

ANALYSIS

Project title: TBD

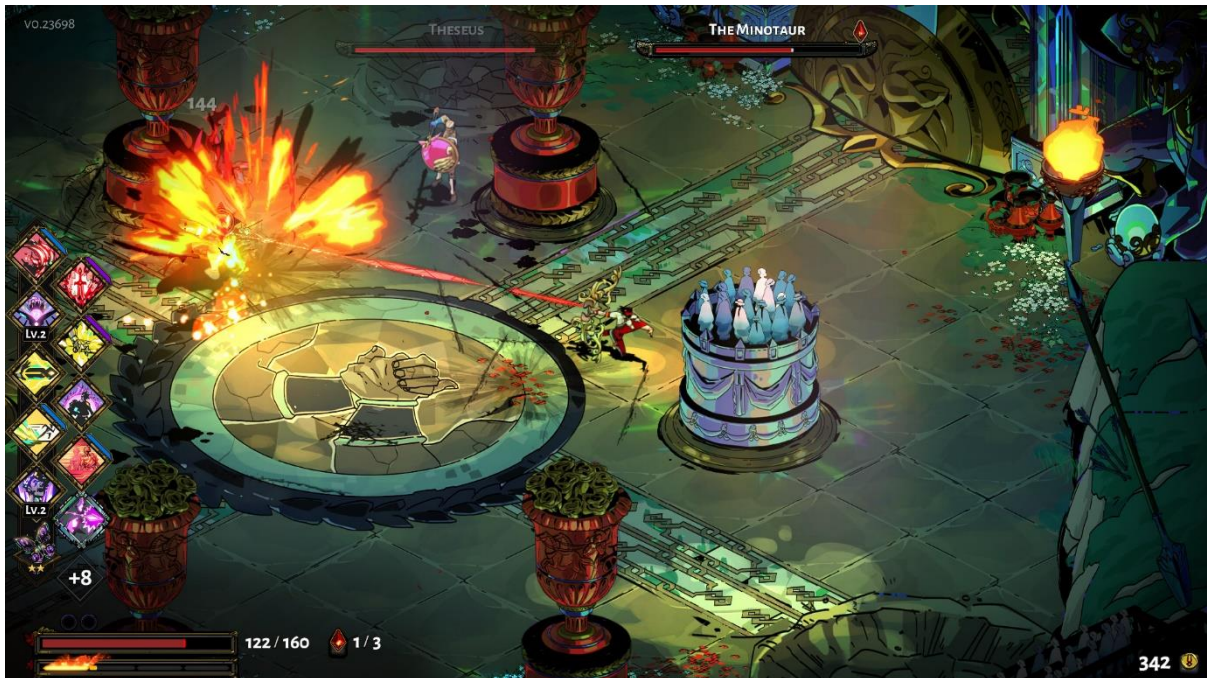
My project will be a 2D side on rougelite in which you battle through procedurally generated rooms trying to reach a final boss. What will set it apart from the other games of this genre will be how you acquire your permeant upgrades as they will be obtained by raising your affinity with the various NPCs that will be littered throughout the game. The game will include a multitude of different bosses of increasing difficulty as you progress through the world. The game would also include a separate area accessed outside of the combat section where you can talk to the NPCs and build up your relationships as well as your build. I believe that this is a promising idea as it would allow for an easier base game that would allow for building harder mechanics on top of the base. To improve the game more characters could be added, or each character could be given longer stories.

There will also be enemies in the rooms that will increase in difficulty the further into the world you get. Which the player will have to defeat before they can progress to the next room.

The game will be developed in unity and developed for PC and so I will be using keyboard input

Inspiration

1. Hades



(image from hades steam page)



(Hades guide: The Trial of the Gods)



(HADES - BOND FORGED)

What I want to use	What I will not be carrying forward
I like the level select of Hades as it allows for customisability in your run and allows you to choose how the run turns out	I don't want to use the top-down isometric view of the game as it would be harder for me to reproduce and more time consuming
I like the intractability of most of the non-enemy NPCs as it helps build the world around you and makes you feel more included in the story	I also don't like the combat system of the game as for all non-boss enemy's the strategy is to hit them repeatedly as they get stun locked and cannot react leaving little room for improvement in the repeated gameplay
I will be copying over the keepsake system as it is the perfect representation of what I would like the upgrade system of my game to be like	I disliked the fact that only one item is used to build an increase your affinity with each NPC as it makes the relationships feel repetitive

2. Dead cells



('Dead Cells' is a fun way to spend some of your midsummer leisure time - The Washington Post)

What I want to use	What I will not be carrying forward
One of the things I will be using from this game is their 2D side on perspective as it allows me to work more comfortably with the game engine	One of the things I won't be using from dead cells is the map layout as it leaves the whole map open and revisit able at any time and would require the map to not loop back on itself
Another thing I liked about dead cells is the ability to upgrade your weapons during the run or even swap them out for different ones all together	Another thing I won't be bringing over is map unlocks that you get between runs as I want all the map to be accessible from your first run without not realising you aren't able to access a place.
I also liked the fluid movement of dead cells and the way you can smoothly rush between enemy encounters feels exceptionally good and I want to include something like it.	I also won't be using the secondary ability's as it complicates the combat and control scheme as well as not being the friendliest for new players and less experienced players

3. neon abyss



(neon abyss on steam)

What I want to use	What I will not be carrying forward
I like the way the rooms are linked together and the way you flesh out the map as you progress	I don't like the use of ranged weapons as it makes the game feel less smooth in its combat

I like the boss fights that are at the end of each level as they are challenging and exiting	I will not be using the art design as it isn't the kind of atmosphere I want in my game.
I like the variety of rooms that you can encounter as it adds a lot of uniqueness to each run.	I will not be taking inspiration from the multitude of locked items that aren't accessible without keys or bombs as they are annoying

STAKEHOLDERS

My stakeholders are gamers aged between 16 and 25 of all genders. Specifically aimed at gamers who have played games of the same genre and enjoyed them as well as those who enjoy the story of a game over just focusing on the combat. However, the game isn't limited to these players and could be used by older users as well. This would be well suited to this game as it allows players to relax after a day of studying or working by destressing in a relaxing story-based game or for more experienced players to let out some of their stress on beating up enemies. I will be involving my stakeholders in both my initial design plans by asking them questions on what they would like to see in the game and after development during testing to see what they would like to see improved and which bits of the game they enjoy.

