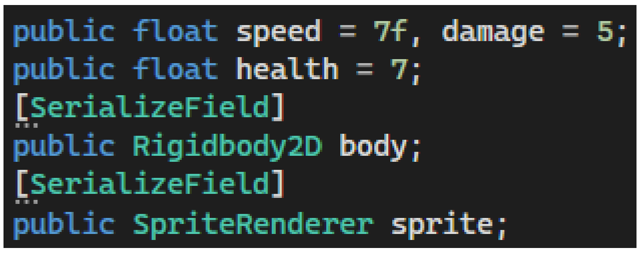
For the beginning of the enemy, I added the movement for them. It will detect the rigidbody2d on the enemy and add horizontal force


```
Unity Message | 0 references
private void OnCollisionEnter2D(Collision2D collision)
{
    GameObject player = GameObject.Find("Player");
    PlayerScript script = player.GetComponent<PlayerScript>();
    Rigidbody2D playerBody = script.GetComponent<Rigidbody2D>();
    if (collision.gameObject.CompareTag("Player"))
    {
        if(sprite.flipX == false)
        {
            body.AddForce(new Vector2(-10, 0), ForceMode2D.Impulse);
            playerBody.AddForce(new Vector2(13, 0), ForceMode2D.Impulse);
        }
        else
        {
            body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
            playerBody.AddForce(new Vector2(-13, 0), ForceMode2D.Impulse);
        }
        script.health -= 5;
    }
}
```

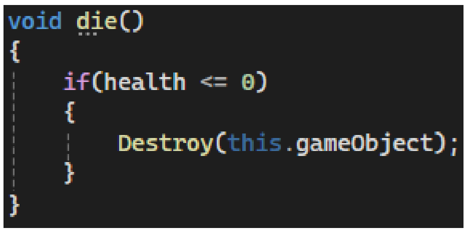
In this collision detection code, it first of all gets references to the player, the rigidbody2d on the player, and the script on the player. It needs the rigidbody2d reference to add knockback to the enemy and the player, by adding force to them according to the direction the enemy was facing. Additionally, it needs the script as it removes health from the player if they are hit


```
private void OnCollisionEnter2D(Collision2D collision)
{
    int collisionLayer = collision.gameObject.layer;
    if (collision.gameObject.CompareTag("Wall") || collisionLayer == 8 || collision.gameObject.CompareTag("floor"))
    {
        if (collision.gameObject.CompareTag("Zombie"))
        {
            GameObject zombie = collision.gameObject;
            Enemy zombieScript = zombie.GetComponent<Enemy>();
            zombieScript.health -= damage;
        }
        else if (collision.gameObject.CompareTag("Gunner"))
        {
            GameObject gunner = collision.gameObject;
            Enemy gunnerScript = gunner.GetComponent<Enemy>();
            gunnerScript.health -= damage;
        }
        Destroy(this.gameObject);
    }
}
```

In the bullet code, I added a collision detection for the bullet to tell what it is colliding with. If it detects collision with a enemy, it will get a reference to that enemy and remove health from them. This allows the bullet to damage the enemies while also removing itself on collision

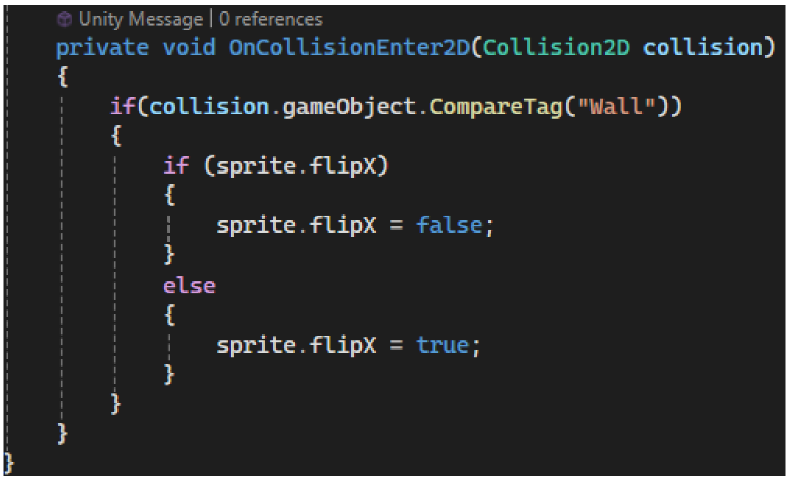

```
public float speed = 7f, damage = 5;
public float health = 7;
[SerializeField]
public Rigidbody2D body;
[SerializeField]
public SpriteRenderer sprite;
```

