Research and Development for a 2D Game: CATMOSIS-Survival from Viruses and Plagues

BY

Abrar Mahir Antor

ID: 172-40-438

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Multimedia and Creative Technology

Supervised By

Md Salah Uddin

Assistant Professor and Research coordinator

Department of Multimedia and Creative Technology

Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

September 24, 2022

APPROVAL

This Project titled **"Research and Development for a 2D Game: CATMOSIS-Survival from Viruses and Plagues"**, submitted by Abrar Mahir Antor & Durjoy Paul to the Department of Multimedia and Creative Technology, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Multimedia and Creative Technology and approved as to its style and contents. The presentation has been held on 01th October 2022.

DECLARATION

I hereby declare that this project has been done by me and Durjoy Paul under the supervision of **Md Salah Uddin,** Assistant Professor and Research Coordinator**,** and the Department of Multimedia and Creative Technology at Daffodil International University. I also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

Salah

Md Salah Uddin
Assistant Professor and Research Coordinator
Department of Multimedia and Creative Technology
Daffodil International University

Submitted by:

[illegible]

Abrar Mahir Antor
ID: 172-40-438
Department of Multimedia and Creative Technology
Daffodil International University

ACKNOWLEDGEMENT

First and foremost, I want to express my heartfelt gratitude to almighty God for His wonderful grace, which has enabled us to successfully finish our final year project.

I am really grateful and wish our profound indebtedness to **Md. Salah Uddin**, Assistant Professor and Research Coordinator, Department of MCT Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of "Game Development" to carry out this project. His never-ending patience, intellectual direction, continual encouragement, constant and energetic supervision, constructive criticism, helpful suggestions, and reading many inferior drafts and revising them at all stages have made it possible to complete the final year project.

I would like to express my heartiest gratitude to **Dr. Shaikh Muhammad Allayear** - Professor and Head, Department of MCT, for his kind help to finish our project and also to other faculty members and the staff of the MCT department of Daffodil International University.

I would like to thank **Mr. Arif Ahmed, Dr. Md.Samaun Hasan, Mr. Mizanur Rahman,** and my entire coursemate at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, I must acknowledge with due respect the unwavering support and patients of my parents.

ABSTRACT

Game development and design is the process of creating an environment inside software that has the ability to create a virtual placeholder to run the virtual environment. In this virtual environment creating any kind of element that will entertain a person is a way of game development. But even if it does sound easier but creating that environment, one must have the knowledge of designing that environment from scratch and programming it with programing language. My passion for game development grew when I used to learn how to design the environment for animations, but this passion fully bloomed when my respected teacher MD Salah Uddin started teaching us the software called Unity that is used for game development worldwide. The final result of my work is an aesthetic 2 dimensional game with various perceptiveness and critical programming logic solved by visual scripting.

This also came with its own challenges as it has to be good in appearance, and also had to maintain a good storyline. Every detail has to be made by me and my teammate Durjoy Paul.

Unity has been used to develop this game, but there were also other applications and software such as Photoshop, Clip Studio Paint, Illustrator, and FL studios used to prepare the designs and audio.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Game Development is the craftwork of making games and defines the structure, evolution, and liberation of a game. It may implicate idea generation, strategy, construct, test and release. While we develop a game, it is essential to think about the game mechanics, prizes, player concentration, and level design.

A game creator could be a programmer, an audio engineer, an artist, a designer, or numerous different roles available in the industry.

Game Development can be launched by a large Game Development Studio or by a single person. It can be as small or big as we like. As long as it lets the player interact with range and can influence the game's features, you can anoint it as a 'game'.

You ought not to write code to get implicated in the Game Development method. Artists may produce and create assets, while developers might concentrate on programming a health bar. A Tester may get applied to see that the game functions as desired.

To settle issues that game frameworks had, mechanisms like libGDX and OpenGL were developed. They assisted game development to be a lot quicker and more comfortable, delivering lots of pre-made functions and components. However, it was always hard to enter the industry or comprehend a framework for someone reaching from a non-programmer background, a typical issue in the game development scene.

This is a mid-range project but it was not easy to work on this project because we were only two people working on this project with a limited amount of programming knowledge. Because of that, the project was consuming a lot of our time and we are always in danger of running out of time and reaching the deadline before finishing the project.

CHAPTER 2

BACKGROUND STUDY

Extensive research was conducted to perform the game development. To achieve the outstanding outcome we had to go through a lot of classic and modern games to learn what technology we needed to use in our game and what kind of art style we should be following to come up with a decent comfortable environment with a fluid motion to everything. Generally, it was supposed to be a story game with a small amount of playing which we made at first and we decided to make it a full fledge game with so many actions and a fluid cinematic And for that reason, we had to do a lot of study on animations and smooth art styles, lightings, environment and on numerous things.

We downloaded a lot of games those has a similar concept as our game and we had to play them for so long to understand the physics. We even extracted multiple open-source games to understand how they programmed them. It may sound like an easy thing to do but we almost lost our sanity while working on this process.

But we managed to figure out the elements we needed and were able to apply them in our game properly.

CHAPTER 3

DETAILS OF SOFTWARE

3.1: Unity(Game Engine):

Unity is a cross-platform game engine created by Unity Technologies, first promoted and liberated in 2005 at Worldwide Developers Conference as a game engine. The engine has heretofore stood slowly developed to sustain a mixture of desktop, mobile, console, and virtual reality platforms. It is extremely famous for IOS and Android mobile game development and is deemed comfortable to use for beginner developers and is famous for indie game development.
The engine can be utilized to make three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations and different experiences. The engine has been assumed by enterprises beyond video gaming, such as film, automotive, architecture, engineering, construction, and the United States Armed Forces.

Fig 3.1: Unity (Game Engine)

3.2: Clip Studio Paint:

Clip Studio Paint (formerly sold as Manga Studio in North America), known in Japan as Kurisuta (クリスタ), is a home software application created by Japanese graphics software company Celsys. It is utilized for the digital design of comics, general illustrations, and 2D animation. The software is obtainable in versions for macOS, Windows, iOS, iPad, Android, and Chrome OS.

The application is marketed in editions with varying attribute sets. The full-featured edition is a page-based, layered illustration program, with help for bitmap and vector art, text, imported 3D models, and frame-by-frame animation. It is developed for use with a stylus and a graphics tablet or tablet computer. It has graphic tools which mimic natural media such as pencils, ink pens, and brushes, as well as designs and ornaments. It is determined from parallel programs by features designed for making comics: tools for creating panel layouts, outlook rulers, sketching, inking, applying tones and textures, coloring, and creating word balloons.

Fig 3.2: Clip Studio Paint

3.3:Adobe Photoshop:

Adobe Photoshop which is a raster photo editor was developed and posted through the company of Adobe Inc. for the users of Windows and macOS. It does both rasters of photos and edits the textual content or renders it and vector photos via clipping route and 3D photos and videos as well. Plugins enhance the character of the software.

Photoshop from the start was able to documents in different formats, which include TIF, PEG, and GIF. These documents are smaller in comparison to the PSD documents because they do not have the editing capabilities of PSD files.

Fig 3.3: Adobe Photoshop

CHAPTER 4

PROJECT WORKFLOW

4.1: Generating Idea:

Generating the idea of what kind of game we should make was a tough choice. We went through a lot of brainstorming seasons and changed our plan for the game's story and the designs several times. It took down a lot of our sanity and we had a very hard time deciding what we should be making and how other people will accept it.

At first, we came up with the idea of a pandemic situation story-lined game that will teach people how they can survive a pandemic related to viruses and plagues. But we dropped this plan after making a prototype of the game and discussing the outcome with our mentors.

After that, we had a clear vision of what we wanted to make. We decided on keeping the same idea of a pandemic but with more story and action twists.

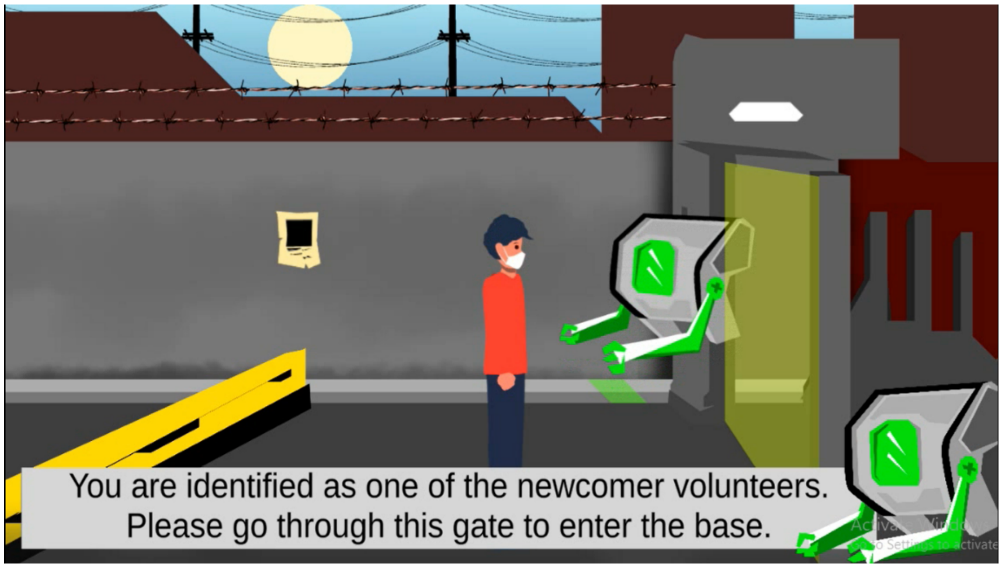
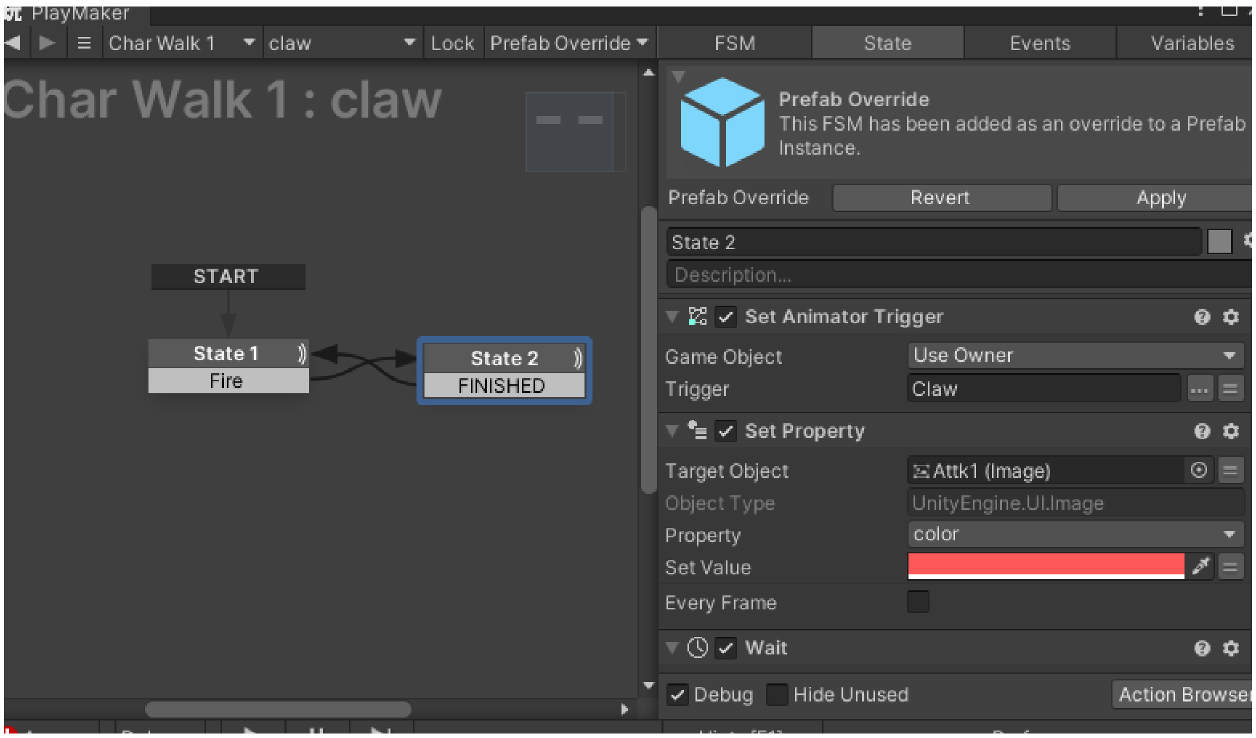

You are identified as one of the newcomer volunteers.
Please go through this gate to enter the base.

Fig 4.1: Cancelled Prototype

4.2: Creating a prototype:

We created a prototype of our game at first with low fidelity arts with simple shapes without having proper light and shadows and decided on how it should be doing. We divided both of our works, since I am good at visual programming I used visual scripting by Playmaker Plugin to work on this which saved a lot of our time.



Prefab Override
This FSM has been added as an override to a Prefab Instance.

State 2
Description...

Set Animator Trigger
Game Object: Use Owner
Trigger: Claw

Set Property
Target Object: Attk1 (Image)
Object Type: UnityEngine.UI.Image
Property: color
Set Value: 🟥
Every Frame: ☐

Wait

✓ Debug ☐ Hide Unused

Fig 4.2: Playmaker Visual Scripting

4.3: Game Design:

After creating a prototype we started to work on the main game. We started with preparing the characters and level design. We divided the work and my teammate Durjoy Paul was in charge of the illustration of the game and I was in charge of all other staff. Here is a small preview of his works on the design below, you can see the whole design workflow in his report.

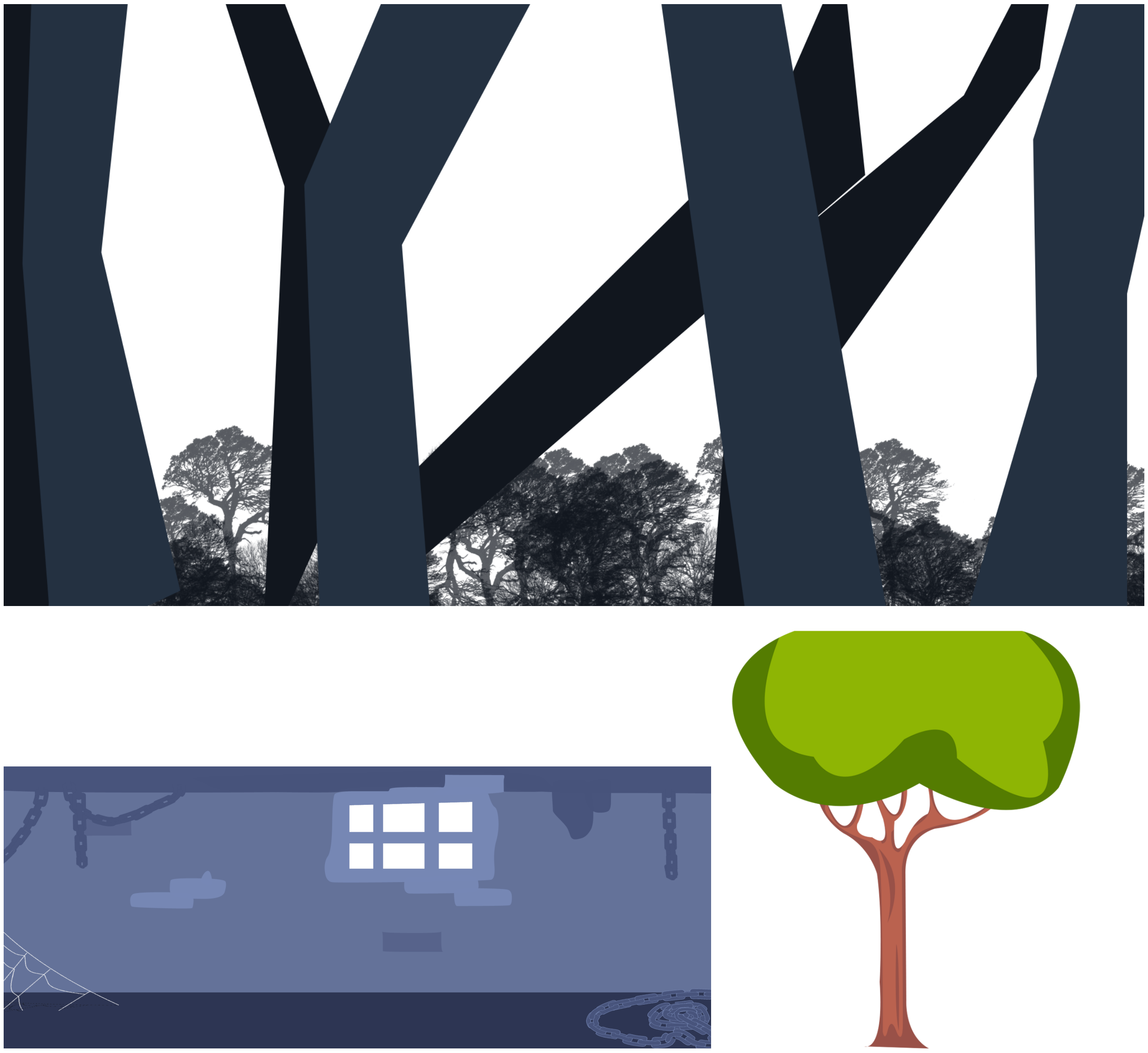


Fig 4.3: Game Design

4.4:Setting up the project

In the meantime of preparing the design of the game, I was working on setting up the unity project with placeholders(Blank 2D objects that can be placed inside a unity project to place the main designs thereby replacing them when the design work is done). I used unity 3D Instead of Unity 2D to pull up the effect of protectiveness without using a script.