Requirements


Feature	Sub-feature	explanation	justification	importance
levels	At least 2 unique levels each with unique enemies	Level designs that are separate to each other	To reduce repetitiveness and increase playtime multiple levels would be exiting for the players	necessary
	Doors in each room	Doors that lead to the next room	To stop players from blitzing through a level and then being underpowered you won't be able to progress until you've completed your current room	necessary

	Different rewards for each room	Depending on what you want you can go there first	Allows the user to have some choice in what they build and can open ideas for more intricate rooms and rewards and makes each run different to increase replay ability	necessary
Main menu	Saving/loading saves	Allows player to keep their save between sessions. Multiple slots	Increases player retention and allows for less repetitive gameplay. Also helps with testing as you can edit your save to test out certain parts.	necessary
	Settings button	To allow changing to the settings	Allows me to add accessibility options if I get the opportunity as well as changing the volume of the game	desired
	Quit game button	Closes the program	Allows the player to close the game safely without breaking anything unlike alt+f4	necessary
Tutorial	NPC interaction	Teaches you the basics of talking to the NPCs	To ease the player into the game smoothly to increase enjoyment and understanding of the mechanics	desired
	Enemy fighting	Teaches you the basics of combat		
	Movement	Teaches you the movement controls		

weapons	At least 4 unique weapons	Weapons with different move sets	This increases replay ability and allows for different builds	necessary
	Weapons selection at start of run	Choose one weapon from the 4 to use for that run	Let's you decide how you want to play without having to rely on luck to give you the weapons you want. My stakeholders prefer this over the other option as well	necessary
	normal and special attacks	The ability to use either a normal or a special attack	Allows for more dynamic combat and an enhanced combat system which was a big point in the feedback that I got from my stakeholders	desired

7. would you prefer

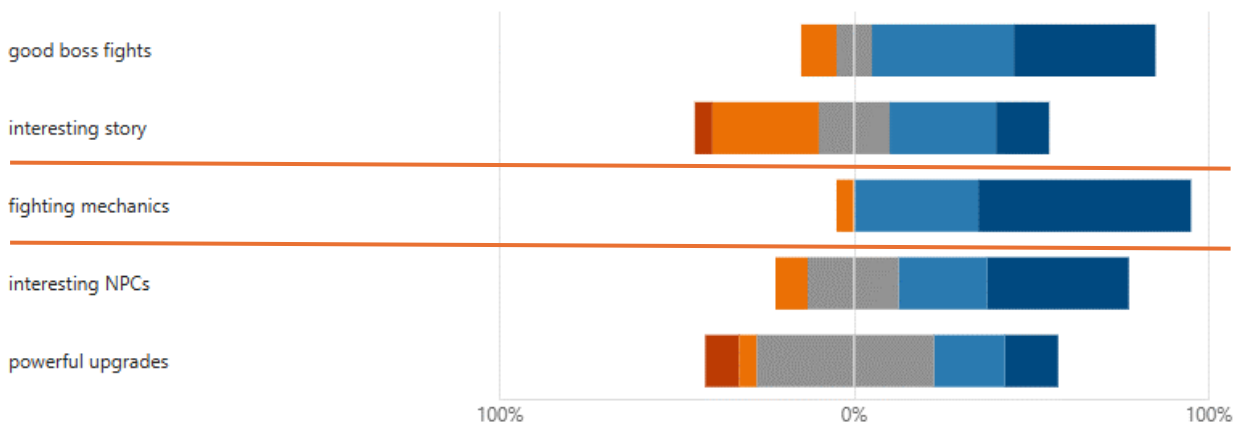
[More Details](#)

 Insights

- find weapons as you go 6
- one chosen weapon per run 14



■ not at all ■ a bit ■ kind of ■ pretty ■ very

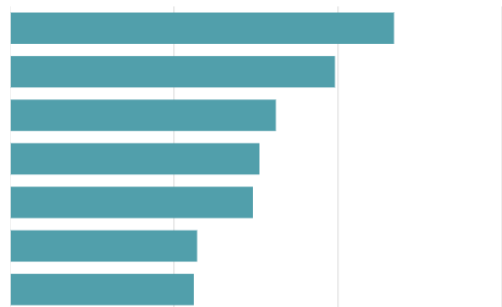


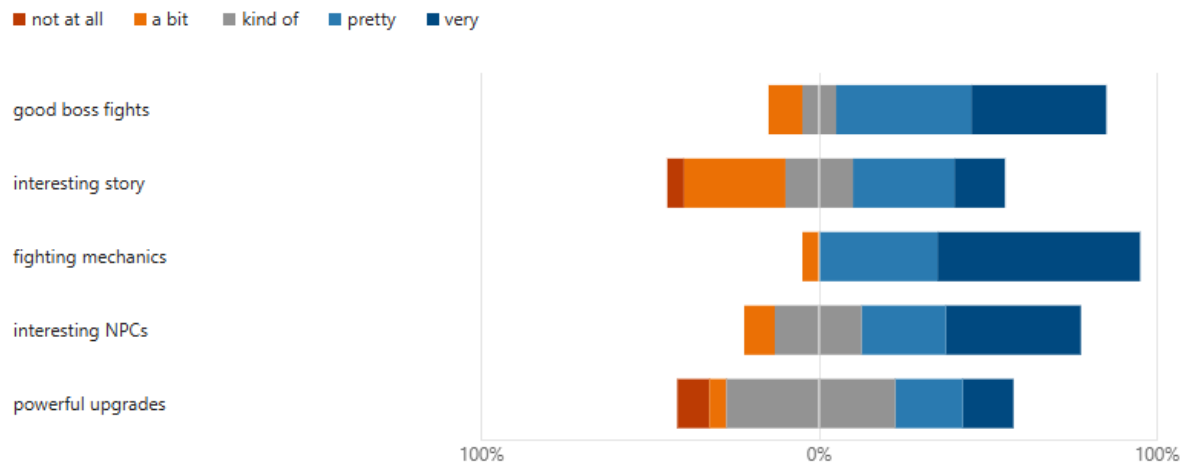
NPCs	NPC affinity levels	Shows how much an NPC likes you.	This will be used to allow for permanent upgrades between runs. It was also well received in my user feedback.	necessary
	At least 5 different NPCs	Each with their own name and personality	This gives the game more of a choice element and allows for a more unique experience and increased replay ability.	necessary
	NPC rewards	Upgrades for your character based on affinity levels	This incentivises you to talk to the NPCs and get more invested in the story and for the people who are more invested in the combat aspects of the game gives them a chance to experience the other side	necessary
	Different questlines	Allows you to complete goals on runs to level up affinity	This would be needed to allow for personalised NPC interactions and would allow you to get more invested in the characters and the story	desired