In the enemy code, I decided to create an enemy class and extend that to the other two enemy scripts. I decided to add speed, health and damage to them, as well as references to the body and sprites. This is because I need to add force to the enemy to move them, as well as the spriterenderer to check which way they are facing



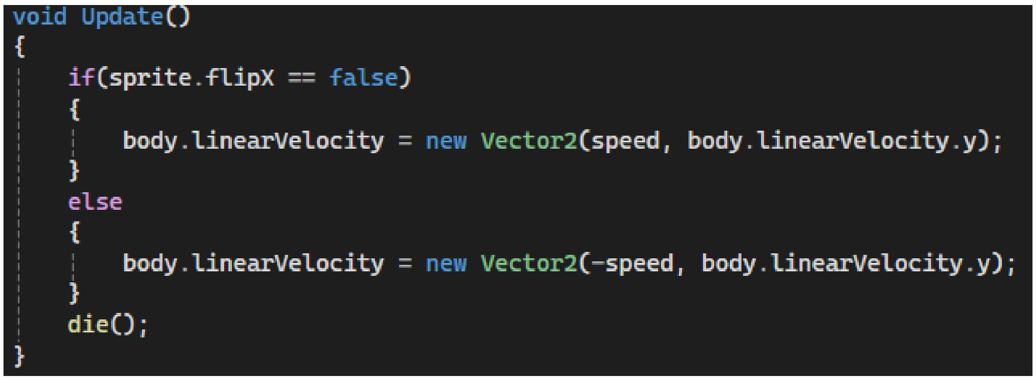
```
void die()
{
    if(health <= 0)
    {
        Destroy(this.gameObject);
    }
}
```

Death will trigger on reaching 0 or less health, removing the enemy



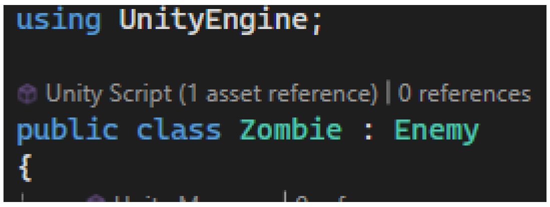
```
    Unity Message | 0 references
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.CompareTag("Wall"))
        {
            if (sprite.flipX)
            {
                sprite.flipX = false;
            }
            else
            {
                sprite.flipX = true;
            }
        }
    }
}
```

This will detect whether or not the enemy if facing forwards or backwards, and it will flip them if they collide with a wall. This is important for the movement part of the code