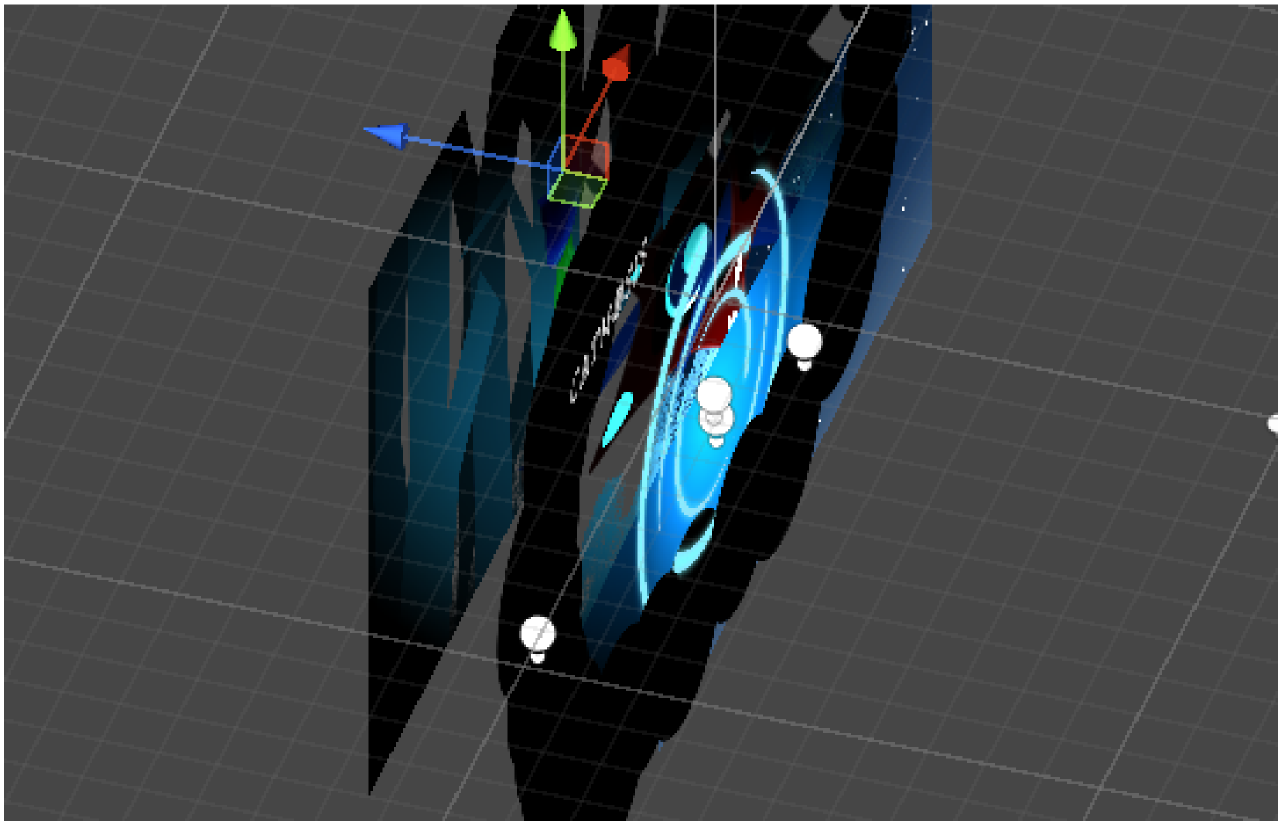


Fig 4.4: Blank PlaceHolder

4.5:Replacing the PlaceHolders:

After one of the level's designs is ready I replaced the placeholders with those designs. Every placeholder had different shapes and locations so after replacing those I had to tweak their size and positions according to what we wanted. The output after the replacement looked something like this.

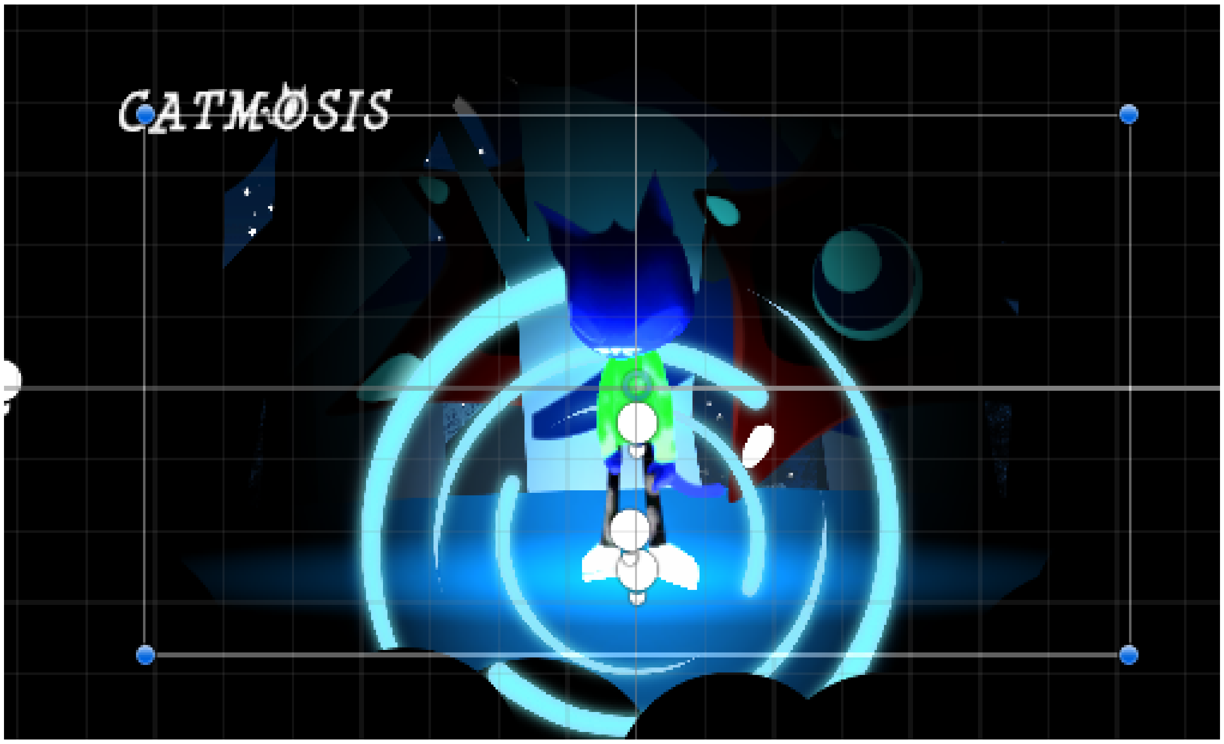


Fig 4.5: Replacing the PlaceHolder

In Figure 4.5 you can see that all the placeholders have been replaced with the main designs of the game. They still look weird here because they haven't been animated and programmed yet to be the way they are supposed to be. For example, the glowing circle behind the character is supposed to be a rotating magic circle that you will see in the game which will keep rotating under the feet of the main character horizontally on the loading screen.

4.6.1:Creating the loading screen(Character Rigging):

After placing all the arts in the workspace It was time for the animation. Now to save our time instead of using the traditional sprite frame drawing method we used the Sprite editing method to rig the characters with 2D bones And used a unity animator to animate them. The process of this is really hard but less time-consuming. So I started with rigging the main character. I went to the character's sprites properties and then Clicked on the sprite editor and I started placing bones on the sprites using the skinning editor.

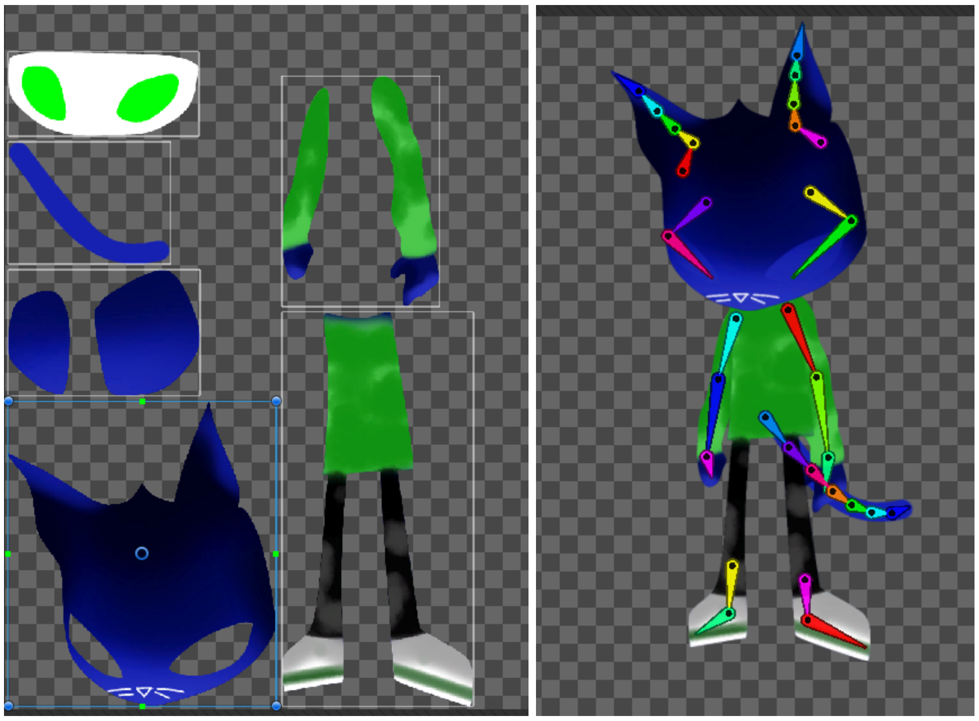


Fig 4.6.1: Sprite Editing

The process of this skinning editor is similar to the process we follow in 3D modeling where after creating the bone we have to bake all those bones into the sprite's weight. After that, we can move the character by just moving the bones.

4.6.2:Creating the loading screen(Character animating):

After rigging the character it was time to animate the character. To animate the character I used the built-in animator of unity. At first, I opened the animator and created a new animation file which also imported the animator component inside the character with the animation. Now I used the keyframing to animate the character, I didn't use is kinematics on this character since it is just a dummy of the main playable character.

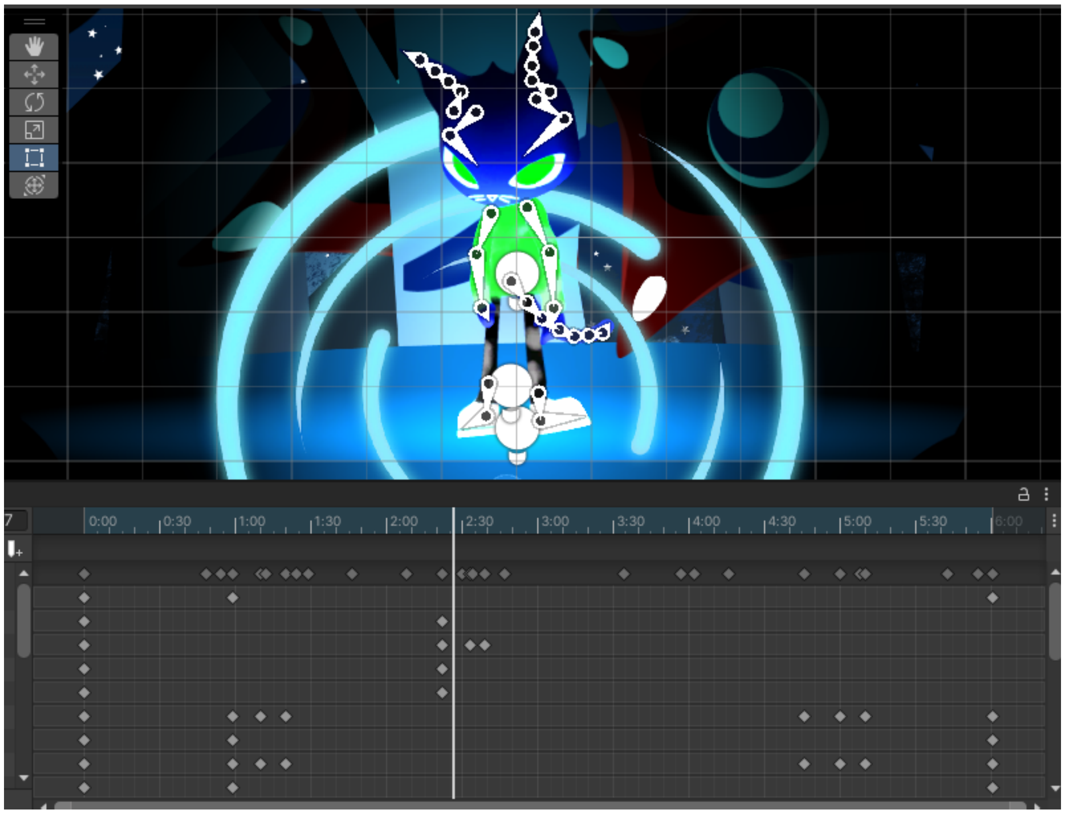


Fig 4.6.2: Animating

After finishing the character animation I set this animation as the default layer to give it an idle loop without any scripting inside the animation control panel.

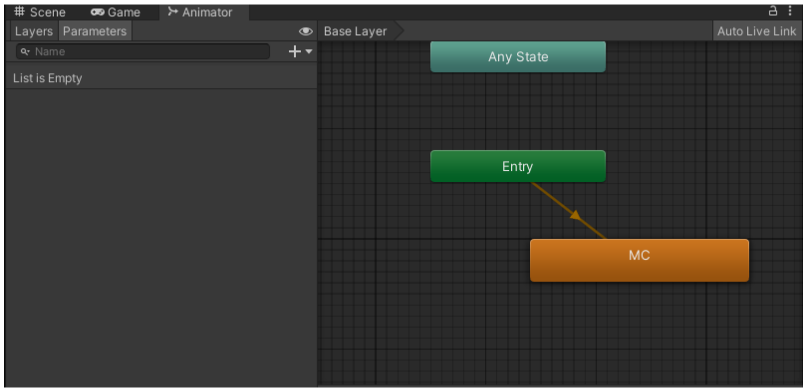
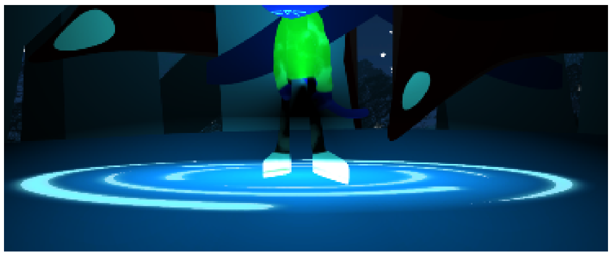
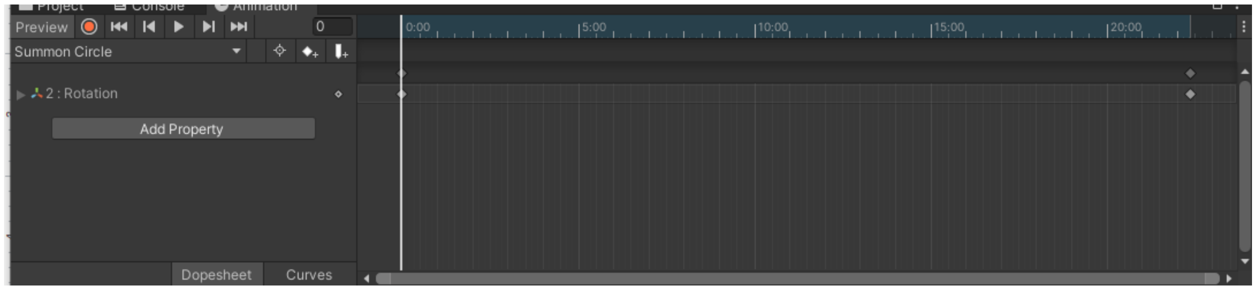


Fig 4.6.2: Animation Cpanel

4.6.3:Creating the loading screen(Magic Circle Animation):

After animating the character I animate the magic circle in the same way but this time I didn't have to rig it and instead I picked the cframe of this object and keyed all the rotations numbers.





After that, I set its rotation animation to loop by setting the animation as default inside the animator control panel.

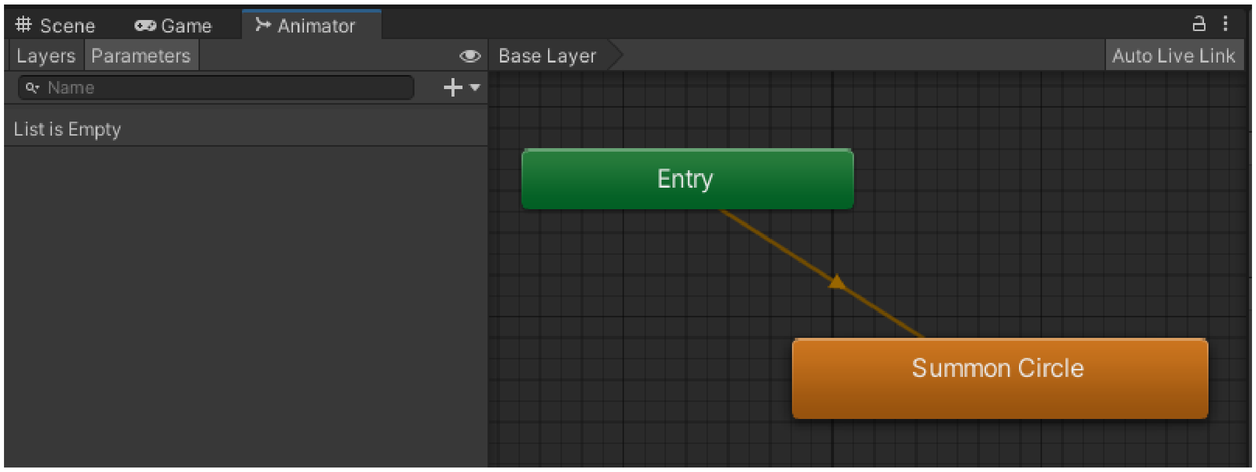


Fig 4.6.3: Animation Cpanel and circle loop

I also had to work a lot inside the properties of the circle such as tweaking its order in layers which I will talk about in the upcoming content.