5. which of these mechanics are most important to you

[More Details](#)

- 1 fluid movement
- 2 challenging boss fights
- 3 NPC affinity
- 4 unique weapons
- 5 interesting story
- 6 art style
- 7 permanent upgrades



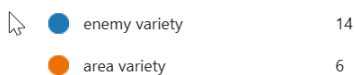


Enemies	Attacks	Would be used to damage the player	Allows for a sense of danger and accomplishment for clearing the rooms	necessary
	Take damage	So that they can be killed	Allows satisfying feedback for the user's actions	necessary
	Have loot	Drop items or currency used for upgrades	Allows for more variance between the enemies and give better rewards for more challenging foes	desired
	Have at least 4 unique types	Each with different sprites and attacks	Players wanted this over level variety and so I will focus more on multiple enemies over levels	necessary

11. which is more important to you

[More Details](#)

[Insights](#)



5. which of these mechanics are most important to you

[More Details](#)



12. do you prefer:

[More Details](#)

[Insights](#)

- all movement options from start... 7
- unlock movement options throu... 13



movement	Make movement fluid	Allow smooth transitions between movement options	Gives the game a more polished feeling and was top of the list of things my target audience wanted	necessary
	Have unlockable upgrades	Ability to acquire new movement options like a double jump or a dash	Allows for a sense of achievement for exploring the game and adds incentive for the players to keep playing. Was preferred over having all of them at the beginning by my stakeholders	necessary
	dodge	Used to evade enemy attacks	Gives more variety to the combat and makes it more engaging as well as adding more depth to boss fights and enemies moves	necessary

Boss fights	Shows up at and of level	Final room at end of level	Adds an extra layer of challenge to the run and gives a substantial ultimate point to the run	necessary
	Has unique move set	At least 4 different attacks	Allows the fights to be entertaining without boring the player and being too hard to remember	necessary
	challenging	Make sure the boss isn't too easy	Gives a larger sense of accomplishment and was highly requested by my stakeholders	desired
	arena	Give the boss their own separate room	Adds more immersion to the fight and adds to r The user experience	desired
story	Cohesive storyline	Story points that flow into each other	Gives the player a reason to keep playing and had good responses on the stakeholder survey	desired
	Different game endings	Different outcomes based on the players actions	Adds to the replay ability and rewards players for doing their own thing and exploring	Optional

5. which of these mechanics are most important to you

[More Details](#)



Limitations

limitation	explanation
Level designs	As my stakeholders were more interested in the enemy variety, I won't put too much work into creating lots of unique level designs only one or two as I'm focusing my time more on the enemies
Difficulty settings	Due to my stakeholders saying they want the game to be challenging I will not focus lots of time on making the bosses easier as they don't want that and I would have to make substantial changes to the enemies
Player customisation	As my stakeholders on average didn't really care about the player customisation and it would be extremely hard and time consuming to implement as I'm not a fast artist
multiplayer	As few of my genre of games include multiplayer or online features and these features would require a totally different type of game style, I have chosen not to implement them
Custom sprites and animations	As looks are not assessed and I'm not a particularly good or fast artist I will rely on free assets and won't add many personalised ones unless necessary or all my requirements have been met.

10. how important is character customisation to your game experience

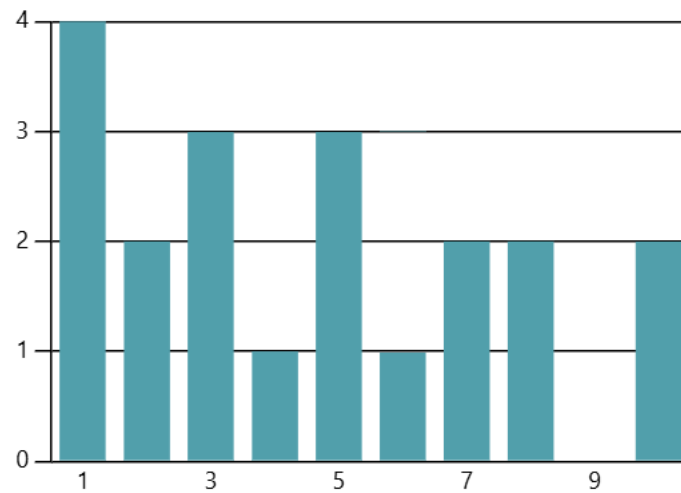
[More Details](#)

[Insights](#)



4.6

Average Rating



My Success Criteria

criteria	explanation	justification
At least 2 unique levels each with unique enemies	Level designs that are separate to each other	To dissuade repetitiveness and increase playtime multiple levels would be exiting for the players
At least 4 unique weapons	Weapons with different move sets	This increases replay ability and allows for different builds
Enemies that you can fight for rewards and to progress	Hostile NPCs that will engage in combat with the player to give them a challenge	This will provide a challenge for the player allowing them to enjoy the game and provide a learning curve to the game to increase playtime
NPCs that you can talk to outside of combat	At least 4 different friendly NPCs that the player will help and be helped by over the course of the game	This is the main selling point of the game and so the game must have at least this much

An affinity system with the NPCs to unlock upgrades	A way to unlock new permanent upgrades by growing closer with the NPCs	This is also the main point of the game that sets it apart from other games of the same genre and so must be included. It also allows for the player to get immersed in something outside of combat
Have at least 4 unique enemies	Each with different sprites and attacks	Players wanted this over level variety and so I will focus more on multiple enemies over levels
One final boss	I big enemy with a custom move set and a challenging fight at the end of the level	This will provide a goal for the player and acts as a final point of the game showing that you have made it to the end of the game.
A playable character	A way for the user to interact with the game world	Otherwise, you couldn't play the game
Fluid movement	A system of movement that feels fluid and nice	Gives the game a more polished feeling and was top of the list of things my target audience wanted

Hardware & Software

Hardware

A computer with:

x86 or x64 architecture

DX10, DX11, DX12 or Vulkan capable GPUs

A keyboard and mouse and a screen

software

Windows 10 21H1 or higher

(unity, 2025)

Computational methods

This project is suitable for computation as it uses multiple aspects of computational thinking:

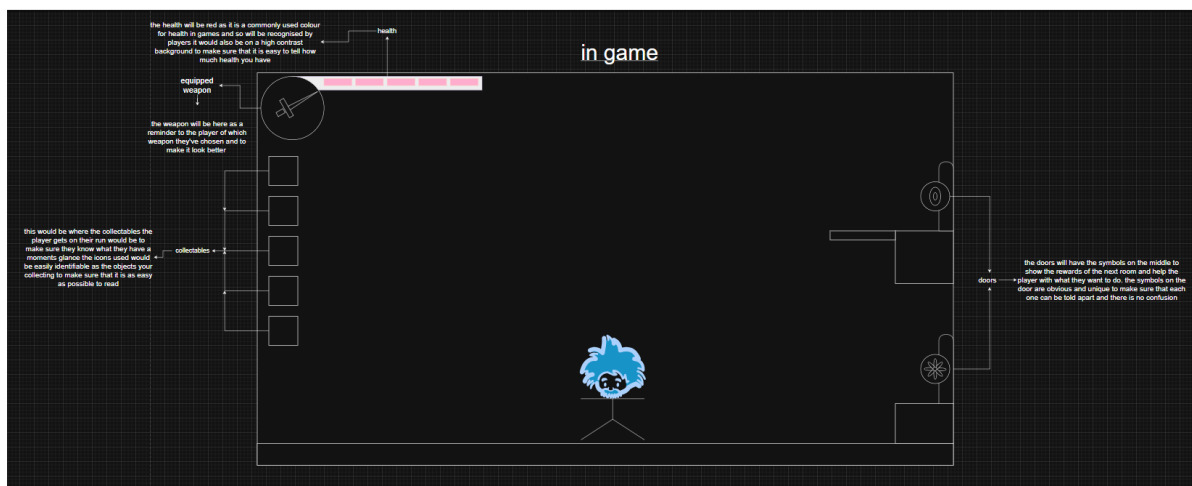
As there will be many things happening at once and so my game will be running them concurrently, this will be used to allow the player and the enemies to be moved at the

same time and allow them to interact with each other and the surrounding environment.

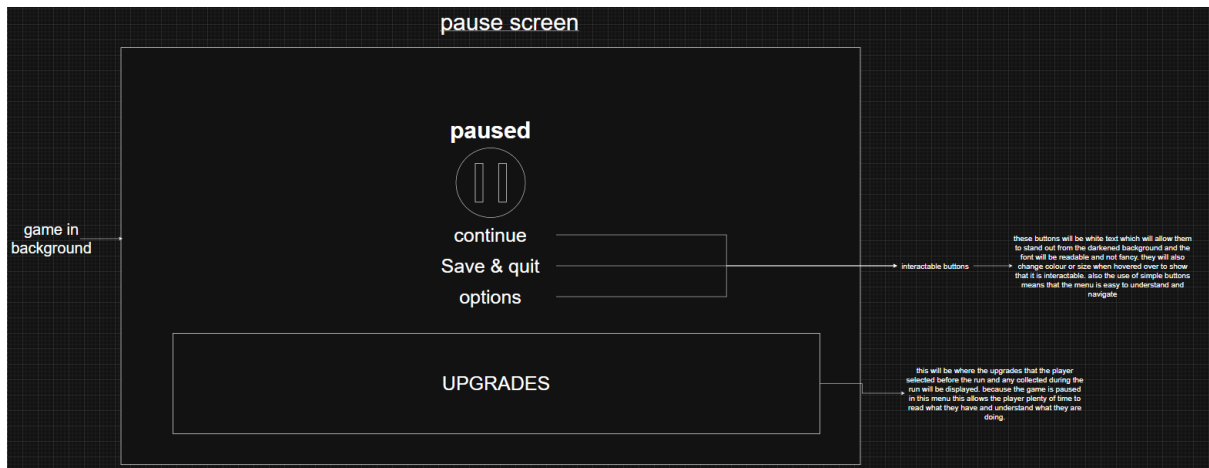
I will also be making use of abstraction to allow me to make sure that the game doesn't has a working version before I move onto adding features that are integral to the game running but are important to the experience. In addition, using abstraction will allow me to develop faster as I can ignore unnecessary parts of a feature and focus first on the main problem like making sure I have working enemies and attacks before I add the sprites for them. It will also be used to enhance the user experience as the user might not want to always see all parts of the GUI, and I could abstract away the parts that aren't being used in that situation.

I will be using decomposition to help me with my problems as it will allow me to break the large problem down into smaller pieces to allow me to focus on each one at a time. I can also pair this with abstraction to break down the most important parts of the program and focus on solving those first. I could use it to keep breaking up the smaller problems to allow myself to plan out the whole problem before starting work on it and having to backtrack.

DESIGN



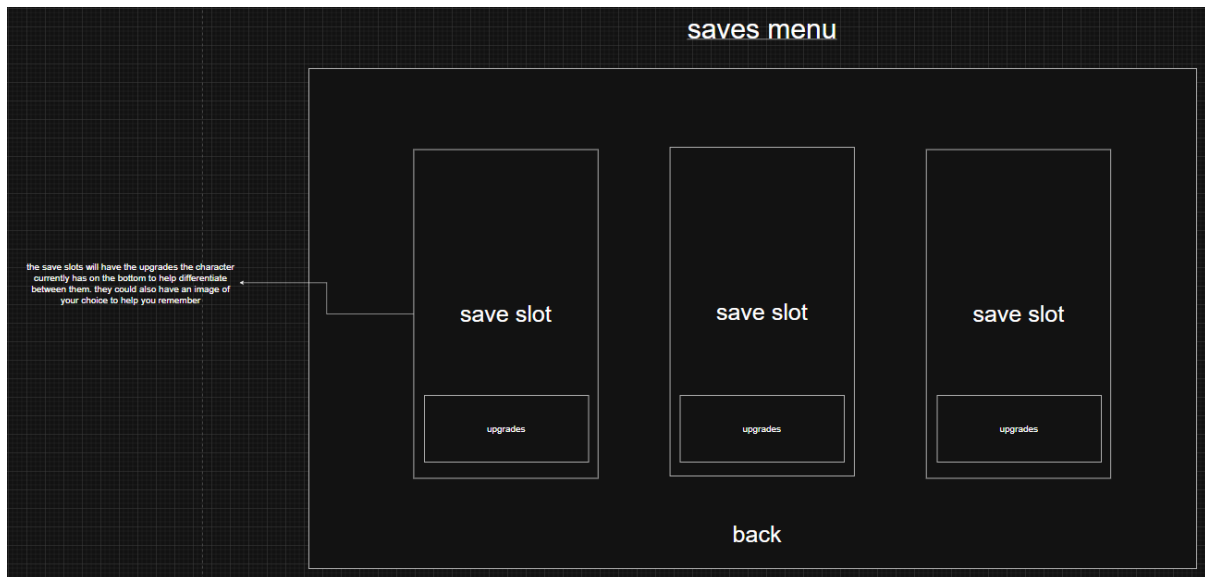
This is the screen for the main game. It has the players health and pickups on the left and is kept minimalistic to reduce distraction from the game with huge overbearing boxes and items. The circle in the top left shows the current weapon to help reduce confusion whilst not annoying the player