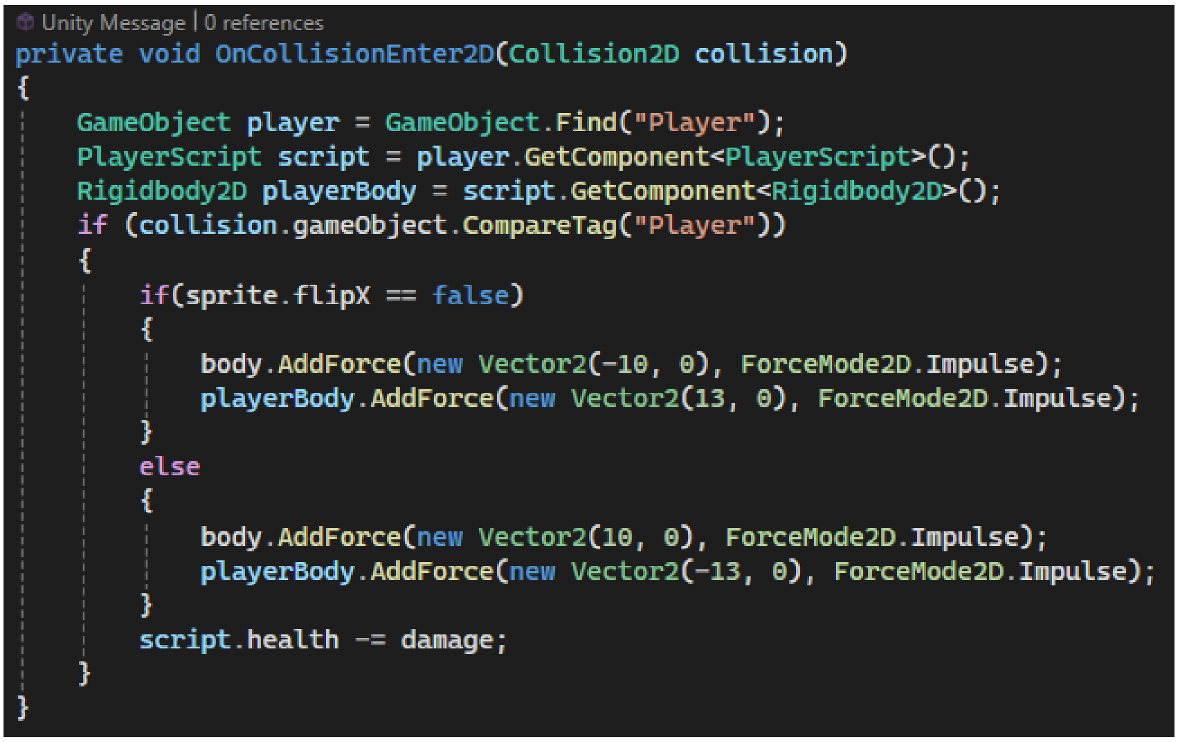
```
void Update()
{
    if(sprite.flipX == false)
    {
        body.linearVelocity = new Vector2(speed, body.linearVelocity.y);
    }
    else
    {
        body.linearVelocity = new Vector2(-speed, body.linearVelocity.y);
    }
    die();
}
```

Within update is the movement code, and it will reverse the speed on the enemy if they are facing forwards or backwards. This also links with the collider code to reverse the sprite on wall collision



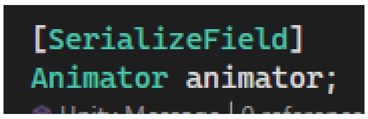
```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Zombie : Enemy
{
```

In zombie, it inherits from the enemy script, giving it access to all prior variables and subroutines



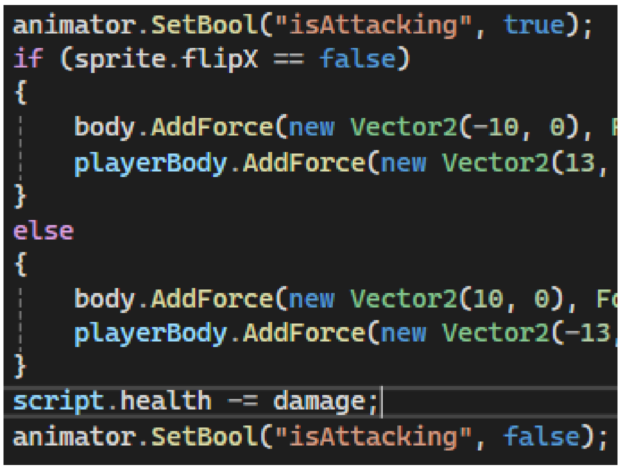
```
Unity Message | 0 references
private void OnCollisionEnter2D(Collision2D collision)
{
    GameObject player = GameObject.Find("Player");
    PlayerScript script = player.GetComponent<PlayerScript>();
    Rigidbody2D playerBody = script.GetComponent<Rigidbody2D>();
    if (collision.gameObject.CompareTag("Player"))
    {
        if(sprite.flipX == false)
        {
            body.AddForce(new Vector2(-10, 0), ForceMode2D.Impulse);
            playerBody.AddForce(new Vector2(13, 0), ForceMode2D.Impulse);
        }
        else
        {
            body.AddForce(new Vector2(10, 0), ForceMode2D.Impulse);
            playerBody.AddForce(new Vector2(-13, 0), ForceMode2D.Impulse);
        }
        script.health -= damage;
    }
}
```