4.6.4:Creating the loading screen(Soul Orb):

After animating the magic circle the big challenge was creating a glowing orb that will glide on the screen randomly and also emit lights to the nearby objects. So at first, I created a default circle object inside the workspace and after that, I placed a spotlight2D inside it and tweaked the property of the light. After that, I used the animation editor to animate its position for a long period to give it a random animation flow.

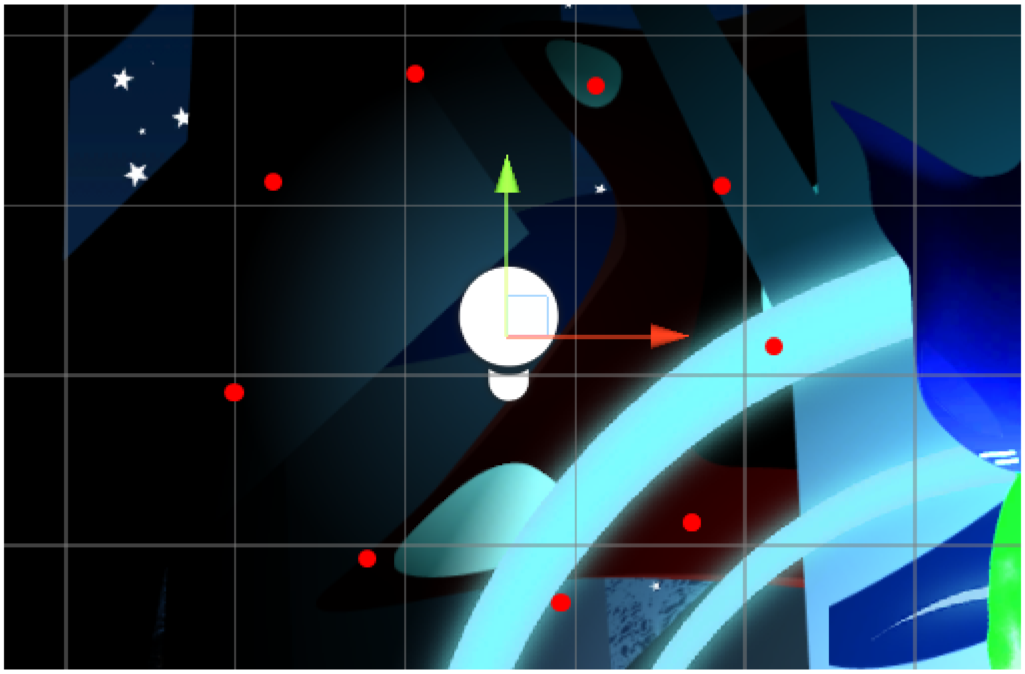


Fig 4.6.4: Soul Orb Light

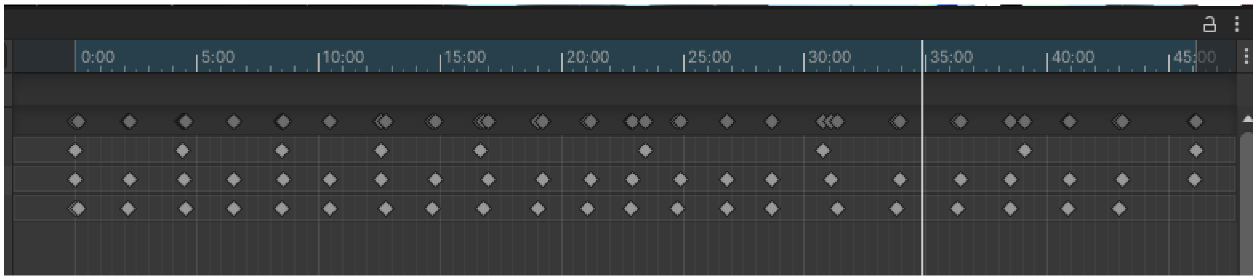


Fig 4.6.4: Orb Animation

4.6.5:Creating the loading screen(Background Monster eye and tentacle animation):

Animation of the monster eye and its tentacle was difficult and since there are two of them inside the loading screen it might look like I can just copy pasted the monsters but that wasn't the case since in unity duplicating any object will use the same animation control panel so had to recreate the whole monster again to have a duplicate of it. And for the monster, I had to rig it with sprite editor and also had to animate it with animation editor.

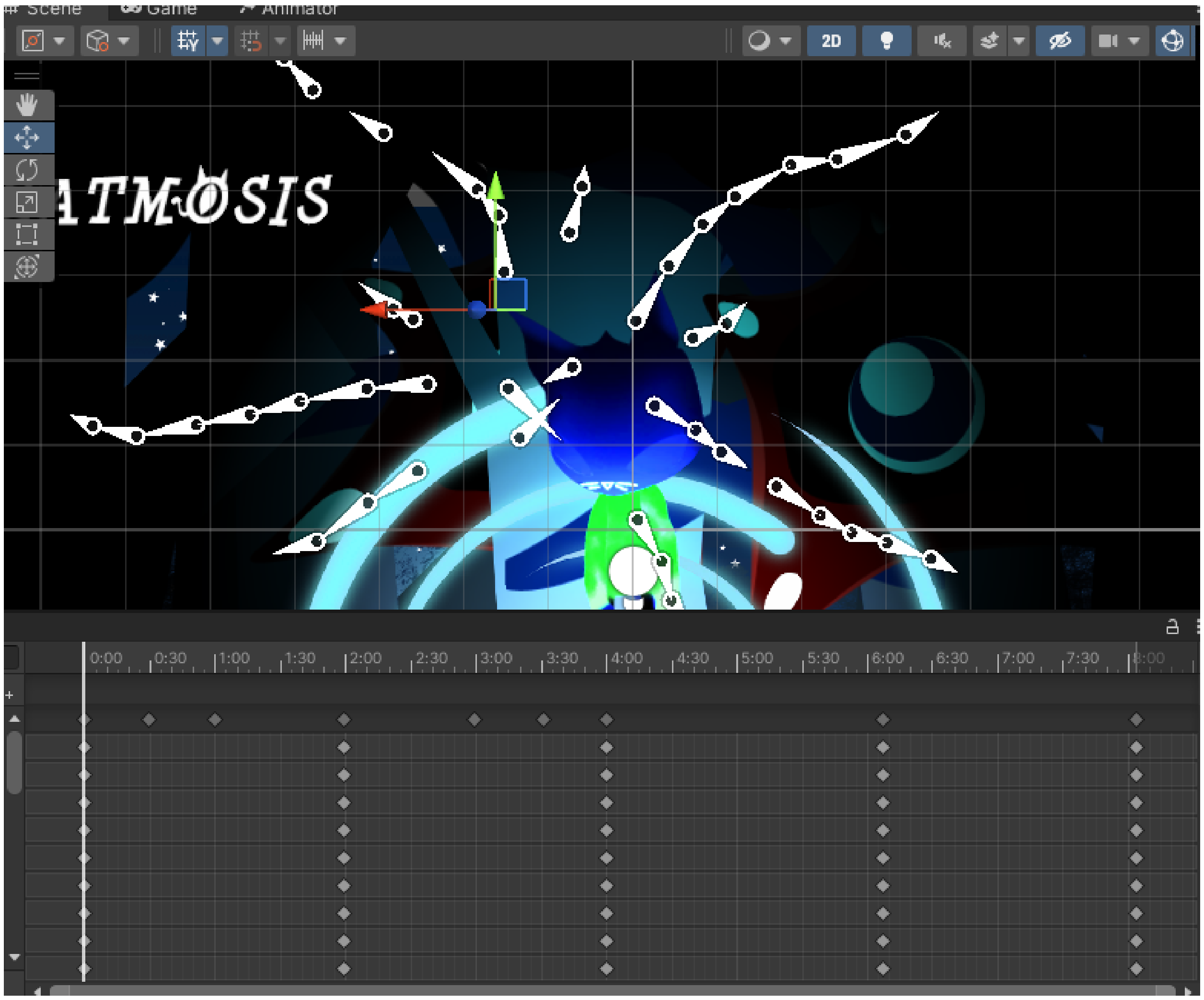
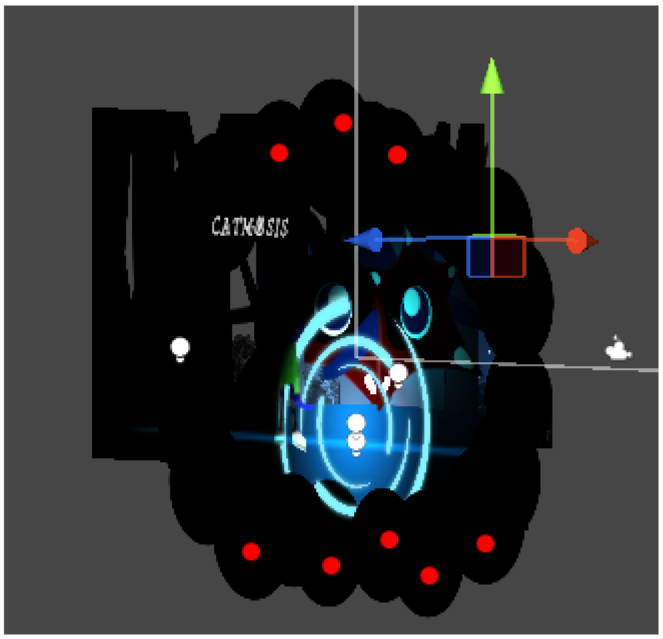


Fig 4.6.5: Monster Rig and animation

4.6.6:Creating the loading screen(Dark Shadow border creation):

The final animation in the loading screen was the dark shadow border. To achieve this effect I had to go through brainstorming and I created a bunch of black circles and animated their scale. Now I had to animate 5of every circle separately so that I can scramble them around the border to have a random effect. You can see the effect when you will start the game. Here is how they look in the workspace marked as red dots:



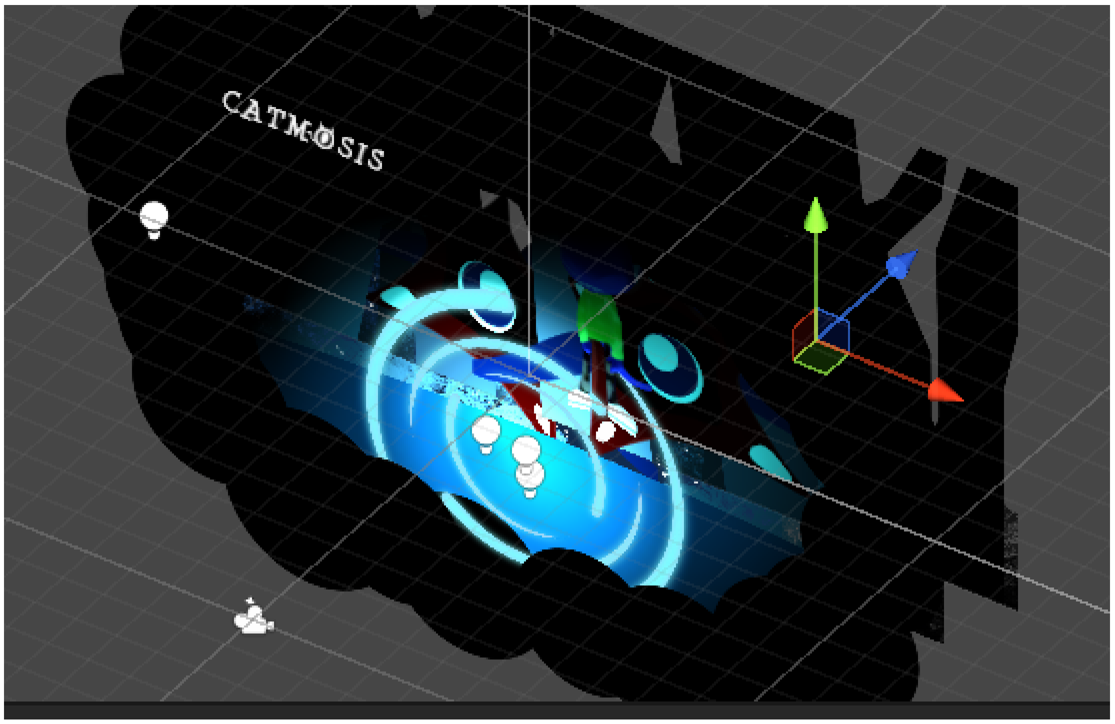


Fig 4.6.6: Dark Shadow Border Creation

4.6.7:Creating the Main Menu(Light Setup):

To light up the scene we first took the advantage of unity 3D, in unity 2D lights basically cover the whole area but in unity 3D even the 2D lights have perspectives and depth. Since the main menu is in night mode we didn't need any ambient lighting, I used the magic circle as our main light source and the soul orb as our secondary light source. I had to tweak the magic circle light source in several ways to pull out the output that we wanted.

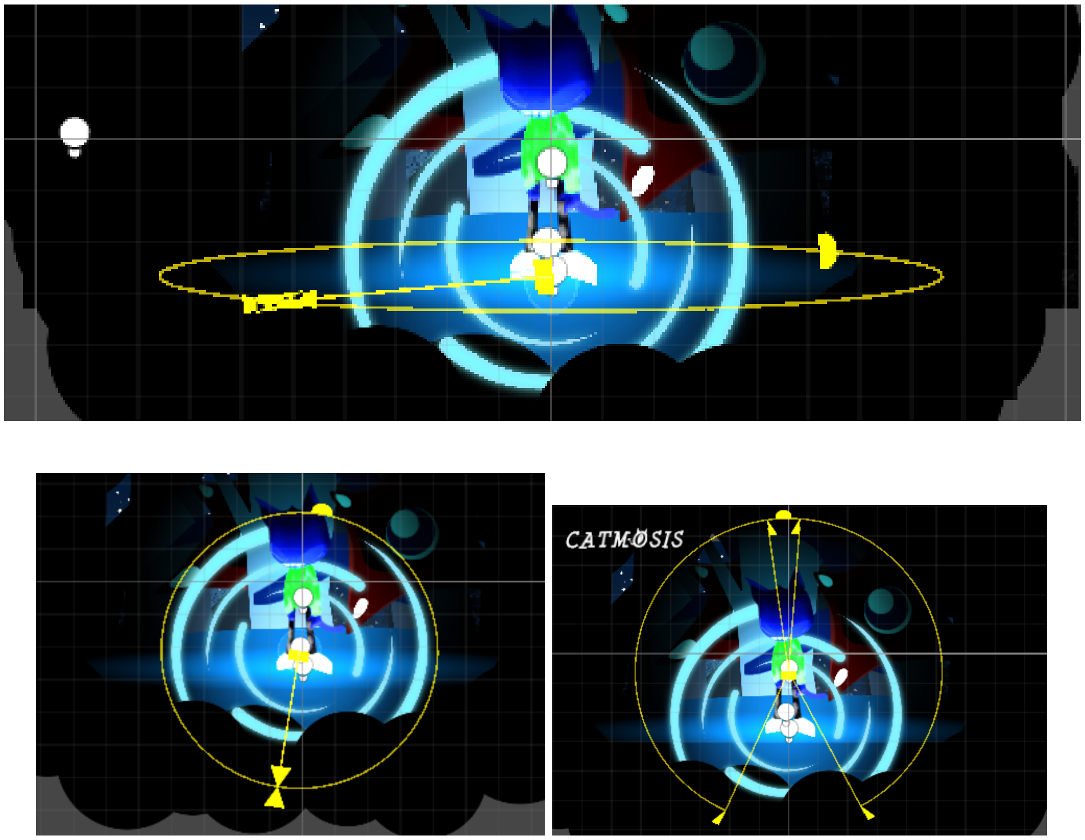


Fig 4.6.7: Light Setup

4.6.8:Creating the Main Menu(Layers):

Setting up the layer properly was a big challenge since the camera doesn't render things properly when it comes to manipulation between 2D and 3D. So I had to spend hours tweaking out the perfect view of layers. I have used multiple layers in the Main menu such as sky, background trees, foreground trees, monsters, characters, etc. Here is an example:

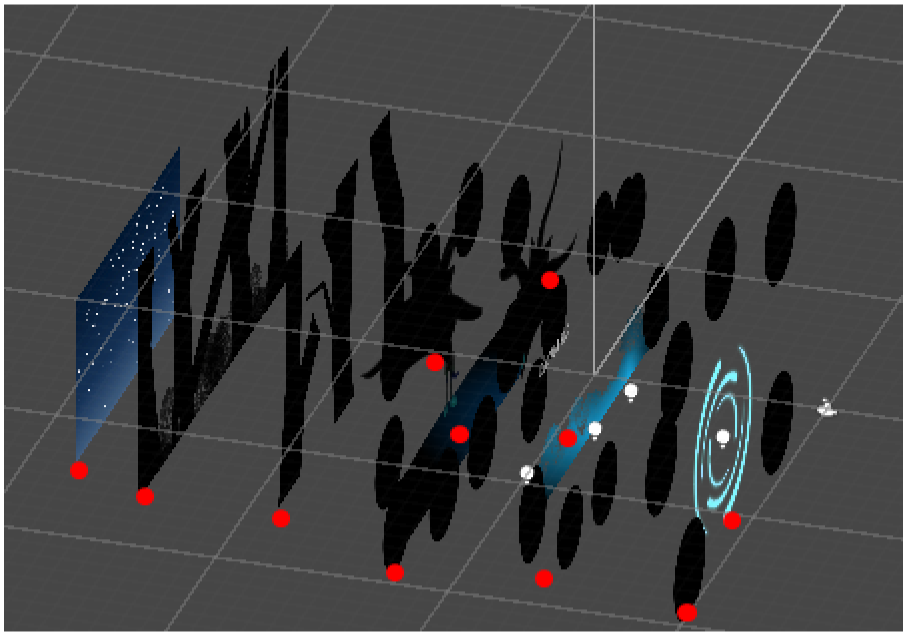


Fig 4.6.8: Layers

4.6.9:Creating the Main Menu(UI):

UI is one of the most important parts of a game. Now our UI is under development just like our project since a game project can take multiple years to build it is really hard to cover up everything in a small time but also with a good output. So we have prepared as much UI as we needed to run the game properly. So inside the main menu, I have added a game logo and a new game button to start the game.