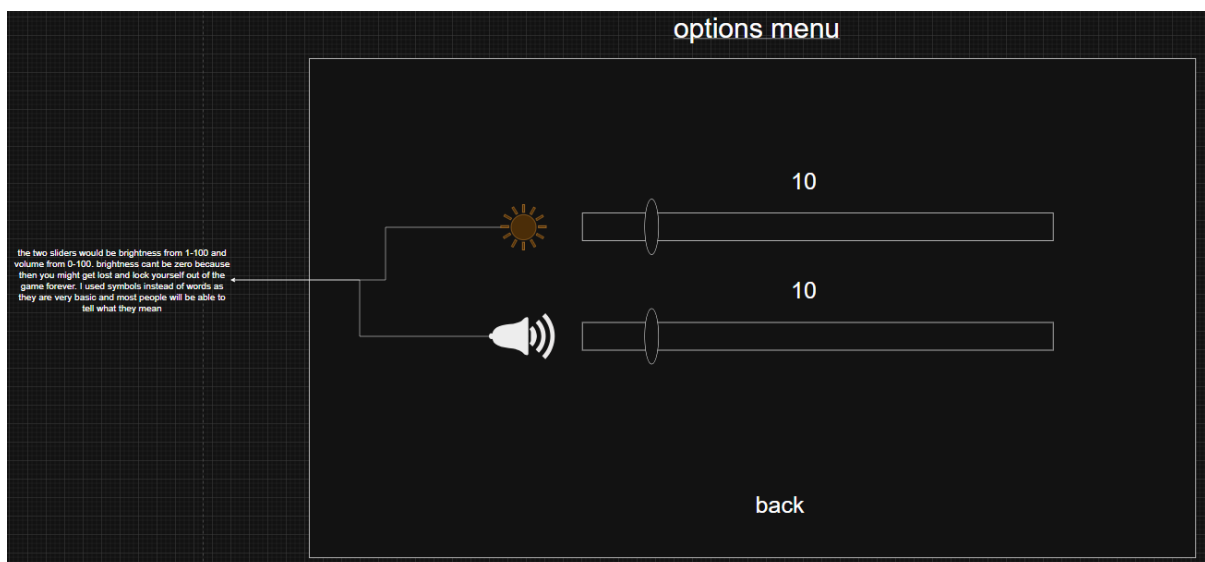
This is the pause screen that will popup when the player presses escape which will pause the game and blurs the game in the background to accentuate the buttons. The buttons will be white text to help them stand out and at the bottom are the upgrades the player has aquired and equipped. There here to keep them out of the way whilst playing but easily accessible if the player wants to check them in a run.



This is the first menu the player will see when they boot into the game and the screen they see when they come back to the game. As so i will make sure that it looks good. I will have an image in the background and good-looking buttons for the options on the left. As well as the title on the top.



The game will have 3 save slots which will display major achievements and progress by showing them via images on the bottom of the slot.



This will be where the settings are and as there aren't many it will only be one page with both on. Volume can go to zero to mute the game, but brightness can't as it would lock you out of the game forever.

Stages of development

Stage 1: player

In this stage I will create the player and the player's movement. As they are both needed to test each other out this and stage 1 may have to be developed simultaneously this also means that these two must be developed together. They are the first two because everything else in the game is built off one of these two concepts. These first two stages should only take 1-2 weeks

Test	Expected outcome	Actual outcome	pass/fail
Pressing the a key	Move the player left		
Pressing the d key	Move player right		
Pressing jump key	Player jumps		
Pressing dash key	Player dashes		
Player animations	Player animations play at the right time and in the right order		

Stage 2: combat world

This is where I will create the world the player will be fighting in including the multiple rooms and transitions

Test	Expected outcome	Actual outcome	pass/fail
Generating the world	World generates with linked rooms		
Moving between rooms	Stepping into the transition moves you to the room linked to that transition		
Different rewards for each room	Each door has a different reward to all the others in the same room		

Stage 3: menus

This is where I will make the non-gameplay menus where the player can adjust volume or save their game. They will be made next as this allows me to save my progress for testing and will save lots of time in the long run. This should be working within a week or two

Test	Expected outcome	Actual outcome	pass/fail
Pressing the escape key	The pause menu will be displayed		
Pressing the options button in the pause menu	Open the options menu		

Pressing the return button in the pause menu	Un pause the game and close menu		
Pressing the save & quit button in the pause menu	Saves and returns the main menu		
Pressing the exit button in the main menu	Closes the game safely		
Pressing the saves button in the main menu	Oppens the save menu		
Opening a new save	Create a new game assigned to this save slot and boot it.		
Opening a preexisting save	Load into the last saved version of this save.		

Stage 4: weapons

This is where I will create the weapons and attacks the player will use against the enemies, I would put these next as they relate to the combat section already produced and allows for the testing of the next two stages. These should take 2 weeks at most.