For the zombie code, the whole of it is one collision detection part. It gets references to player, the player script, and the body of the player. This is because later in the code it knocks back the player. It also removes health from the player



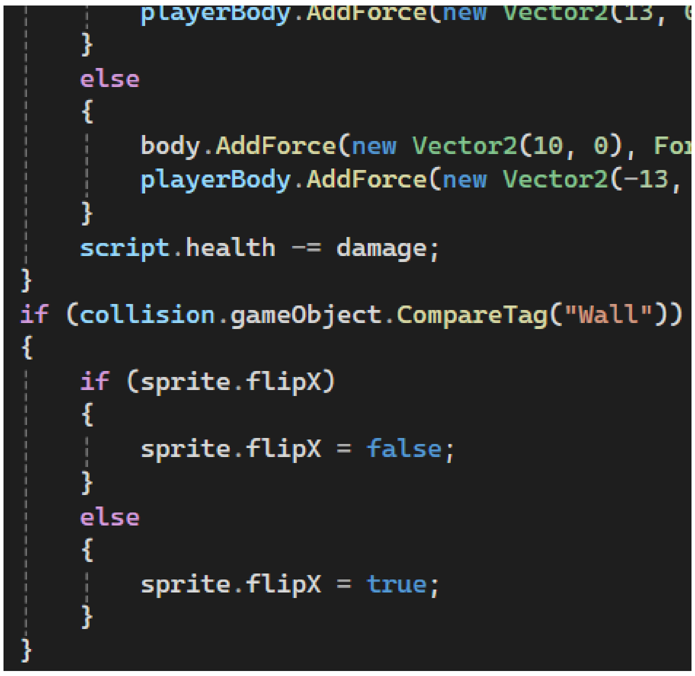
```
[SerializeField]
Animator animator;
```

In addition to the zombie code, I added an animator for the zombie to play when it attacks the player

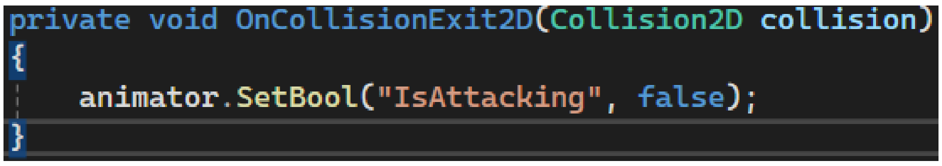


```
animator.SetBool("isAttacking", true);
if (sprite.flipX == false)
{
    body.AddForce(new Vector2(-10, 0),
    playerBody.AddForce(new Vector2(13,
}
else
{
    body.AddForce(new Vector2(10, 0), F
    playerBody.AddForce(new Vector2(-13
}
script.health -= damage;
animator.SetBool("isAttacking", false);
```

In this, I change the bool isAttacking to true, to play the animation to true, then to false. This is to play the attack animation