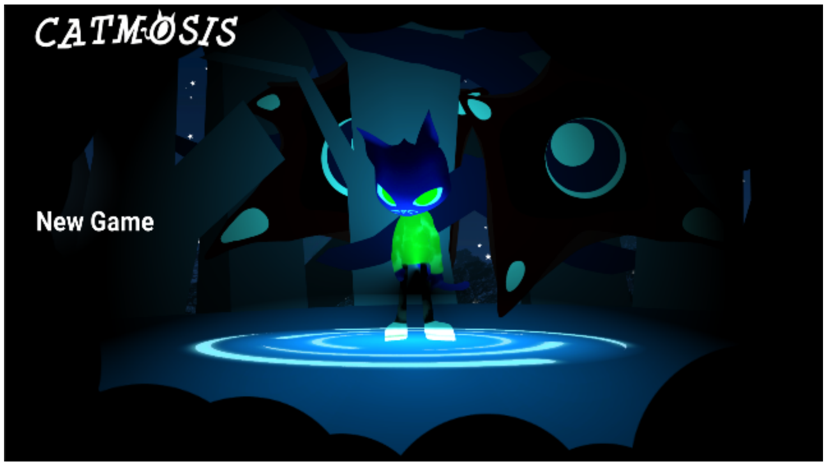
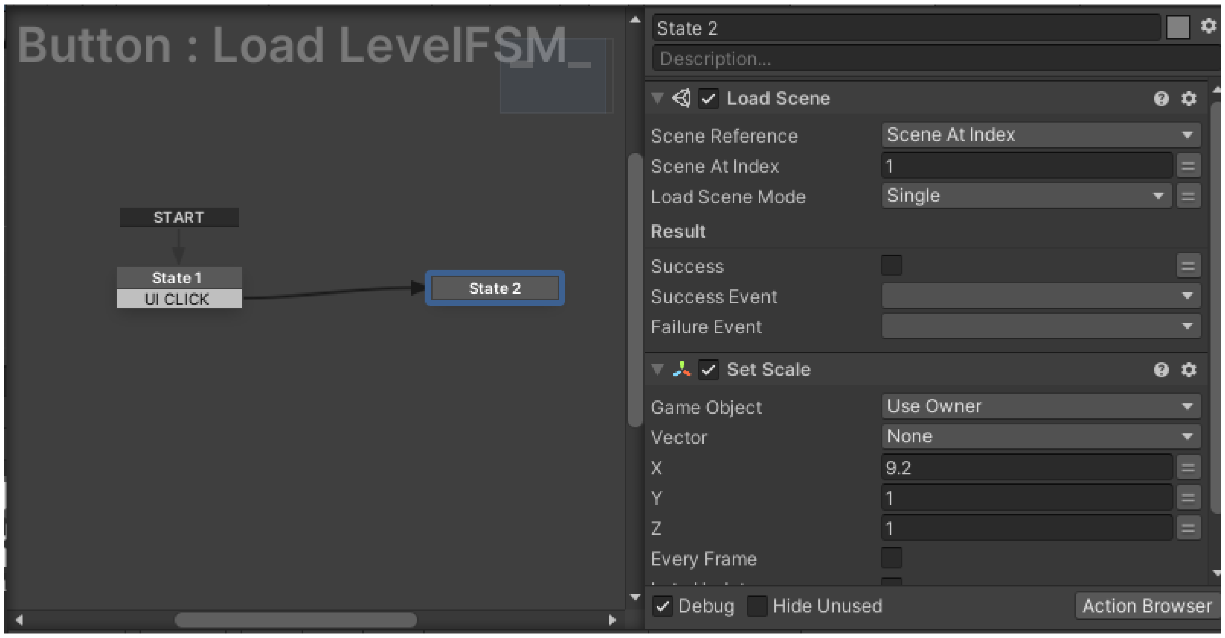


Fig 4.6.8: UI

The New Game button has two visual scripts inside it, one is to animate the button and the other one is to load the scene it is set to load.



State 2

Description...

Load Scene

Scene Reference	Scene At Index
Scene At Index	1
Load Scene Mode	Single
Result	
Success	
Success Event	
Failure Event	

Set Scale

Game Object	Use Owner
Vector	None
X	9.2
Y	1
Z	1
Every Frame	

Fig 4.6.8: Load Scene

4.7:Creating the Level Selection(Setting Up):

The level selection scene was a complicated one because I had to create a mouse hovering effect that make the slots of level rotate when the mouse is on the slot and it also follows the direction where the cursor moves. Here is an example of it:

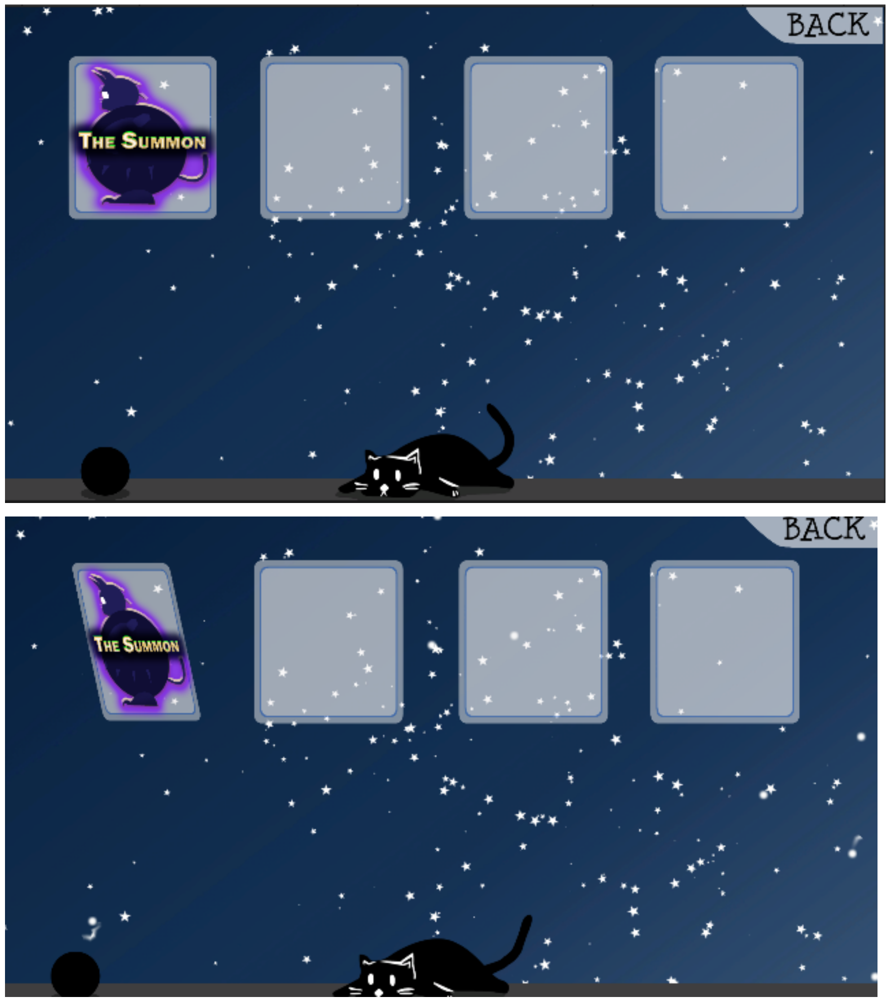


Fig 4.7: Level Selection

The cat at the bottom has animations and the black ball is an easter egg to the game which I will explain later. Selecting the level card will start the first level scene. And pressing the back button will take the player back to the main menu.

4.7.1:Creating the Level Selection(Layers):

The level selection menu does not have much of layers, most of the works are done in the canvas section. There are only the cards and the back button inside the workspace to make it easier to move freely according to the mouse.

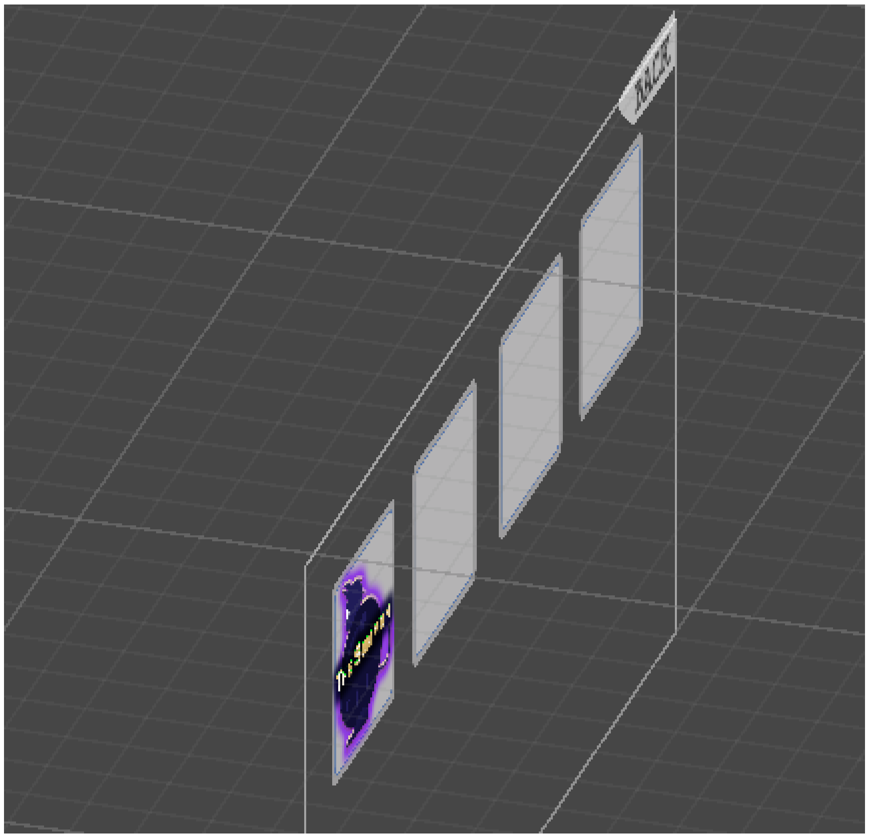


Fig 4.7.1: Layers

Everything else is inside the canvas section including all the scripts and other animated staffs.

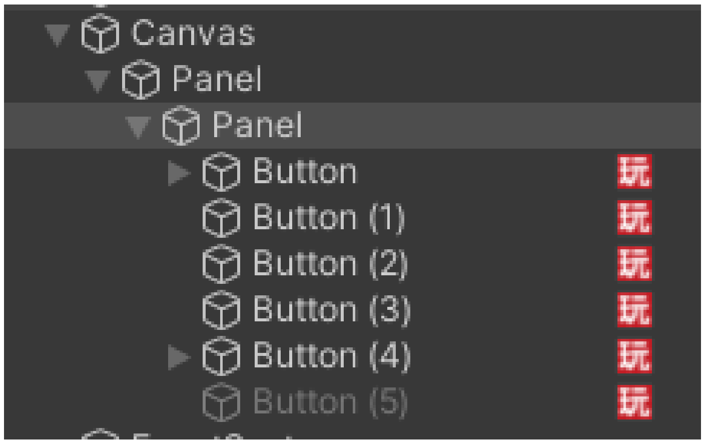


Fig 4.7.1: Canvas Layers

4.7.2:Creating the Level Selection(Card Mouse Hover Effect):

Cards/Level selection button has an effect that the card rotates when the mouse is on the slot and it also follows the direction where the cursor moves. To achieve this effect I had to create an image in the workspace that will be able to move freely and then make it parented to a button inside the canvas at first like this:

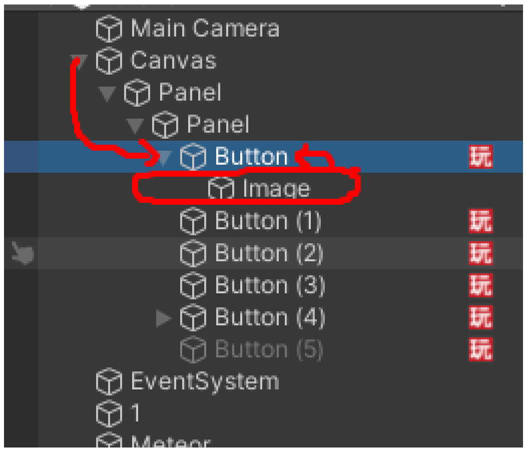
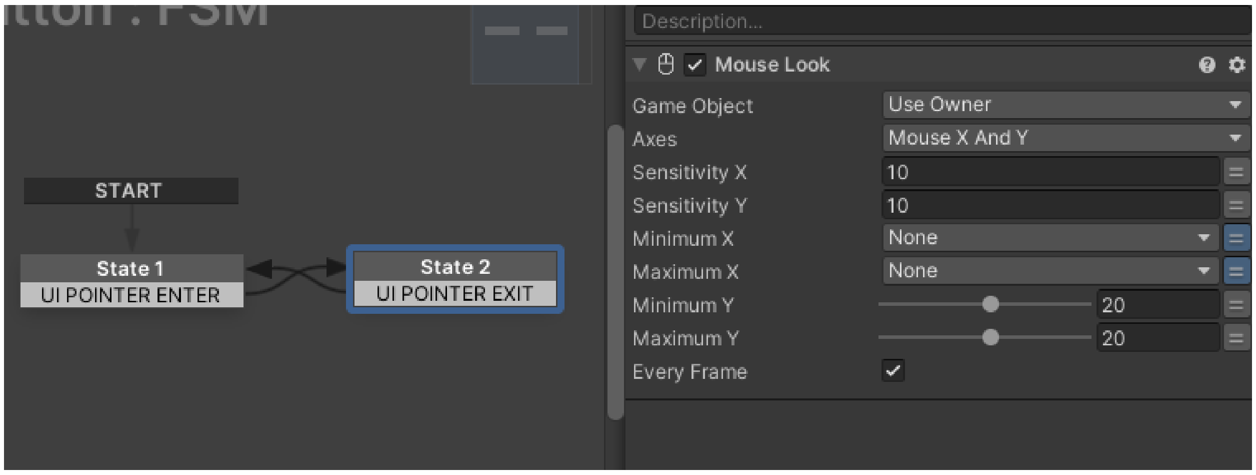


Fig 4.7.2: Image parented to button

After that I had to set a few scripts inside the button and the core part of the script was this mouse look function which determines where the mouse cursor location is.



Game Object	Use Owner
Axes	Mouse X And Y
Sensitivity X	10
Sensitivity Y	10
Minimum X	None
Maximum X	None
Minimum Y	20
Maximum Y	20
Every Frame	✓

Fig 4.7.2:Mouse look at

4.7.3:Creating the Level Selection(Cat Animation):

Cat at the bottom of the scene has some rigging and animations as well. I followed the same process of animation I did before to animate the cat and then I put it on loop.

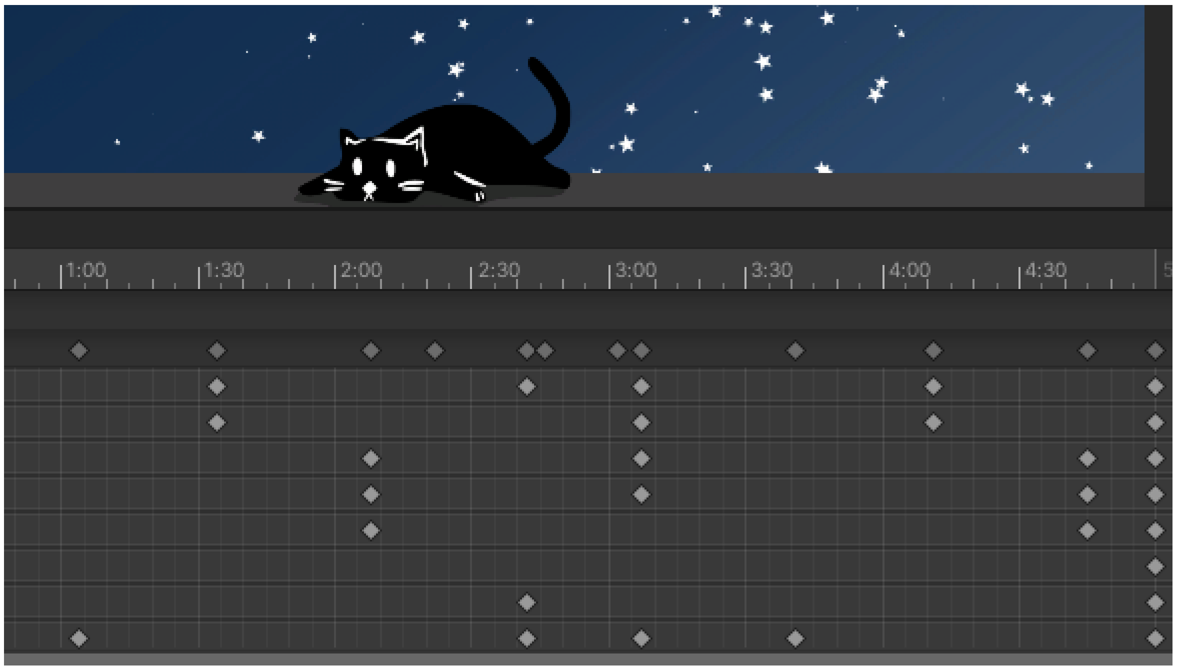


Fig 4.7.3:Cat Animation

4.7.4:Creating the Level Selection(Ball Easteregg):

I have added a ball that has some secret function in it. Normally it looks like a simple ball that is moving around colliding with cats and going back and forth but if a player clicks on that ball it will become a living monster that has an animated big red eye.