Test	Expected outcome	Actual outcome	pass/fail
Attack button pressed	Do a quick attack		
Attack button held	Do a heavy attack		

Stage 5: enemies

This is where the base enemies the player fights against during their run will be made including their attacks and what they will reward the player with for defeating them. I will develop these here as it feels best to complete one world before moving on to the other and are more important than will then allow the testing of upgrades for the weapons. These also should only take 2 weeks

Test	Expected outcome	Actual outcome	pass/fail
Enemies' movement	Enemies move		

Enemies' attacks	Enemies can attack and do deal damage		
Enemies AI	Enemies try to get close to player and do different attacks		

Stage 6: upgrades

This is where I will create the upgrades the player can equip and modifications the player can do to their weapons like higher damage but slower or less range. I would put this stage next as it is the last part of the combat world and isn't needed for any of the previous parts so won't be missed allowing for a smooth programming flow. These should be easier than the weapons so should on take 2-2.5 weeks

Test	Expected outcome	Actual outcome	pass/fail
Player picks an upgrade	Upgrade is added to possible upgrades		
Player uses upgrade	Upgrades affect is produced		
Player equips upgrade	Upgrade can now be used and is added to equipped upgrades		
Player removes upgrade	Upgrade can no longer be used and is unequipped		

Stage 7: non-combat world

This is where the world the player returns to after they fail a run and where they will acquire the upgrades from the NPCs that live there and be given quests to increase their friendship. This is being put next as it is needed for the NPCs to live in and so must be completed before them. This should only take 1 week as it's not very technical with little programming

Test	Expected outcome	Actual outcome	pass/fail
Going into the world			

Leaving the world			
Changing upgrades			

Stage 8: interactable NPC

This is where I will make the NPCs that reward you from your quests including their dialogue. This is second to last as it shouldn't be too hard and has multiple assets that require other stages to have been completed first to even start working on. This is another big portion of the game so should be given 2.5-3 weeks.

Test	Expected outcome	Actual outcome	pass/fail
Talking to NPC	Appropriate dialogue		
Getting quest from NPC	Quest is given and added to ongoing quests list		
Completing quest	Quest is removed from list and rewards are given		
Adding new NPC	New NPC now spawns when you go to the camp		
Becoming friends with NPC	NPC gives player an upgrade		

Stage 9: boss & ending

This is where I will make the final boss of the game that will have to be defeated to finish the game. This will include multiple attacks and phases. I am putting this last as it seems like it could be complicated and isn't completely necessary for any of the other stages. This could be hard and so should be given 1-1.5 weeks at is only one enemy

Test	Expected outcome	Actual outcome	pass/fail
Boss movement	boss moves		
Boss attacks	boss can attack and do deal damage		
boss AI	Boss tries to get close to player and do different attacks		

Structure diagram



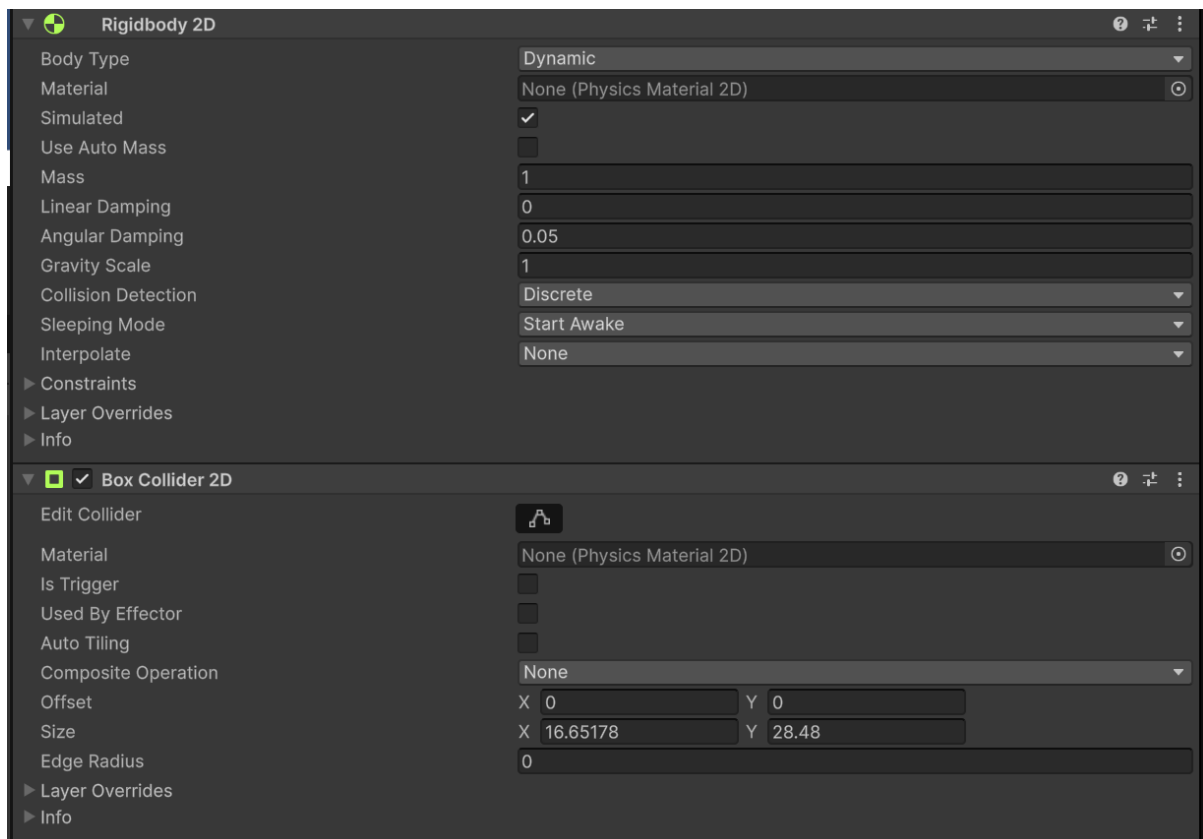
STAGE 1

This is the stage in which I develop the players movement as without this there isn't anything else that could be done as the player is the centre of the whole game. I will add a player with a sprite that can be moved by the movement controls built into unity (w,a,s,d and arrow keys). Then the camera will be centred on the player so that they can never go off screen.

Player Class diagram
-moveDirection: int -Grounded: bool -speed: int
+setGrounded() -move() -jump()

I started by adding the player sprites and a floor to stand on and giving them hitboxes so that they could interact. (sprites are temporary)





I then added a script to the player and started by initialising the variable

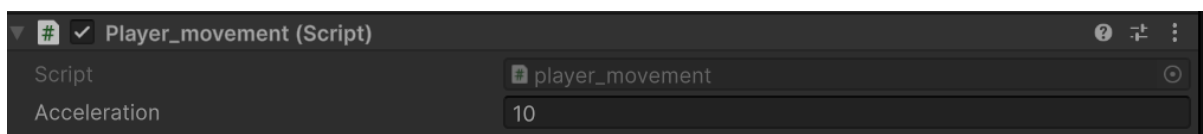
```
[SerializeField] private float x_speed;
[SerializeField] private float y_speed;
[SerializeField] private float acceleration;
```

I then added this line for horizontal movement

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

I used this code for movement as it is simple and works for multiple control schemes and so can be used by lots of people.