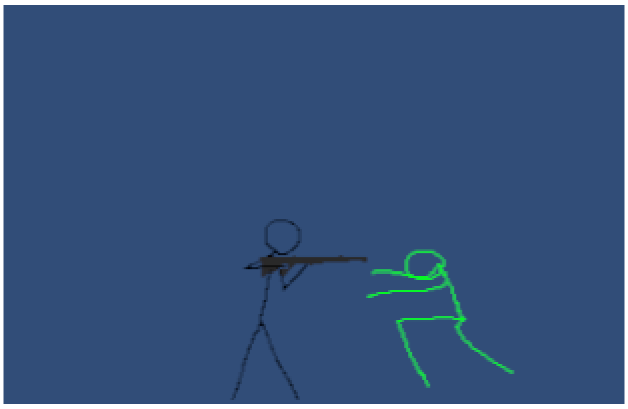


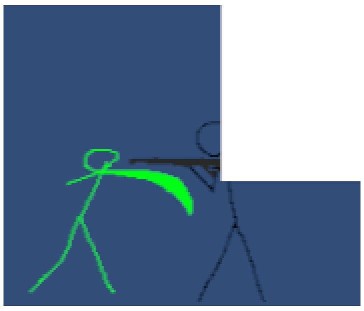
```
        playerBody.AddForce(new Vector2(13, 
    }
    else
    {
        body.AddForce(new Vector2(10, 0), For
        playerBody.AddForce(new Vector2(-13,
    }
    script.health -= damage;
}
if (collision.gameObject.CompareTag("Wall"))
{
    if (sprite.flipX)
    {
        sprite.flipX = false;
    }
    else
    {
        sprite.flipX = true;
    }
}
```

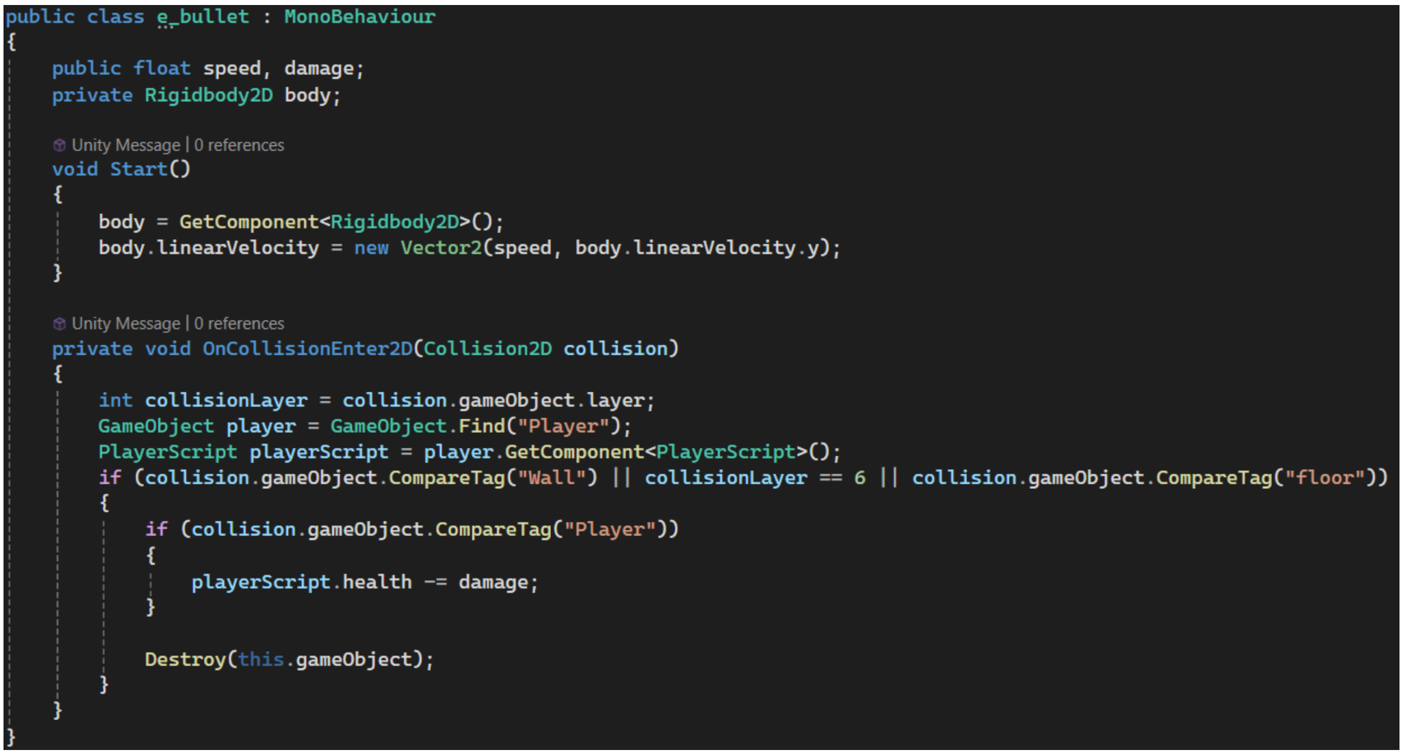


```
private void OnCollisionExit2D(Collision2D collision)
{
    animator.SetBool("IsAttacking", false);
}
```