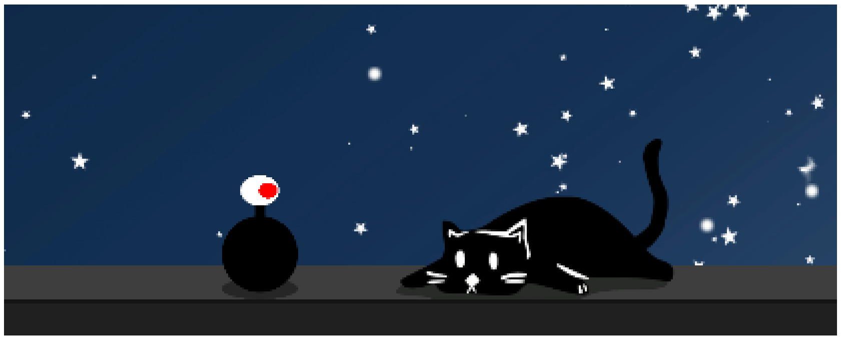
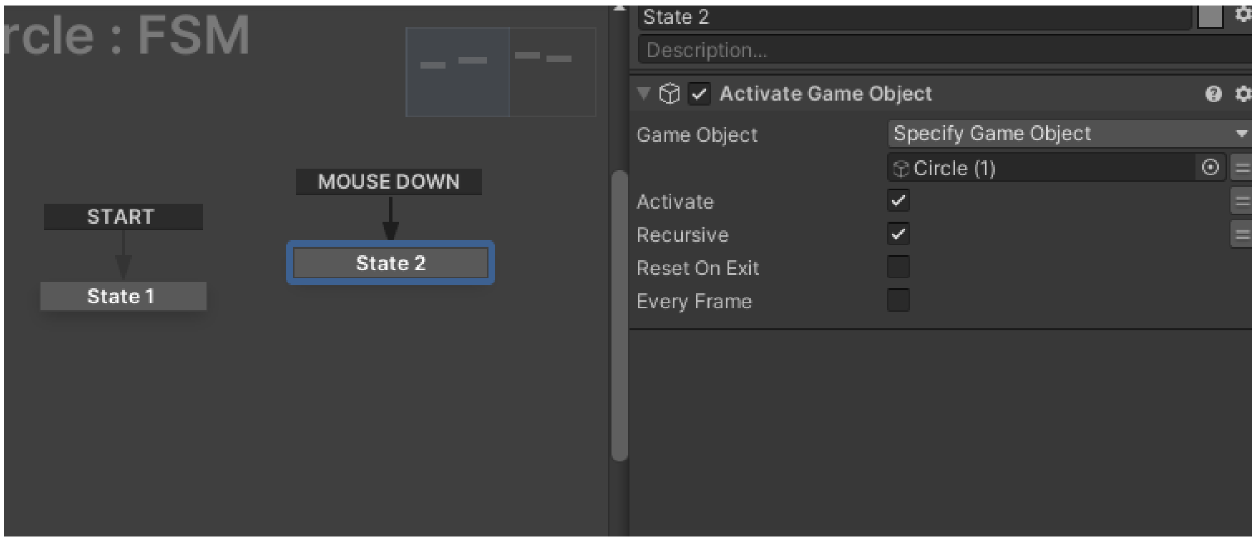


Fig 4.7.4:Ball Easter Egg

So the ball basically has a moving animation and a script in it that detects the mouse click on the ball and activates the eye part which is parented to the ball.



State 2
Description...
Activate Game Object
Game Object: Specify Game Object
Circle (1)
Activate: ✓
Recursive: ✓
Reset On Exit:
Every Frame:

Fig 4.7.4:Ball Easter Egg Script

4.8:Creating the Main Game (Chinamatic):
After the player selects the game, a cinematic will show up and it will start explaining to the player what he has to do in the game. The cinematic has tons of work inside it from camera manipulation to lots of animations and particle effects.

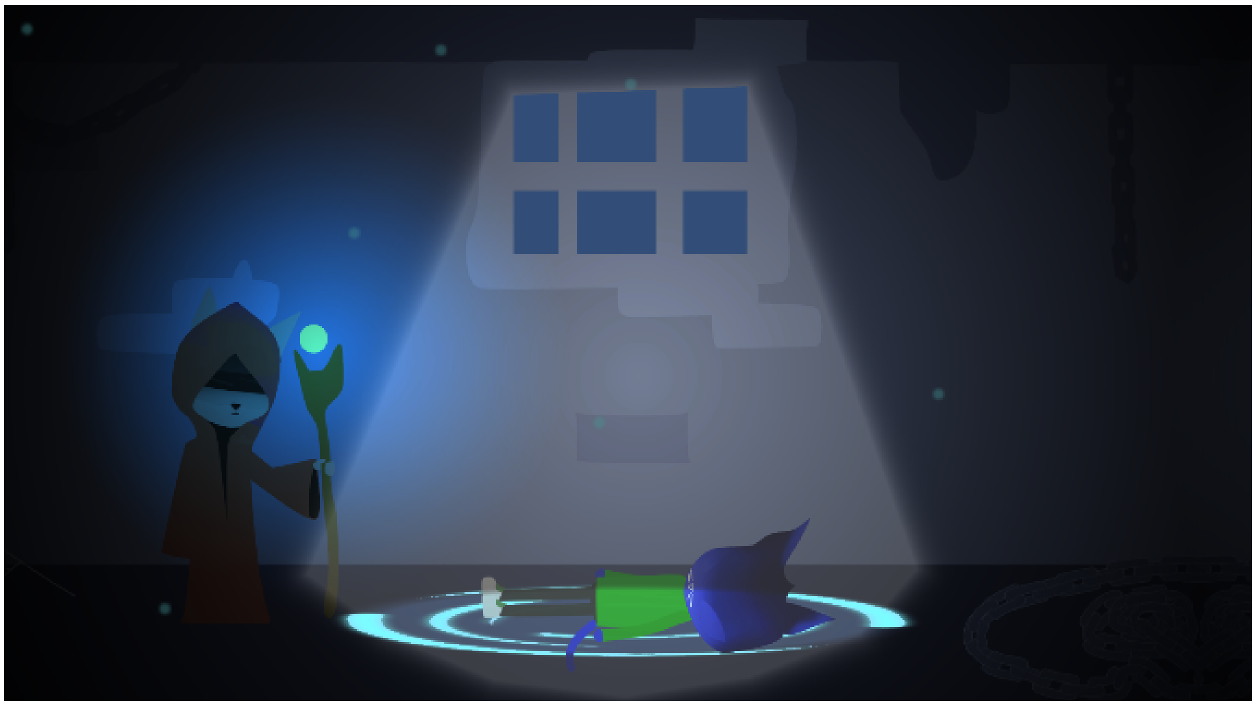


Fig 4.8: Cinematic

4.8.1:Creating the Main Game (Chinemetic Layers):
The cinematic layer is made with about 7 layers. The backgrounds, characters, props, etc. The cinematic was mainly focused on lighting and animation to pull out the rune vibe we wanted.

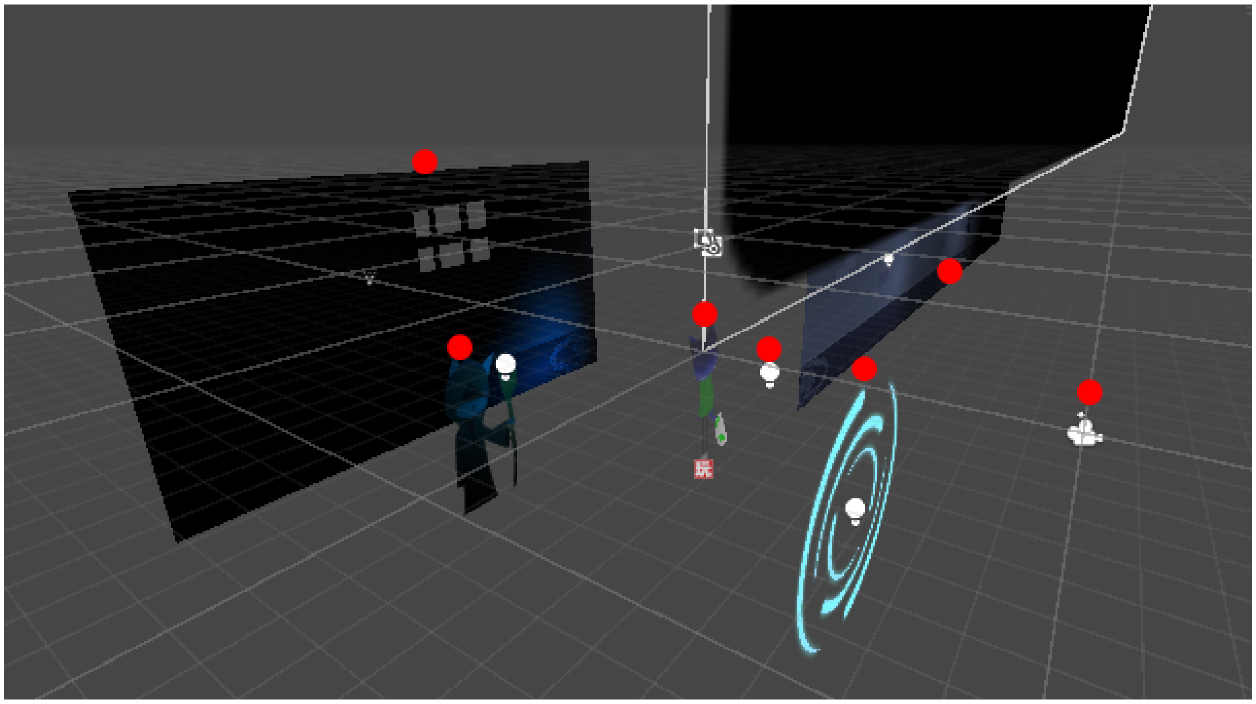


Fig 4.9: Cinematic Layers

4.8.2:Creating the Main Game (Main character building):

The cinematic has a different and more agile character build than the main menu character, this one was built differently which you can see in my teammates report. I had to rig and animate this character from the beginning and animate it several times for the cinematic, But I also had to prepare it as a playable character where I can make it idle, walk, run, and fight. And it was a pretty big challenge doing this for the first time. Here is all the rig and animation I had to make for this character in chinemetic.

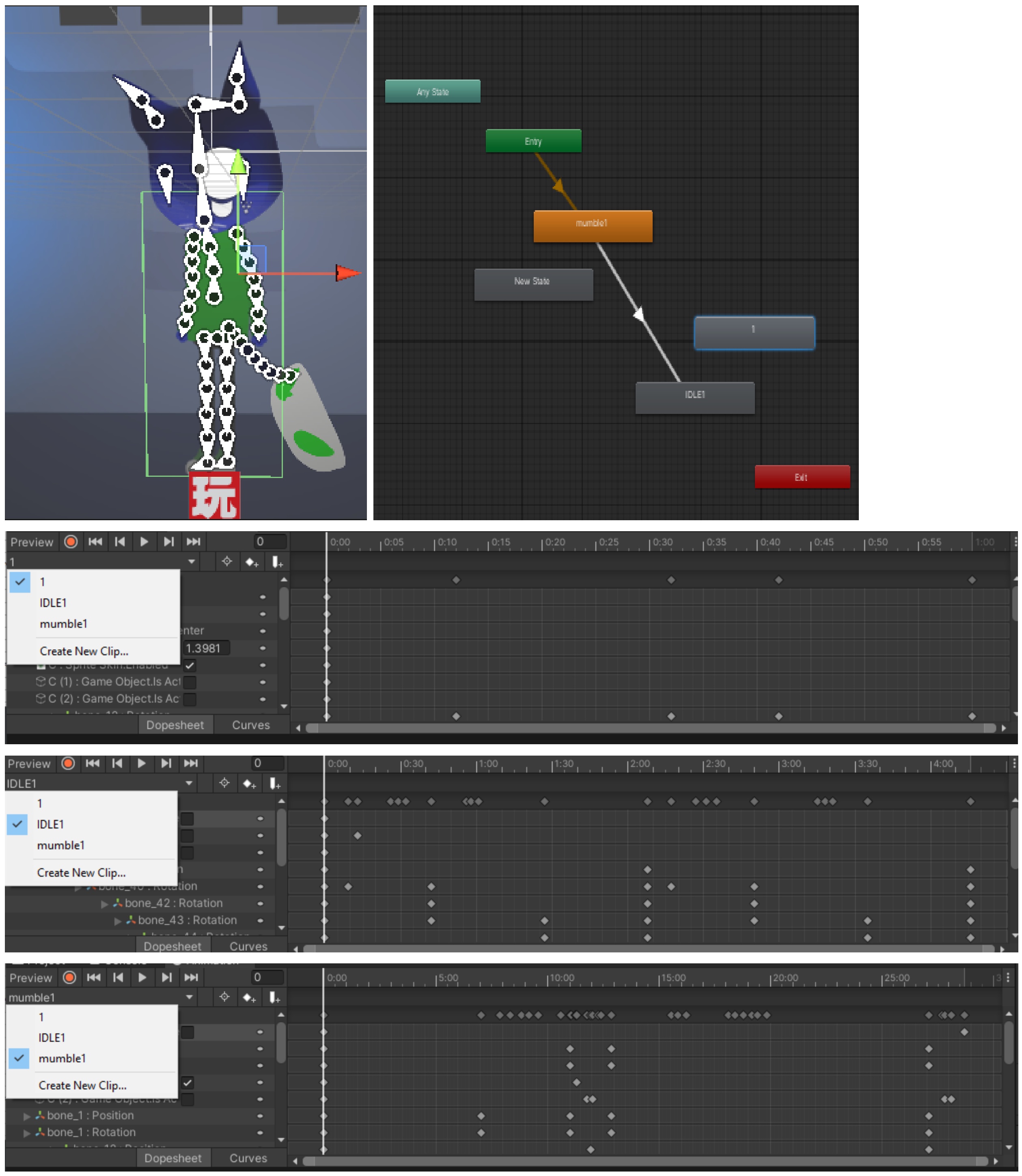


Fig 4.8.2: Main Character Rig And Animations

4.8.3:Creating the Main Game (Summoner Character Building):

The Cinematic scene also has a summoner character from our storyline. The summoner doesn't have many animations because we used bubble chart animation to express more feelings for him. Here is the Rig and Animation of the summoner:

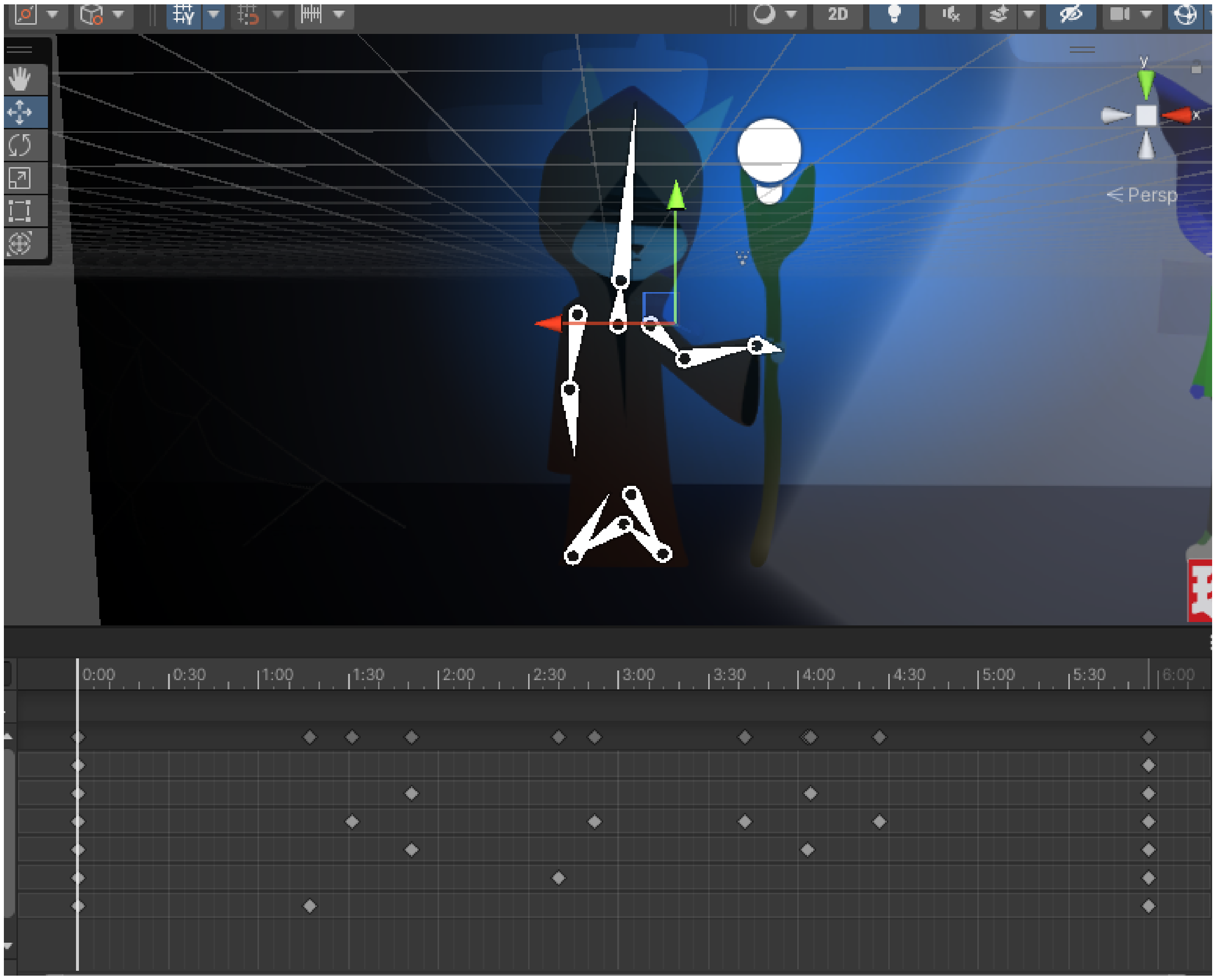


Fig 4.8.3: Summoner Rig And Animations

4.8.4:Creating the Main Game (Environments Lights):
The Cinematic has multiple light passes and the major challenge for me was to create the moonlight coming out of the window. I figured that the freeform light2d will be the best choice to use it:

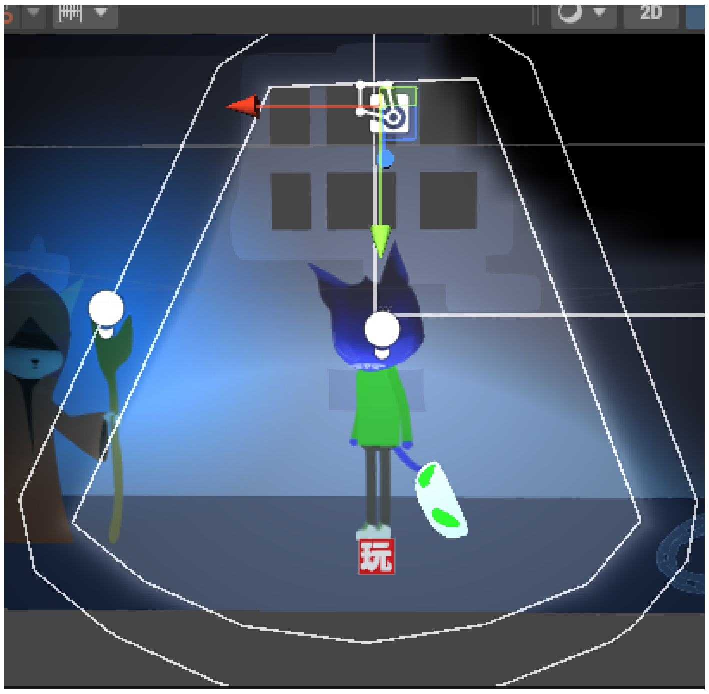


Fig 4.8.3: Environment Lights

There are other lights inside this environment and one of them is the light coming out from the summoners wand which also has animation on tweening its size:

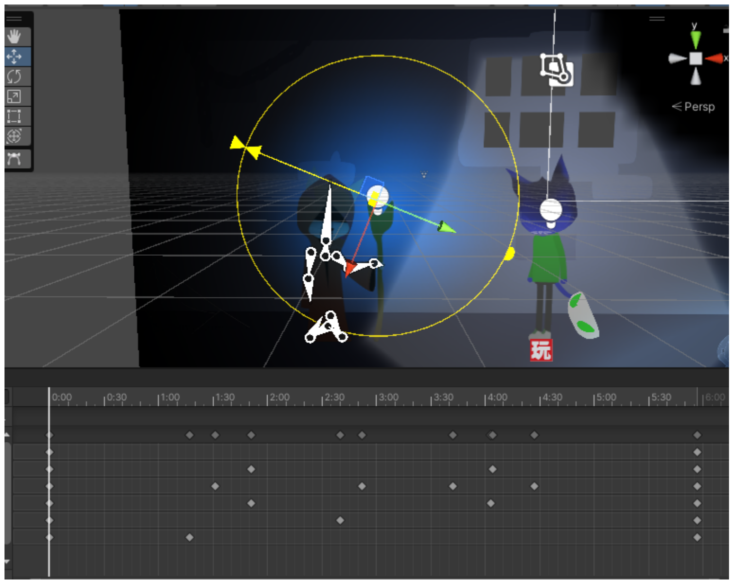


Fig 4.8.4: Summoners wands light

4.8.5:Creating the Main Game (Dialogue Box):
The Cinematic animated dialogue boxes and each of them was animated differently using bone rigging so that they can move in an express full way when the dialogues appear. For example, when the dialogue is about scary facts it wiggles and shakes in a jumpscare way.

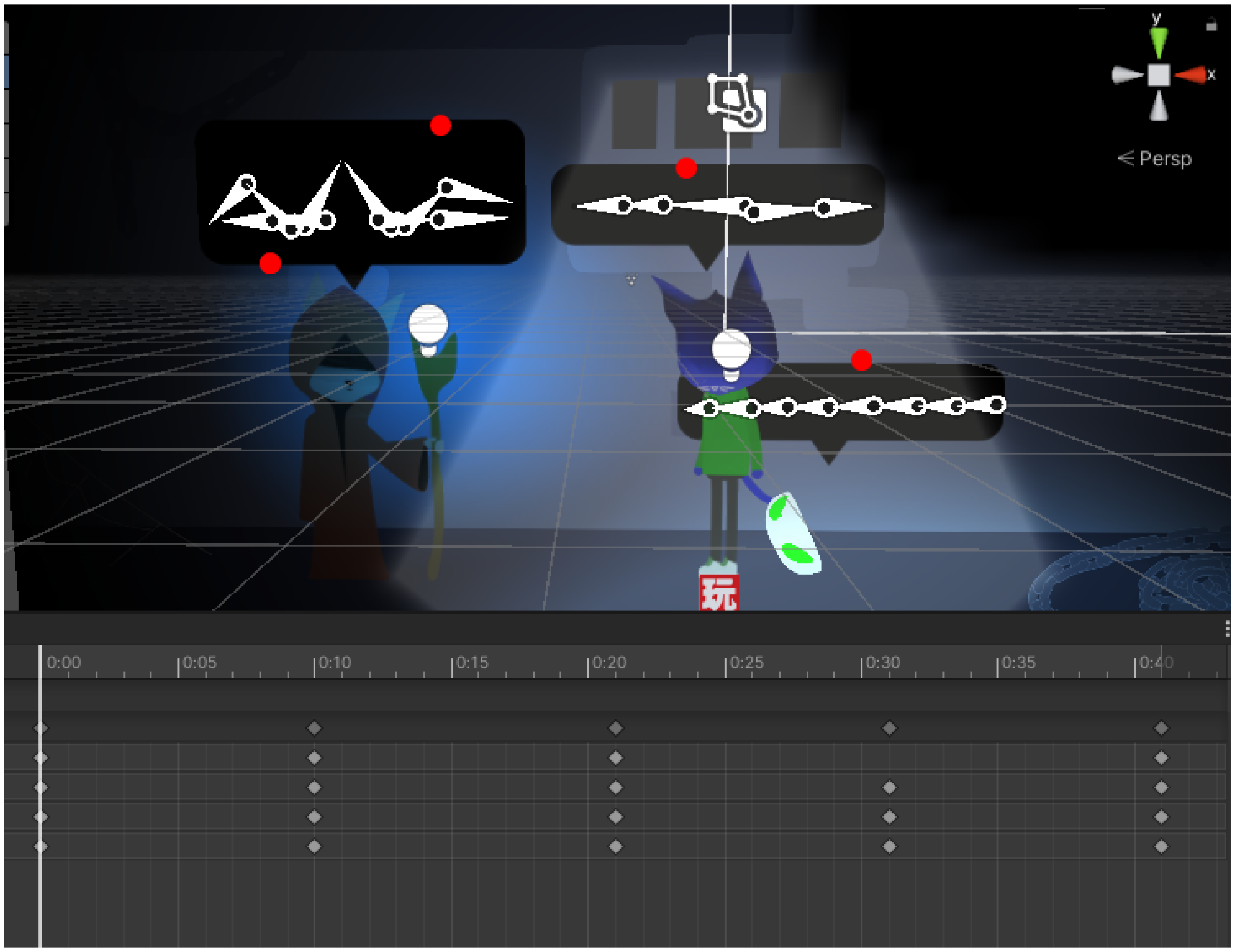
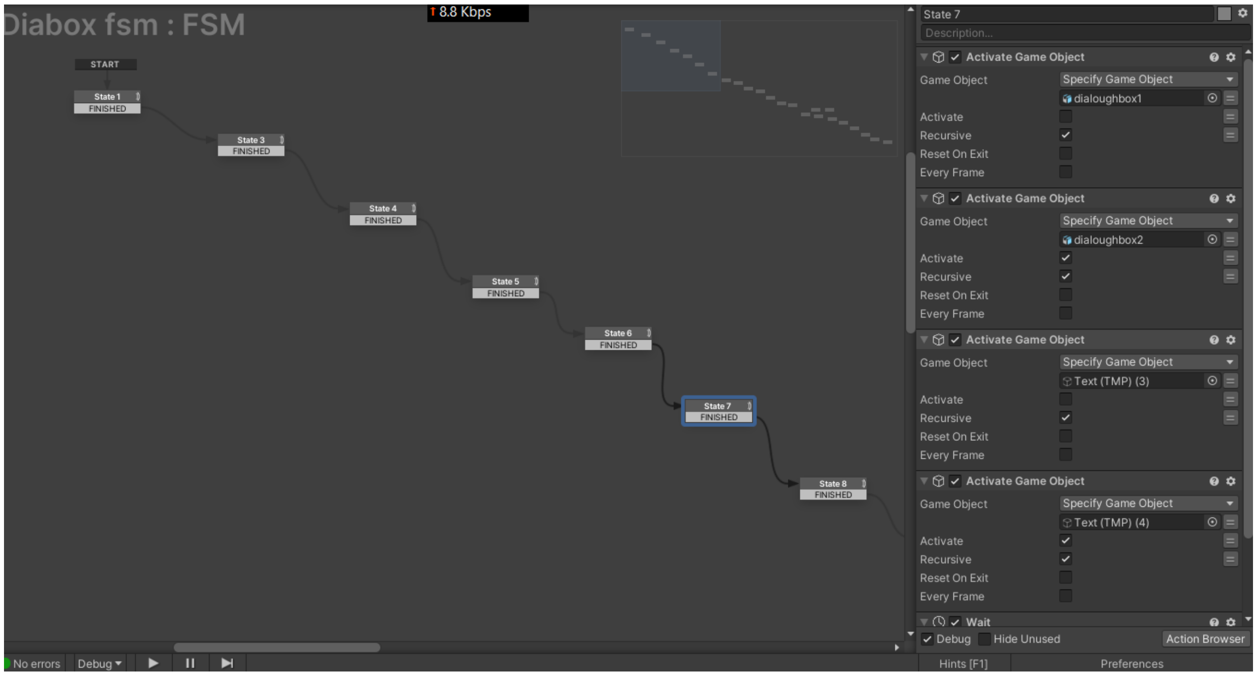


Fig 4.8.5: Dialogue Box rig and animation

4.8.6:Creating the Main Game (Dialogue Box Script):

The Cinematic has a very long visual scripting and it was necessary to put all the stats of the script inside a single FSM to keep everything a flow. I will explain them separately, I will start with the dialogue box process. Here in this image what you can see is the activation and deactivation of the dialogue box based on times and trigger events.



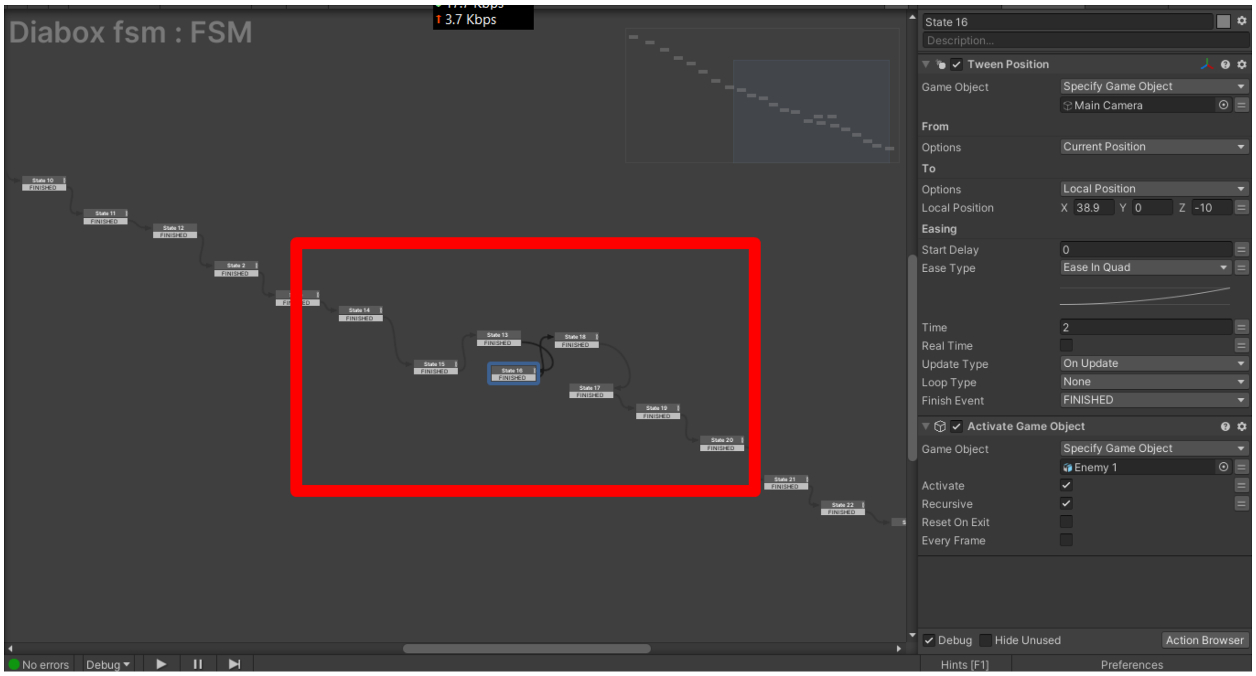
```
State 7
Description...
Activate Game Object
Game Object      Specify Game Object
                 dialoughbox1
Activate
Recursive        ✓
Reset On Exit
Every Frame
Activate Game Object
Game Object      Specify Game Object
                 dialoughbox2
Activate         ✓
Recursive        ✓
Reset On Exit
Every Frame
Activate Game Object
Game Object      Specify Game Object
                 Text (TMP) (3)
Activate
Recursive        ✓
Reset On Exit
Every Frame
Activate Game Object
Game Object      Specify Game Object
                 Text (TMP) (4)
Activate         ✓
Recursive        ✓
Reset On Exit
Every Frame
Wait
✓ Debug   Hide Unused   Action Browser
```

Fig 4.8.6: Dialogue Box Script

Every stat that you can see here has these similar actions inside it and there are more than 20 of those visual script blocks there connected to each other.

4.8.7:Creating the Main Game (Camera Script):

The Cinematic also has camera movements that require scripts to move. Here is the script that has the actions called tween position to move the camera:



State 16
Description...