But when I tried to use it, I couldn't move. This was because when initialising acceleration, I forgot to give it a value meaning I was multiplying by zero which I fixed by giving it a value.



After that I made the coroutine for jumping and added the variable jumpForce.

```
[SerializeField] private float jumpForce;
```

```
private void Jump()
{
    if(Input.GetButtonDown("Jump"))
    {
        GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpForce), ForceMode2D.Impulse);
    }
}
```

The biggest problem was that you could jump anytime even in the air. I fixed this by adding a grounded variable that sets to true when on the ground.

```
if (Input.GetButtonDown("Jump") && Grounded)
{
    GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpForce), ForceMode2D.Impulse);
    Grounded = false;
}
```

And code that sets grounded to true when you touch the ground. Ive done it this way so that in future any new ground can be made into floor just by adding a tag.

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.collider.CompareTag("ground"))
    {
        Grounded = true;
    }
}
```

After that I wanted to make the camera follow the player so that they would always be in the centre of the screen.

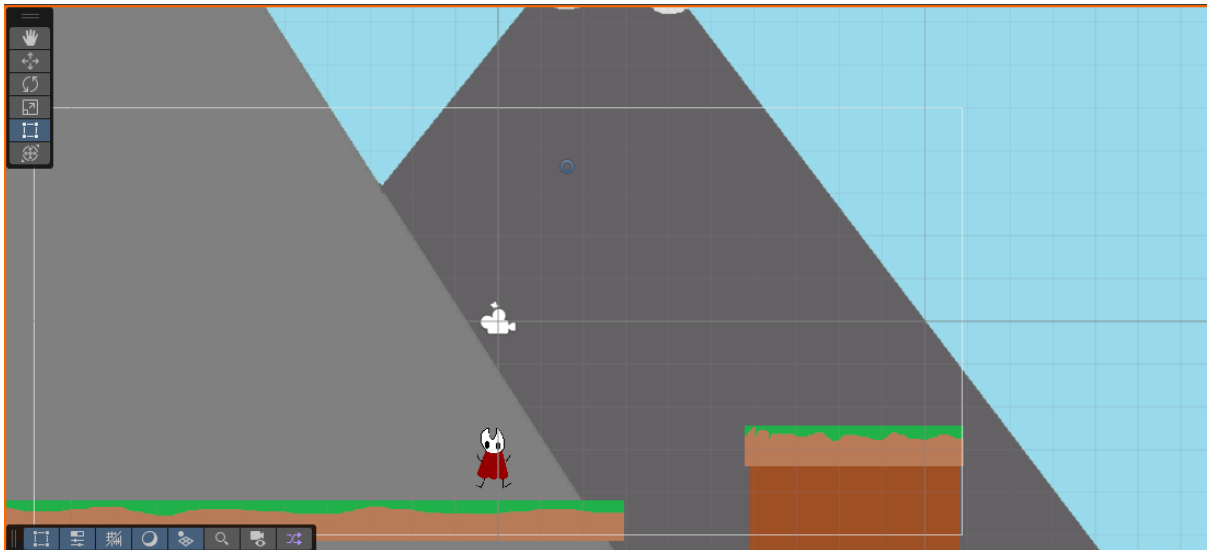
I started by initialising the variables for what to follow and where to go.

```
[SerializeField]
private Transform target;
private Vector3 wantedPosition;
```

I then wrote code that gets the targets position and modifies it so that they look good in the camera. It then goes to the new position.

```
void Update()
{
    wantedPosition = target.localPosition;
    wantedPosition.y = transform.position.y;
    transform.localPosition = new Vector3(wantedPosition.x, wantedPosition.y, -4);
}
```

I then proceeded to add some ground and more sprites that I drew for testing



Next I wanted the camera to follow the player up and down but not show the areas under the ground platform i did this by adding limits to the y movement of the camera instead of locking it to one height

```
if(wantedPosition.y<miny)
    wantedPosition.y = miny;
if(wantedPosition.y>maxy)
    wantedPosition.y = maxy;
```

This sets the height of the camera to the minimum or maximum height if it overrides it.

Which works well.

The last coding part of this phase was the dash. This would need to have a cooldown.

I tried to do this with a IEnumerator and the same code for jumping but that didn't work and would multiply the force each time you dashed for some reason

```
private IEnumerator dash()
{
    while(true){
        yield return new WaitForSeconds(1);
        if(Input.GetKeyDown(KeyCode.LeftShift))rb.AddForce(new Vector2(2, 0), ForceMode2D.Impulse);
    }
}
```

To fix this I made a new system for movement including x_speed and y_speed that would make it easier to do things like dashing in the future and allows you to control them separately.

```
[SerializeField] private Rigidbody2D rb;
```



```
private void Move()
{
    x_speed = Input.GetAxis("Horizontal") * acceleration;
    rb.linearVelocity = new Vector2(x_speed, rb.linearVelocity.y);
}
```

Instead of moving the player in increments the new code will add a speed in one direction or the other.

```
private void Jump()
{
    if (Input.GetButtonDown("Jump") && Grounded)
    {
        rb.linearVelocity = new Vector2(rb.linearVelocity.x, jumpForce);
        Grounded = false;
    }
}
```

This is the same for jumping as the velocity is decreased by gravity.

Now its dashing time:

For dashing all I did was change the x velocity to 2 times the max speed

```
private IEnumerator dash()
{
    rb.linearVelocity = new Vector2(x_speed*2, 0);
}
```

```
if (Input.GetKeyDown(KeyCode.LeftShift))
{
    StartCoroutine(routine: dash());
}
```

This didn't work because the frame after you dashed it would set the speed back to normal. I fixed this by only running the movement scripts if you weren't dashing as well as adding a timer for the dash.