I had to graft the sprite flipping into the oncolliisonenter2d in the zombie code as it cannot be called without it overwriting the initial one. It can now flip the sprite successfully as well as play the attacking animation. I also had to add an exit to change the sprite from flipping as update would constantly turn it off each frame:

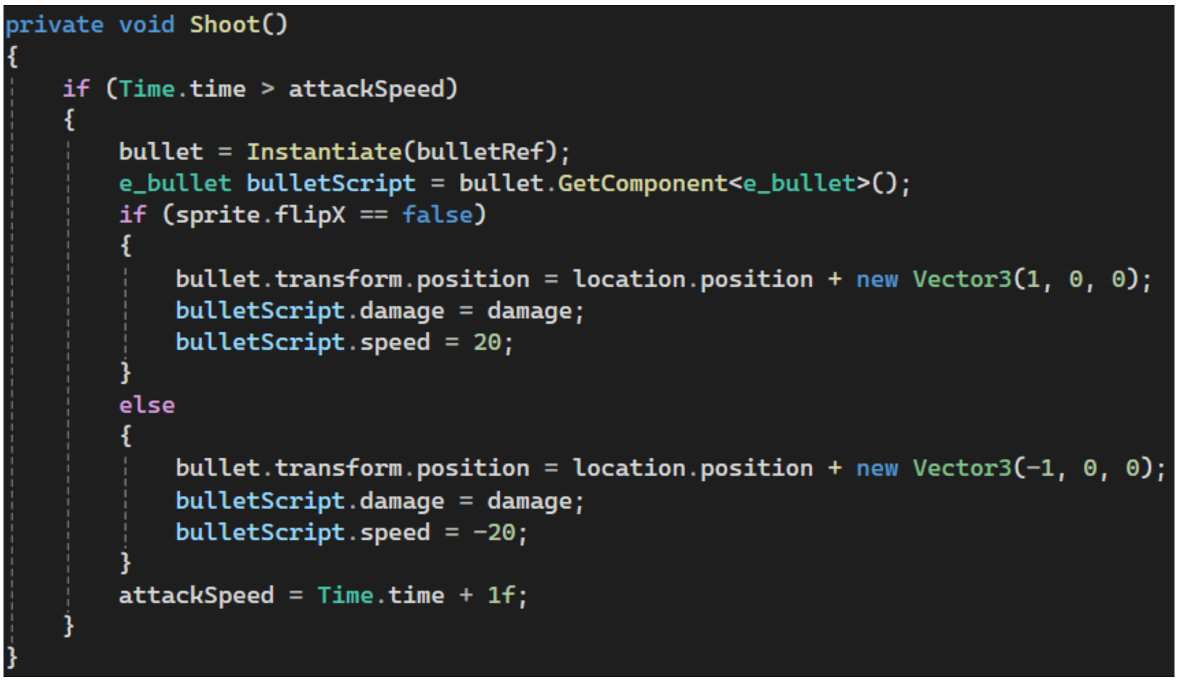






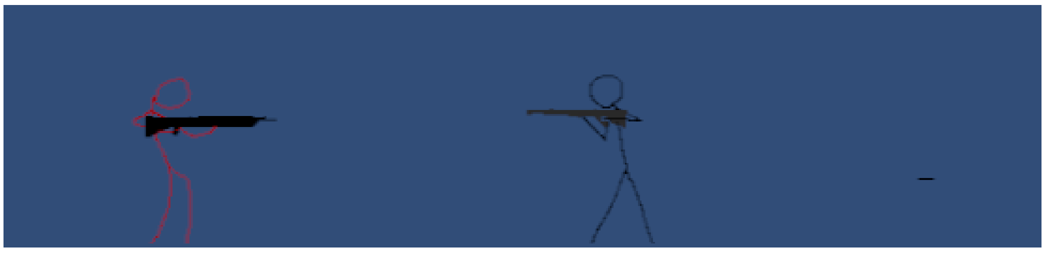
```
public class e_bullet : MonoBehaviour
{
    public float speed, damage;
    private Rigidbody2D body;

    Unity Message | 0 references
    void Start()
    {
        body = GetComponent<Rigidbody2D>();
        body.linearVelocity = new Vector2(speed, body.linearVelocity.y);
    }

    Unity Message | 0 references
    private void OnCollisionEnter2D(Collision2D collision)
    {
        int collisionLayer = collision.gameObject.layer;
        GameObject player = GameObject.Find("Player");
        PlayerScript playerScript = player.GetComponent<PlayerScript>();
        if (collision.gameObject.CompareTag("Wall") || collisionLayer == 6 || collision.gameObject.CompareTag("floor"))
        {
            if (collision.gameObject.CompareTag("Player"))
            {
                playerScript.health -= damage;
            }

            Destroy(this.gameObject);
        }
    }
}
```

The code for the enemy bullet is largely the same as the regular bullet. The only change is that it gets a reference to the player instead of the enemies and it removes player health on collision



```
private void Shoot()
{
    if (Time.time > attackSpeed)
    {
        bullet = Instantiate(bulletRef);
        e_bullet bulletScript = bullet.GetComponent<e_bullet>();
        if (sprite.flipX == false)
        {
            bullet.transform.position = location.position + new Vector3(1, 0, 0);