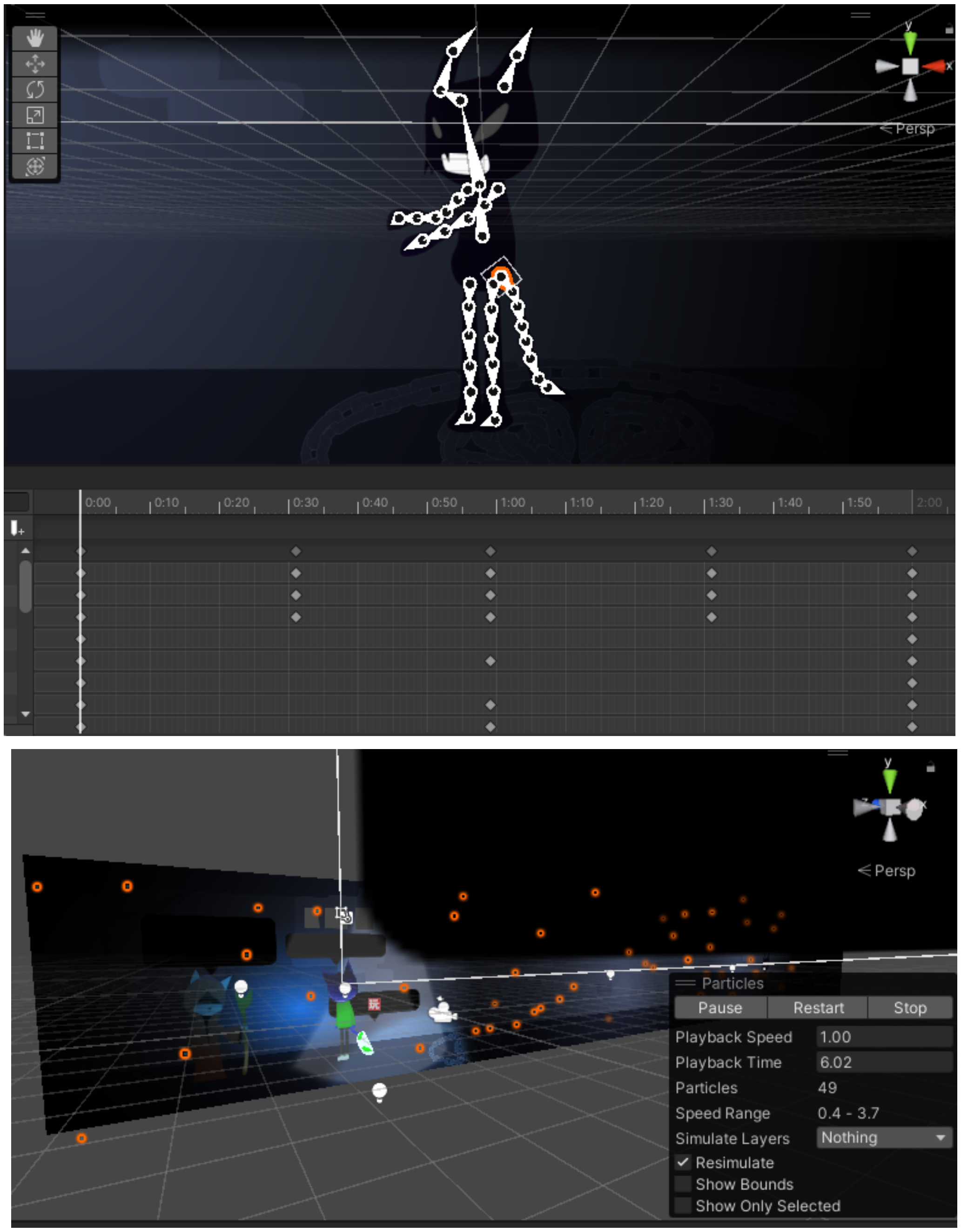
Tween Position
Game Object: Specify Game Object
Main Camera
From
Options: Current Position
To
Options: Local Position
Local Position: X 38.9 Y 0 Z -10
Easing
Start Delay: 0
Ease Type: Ease In Quad
Time: 2
Real Time
Update Type: On Update
Loop Type: None
Finish Event: FINISHED

Activate Game Object
Game Object: Specify Game Object
Enemy 1
Activate: ✓
Recursive: ✓
Reset On Exit
Every Frame

✓ Debug Hide Unused Action Browser
Hints [F1] Preferences

Fig 4.8.7: Camera Movement

4.8.8:Creating the Main Game (Other Staffs):
The Cinematic Scene has many more other works in it, such as rigging and animating the enemy , scripting the transitions of start and end scene, creating the air particles and other many more small works that make the cinematic look more complete. Here is some of the image of those works:



Playback Speed 1.00
Playback Time 6.02
Particles 49
Speed Range 0.4 - 3.7
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

Fig 4.8.8: other works in cinematic.

4.9:Creating the Main Game (Main game scene setup):
After the cinematic, the scene will have a transition and will load the main game scene. I made the game scene in a way so that we can create a new level, put it inside the scene and it will run properly more like a plugin but also a part of the game. We have added a level as an approval of it:

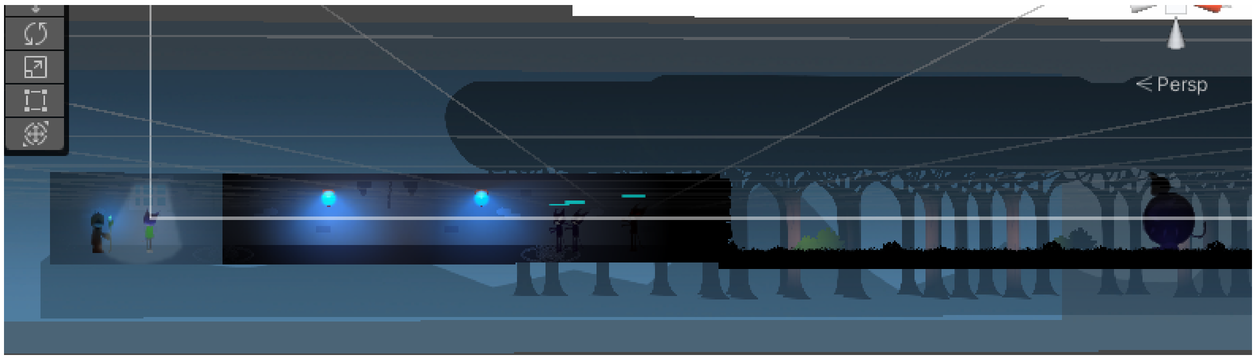


Fig 4.9: Levels.

The main game is a level base game where the player will have to kill an enemy to level up and unlock new abilities and keep forwarding to new enemies and levels.

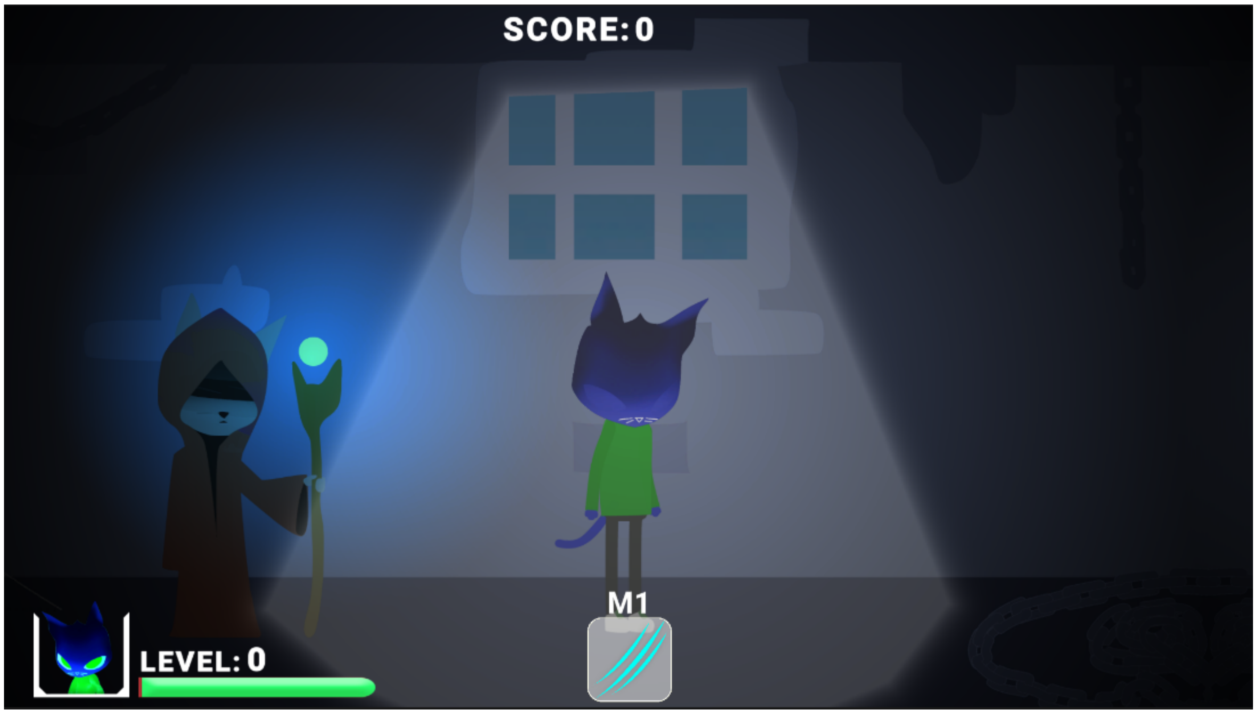


Fig 4.9: Scene

4.9:Creating the Main Game (Layers):

The main game has more than 20 layers of artwork inside it and all of them are well placed to pull out the prospectiveness of the world.

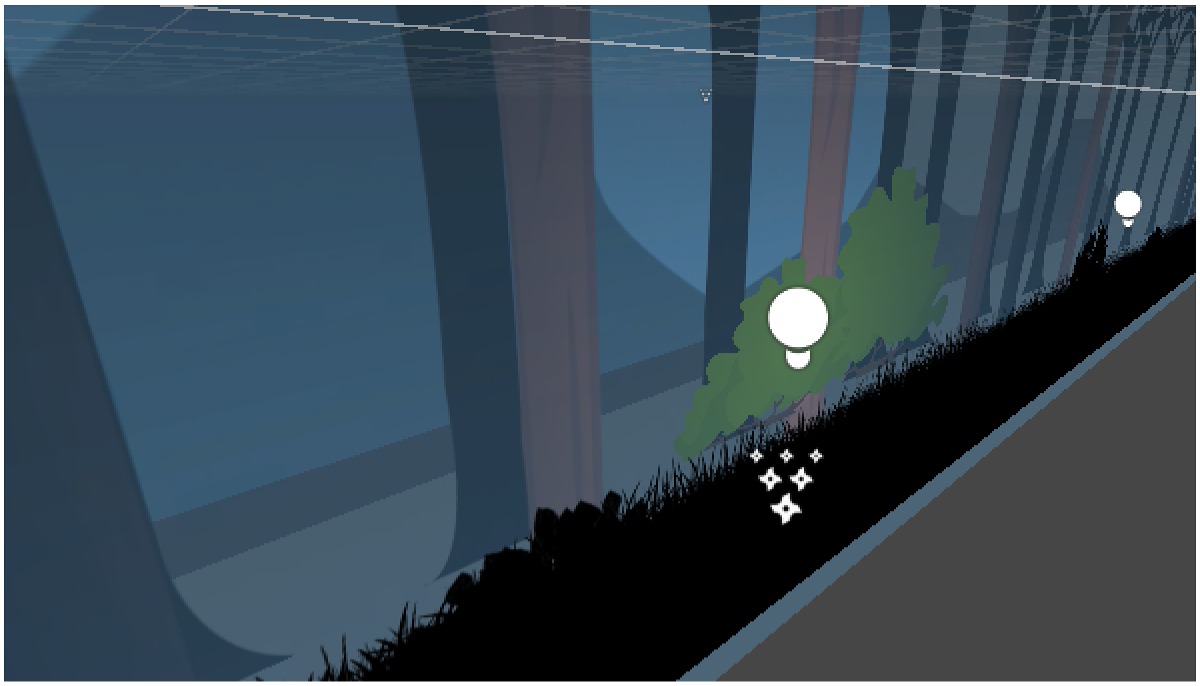
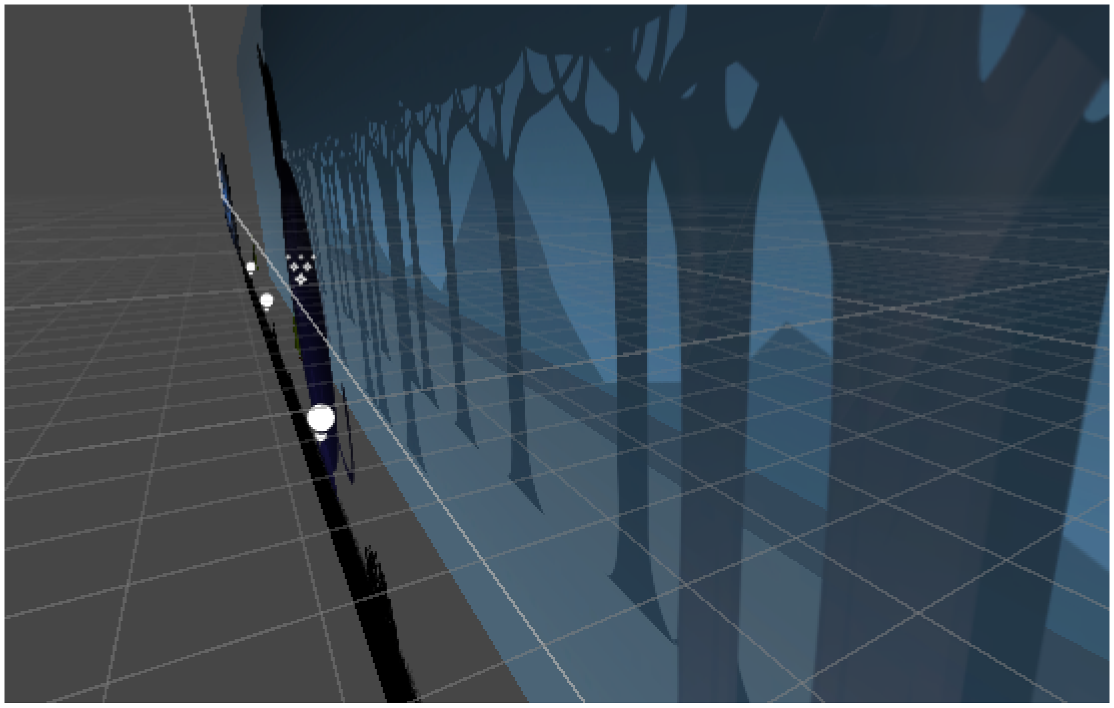



Fig 4.9.1: Layers.

4.9.2:Creating the Main Game (Main Character):
The main character in this level has more options inside it than the previous ones such as in the properties there are RigidBody2D and Capsule Collider and box Collider to trigger and support the character more compatibly.

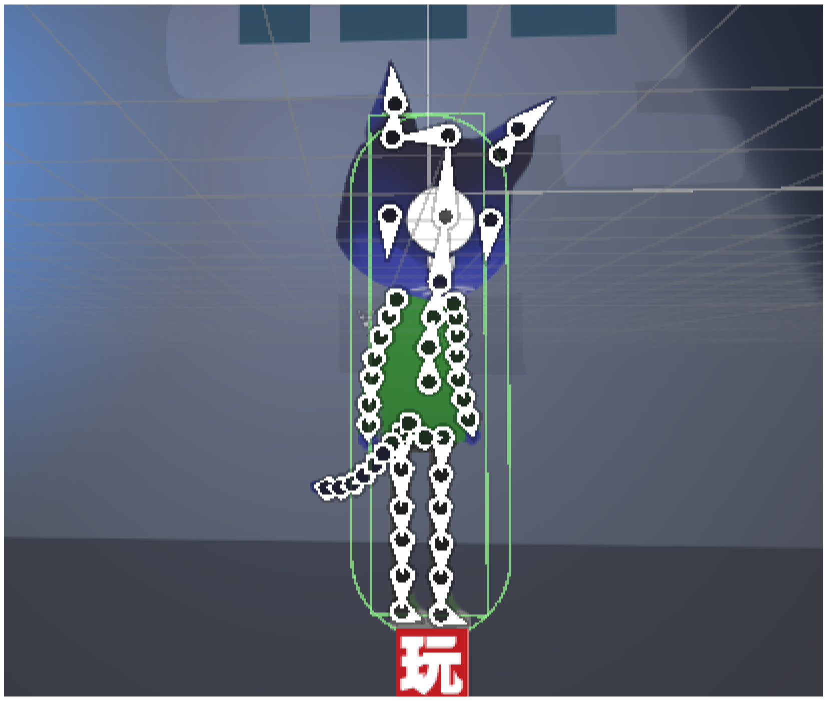
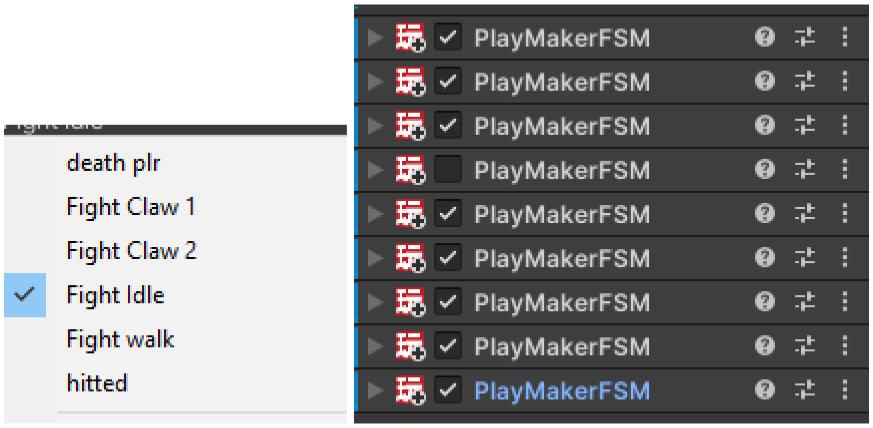


Fig 4.9.2: Player.