```
private bool Dashing;
```

```

if (!Dashing)
{
    Move();
    Jump();
    if (Input.GetKeyDown(KeyCode.LeftShift))
    {
        StartCoroutine(routine: dash());
    }
}

```

```

private IEnumerator dash()
{
    Dashing = true;
    rb.linearVelocity = new Vector2(x_speed*2, 0);
    yield return new WaitForSeconds(0.2f);
    Dashing = false;
}

```

After that I needed to add a cooldown to the dash which I did by adding a wait for one second and a can dash variable

```

if (Input.GetKeyDown(KeyCode.LeftShift) && canDash)
{
    StartCoroutine(routine: dash());
}

```

```

private IEnumerator dash()
{
    canDash = false;
    Dashing = true;
    rb.linearVelocity = new Vector2(x_speed*2, 0);
    yield return new WaitForSeconds(0.2f);
    Dashing = false;
    yield return new WaitForSeconds(1);
    canDash = true;
}

```

This makes the player wait one second before they can dash again.

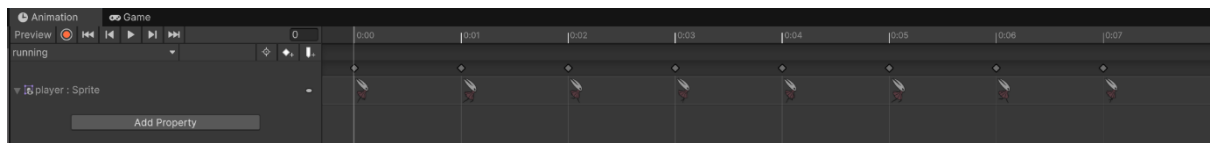
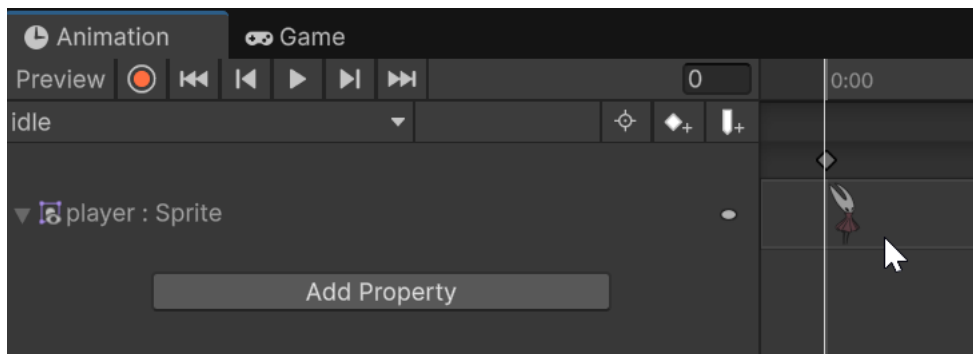
Finally, I wanted to give it a floating effect and so as the player dashes, I removed the gravity of the player to make a straight dash.

Now that the code was done, I needed to add animations:

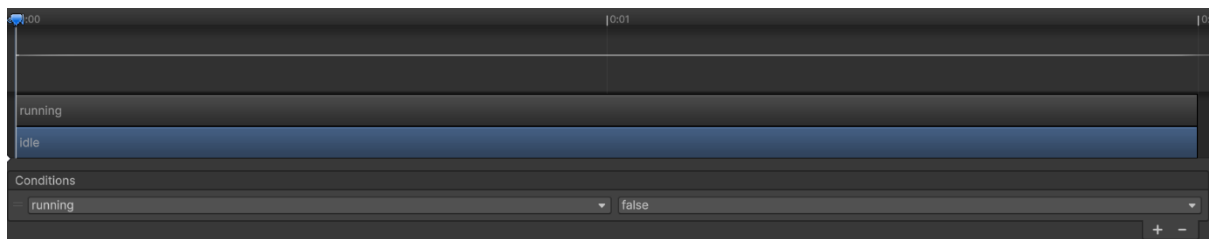
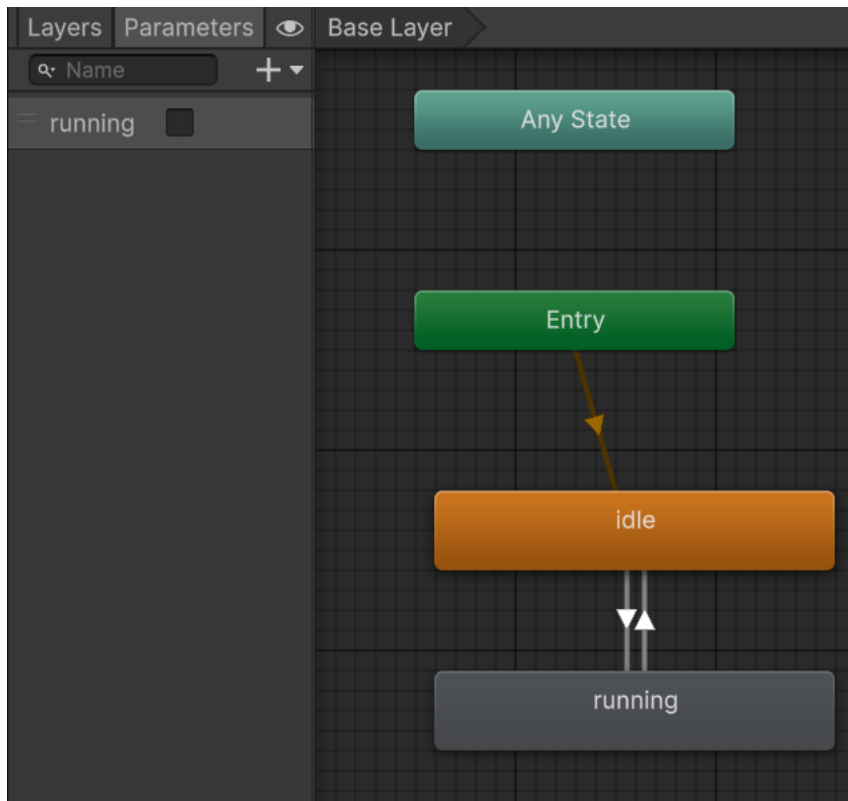
For now, I will be using sprites from Hollow Knight: silksong

(cherry, 2025)

I made an idle and a running animation



Then I added the rules for switching between the two by making a new Boolean for animations called running



Then in the code I added the conditions for switching between the two which is when the player has a speed attributed to them

```
if (x_speed != 0) animator.SetBool( name: "running", true);  
else animator.SetBool( name: "running", false);
```

Then to flip the sprite I added an if statement for which way the character is moving

```
if (x_speed > 0) sr.flipX = true;  
else sr.flipX = false;
```

This was actually the wrong way round, so the player was constantly moving backwards. So, I flipped the bigger than sign to fix it.

References