            bulletScript.damage = damage;
            bulletScript.speed = 20;
        }
        else
        {
            bullet.transform.position = location.position + new Vector3(-1, 0, 0);
            bulletScript.damage = damage;
            bulletScript.speed = -20;
        }
        attackSpeed = Time.time + 1f;
    }
}
```

Shoot is also similar to the player’s code, as it works mostly the same. It instead gets a reference to the e_bullet, or the enemy bullet.



It is firing the bullets, meaning it works

Test No.	Test	Expected outcome	Success?	Outcome
1	Movement – allows the enemy to automatically move	It will detect the player and move towards them	YES	The enemies move towards a wall until collision, then they flip
2	Attack – it will try to attack towards the player	It will attempt to attack the player	YES	They do try to attack the player when they are colliding
3	Damage – it will deal damage to the player on contact.	It will remove the health from the player and stun the player	YES	They do deal damage to the player
3.1 – player class	Death – on taking cumulative damage which reduces the player health to zero or less, it will play an animation and die	The player will die	YES	The player can die when hit enough
4	Taking damage – the player can damage the enemy by shooting it with their weapons	It will remove health from the enemy on hit	YES	The player can damage and kill the enemies
5	Death – on HP being reduced to 0 or less, it will remove itself	It will die	YES	The enemies can die

<u>Part 3 complete</u>

References:

Learn Unity - Beginner's Game Development Tutorial

C# Exceptions (Try..Catch)

destroying assets is not permitted to avoid data loss

How do i call a function in another gameobject script