There are now many more animations and fsm insite it which i will explain now:


death plr
Fight Claw 1
Fight Claw 2
✓ Fight Idle
Fight walk
hitted

✓ PlayMakerFSM
✓ PlayMakerFSM
✓ PlayMakerFSM
PlayMakerFSM
✓ PlayMakerFSM
✓ PlayMakerFSM
✓ PlayMakerFSM
✓ PlayMakerFSM
✓ PlayMakerFSM

Fig 4.9.2: Animation and Fsm

4.9.3:Creating the Main Game (Main Character Animation):

The main Character needed about 6 animations to perform. Here are all the animations I have made by the animator of unity:

This one is the death animation when a player will die this animation will be played:

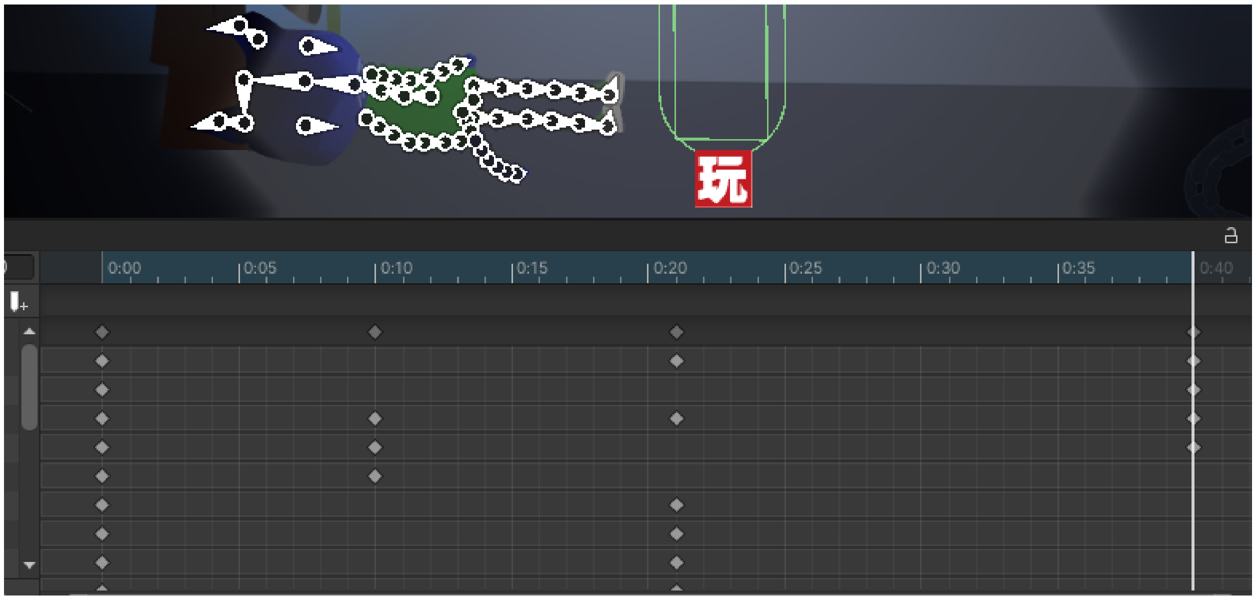


Fig 4.9.3: Death Animation

This is the fighting animation attack 1 claw attack:

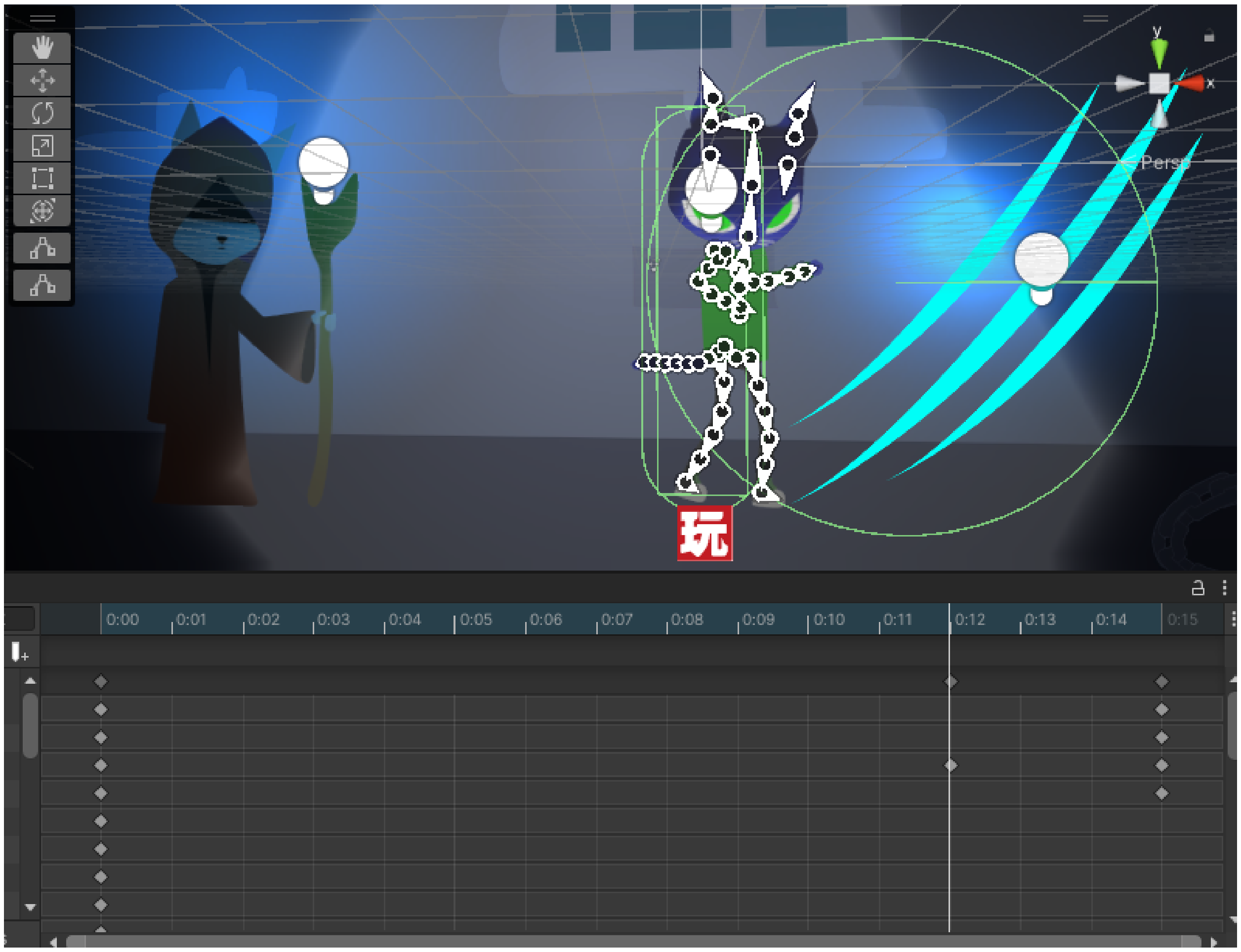


Fig 4.9.3: Attack Animation

The claw attack image is also a part of the animation attached to the player and will activate when the player will click the attack button.

This is the Shield animation power2:

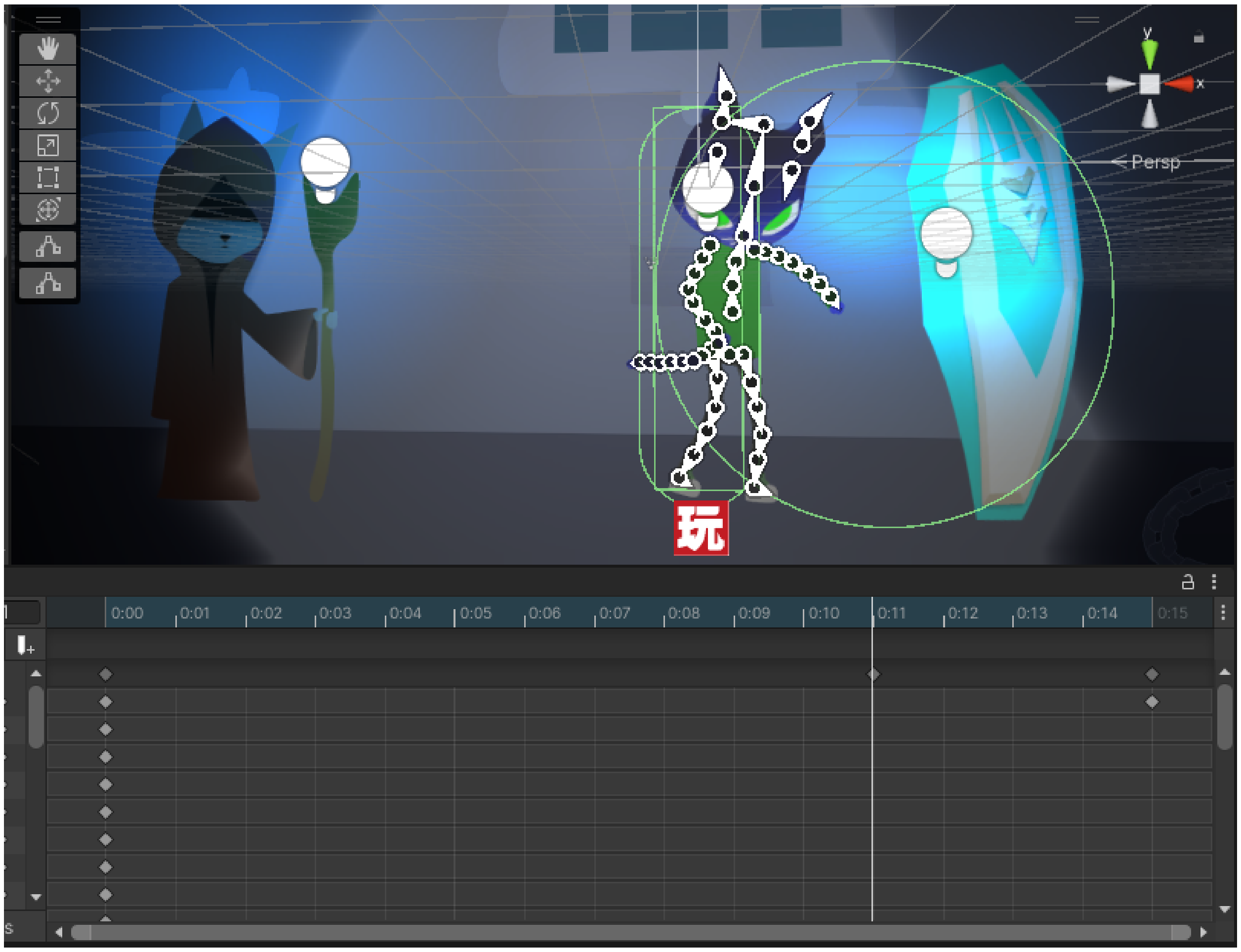


Fig 4.9.3: Shield Animation

The Shield image is also a part of the animation attached to the player and will activate when the player will click the Shield button.

Idle Animation:

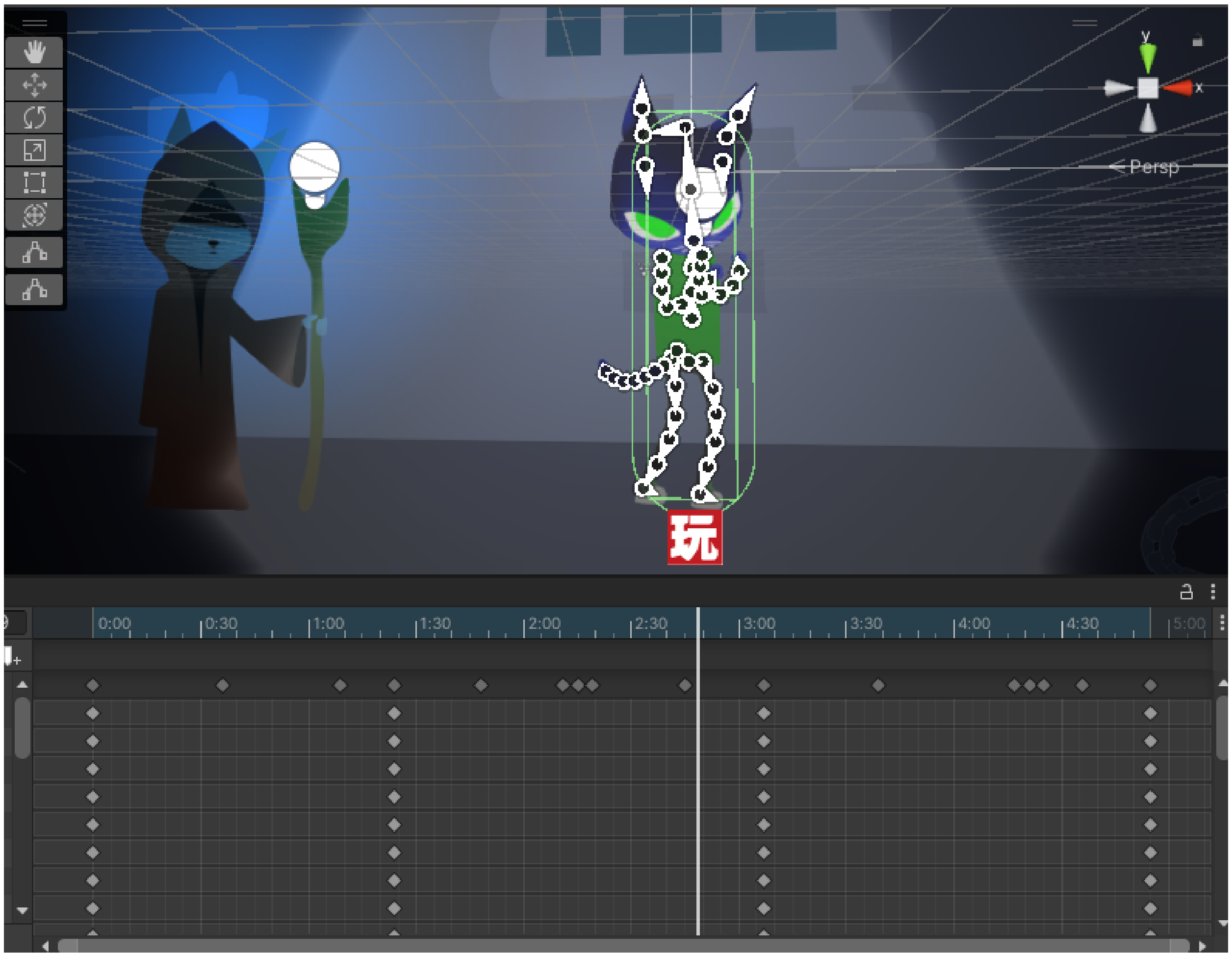


Fig 4.9.3: Idle Animation

It plays when the player is in idle mode and doing nothing.

Walk animation:

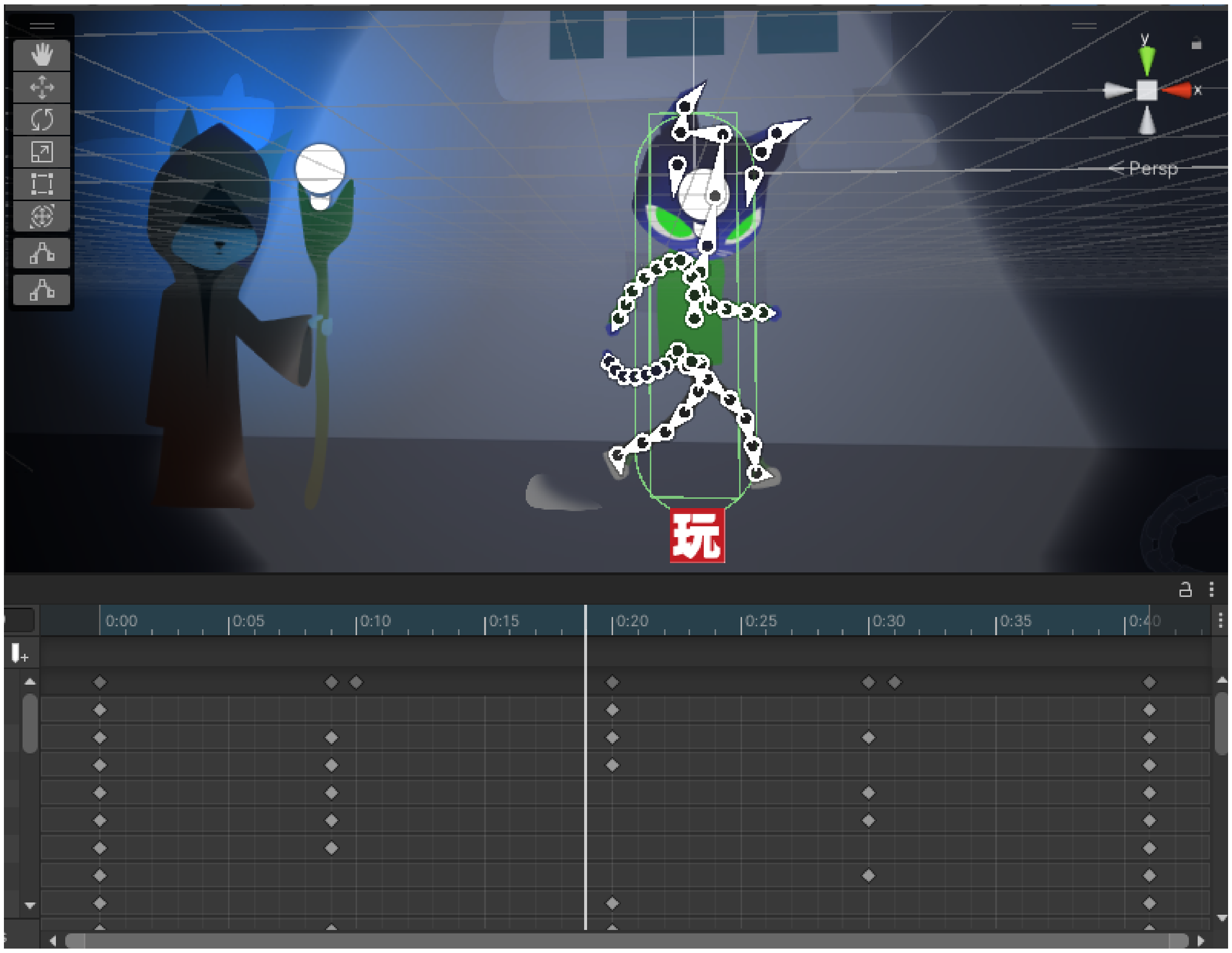


Fig 4.9.3: Walk Animation

Plays when the player is walking.

Hitted animation:

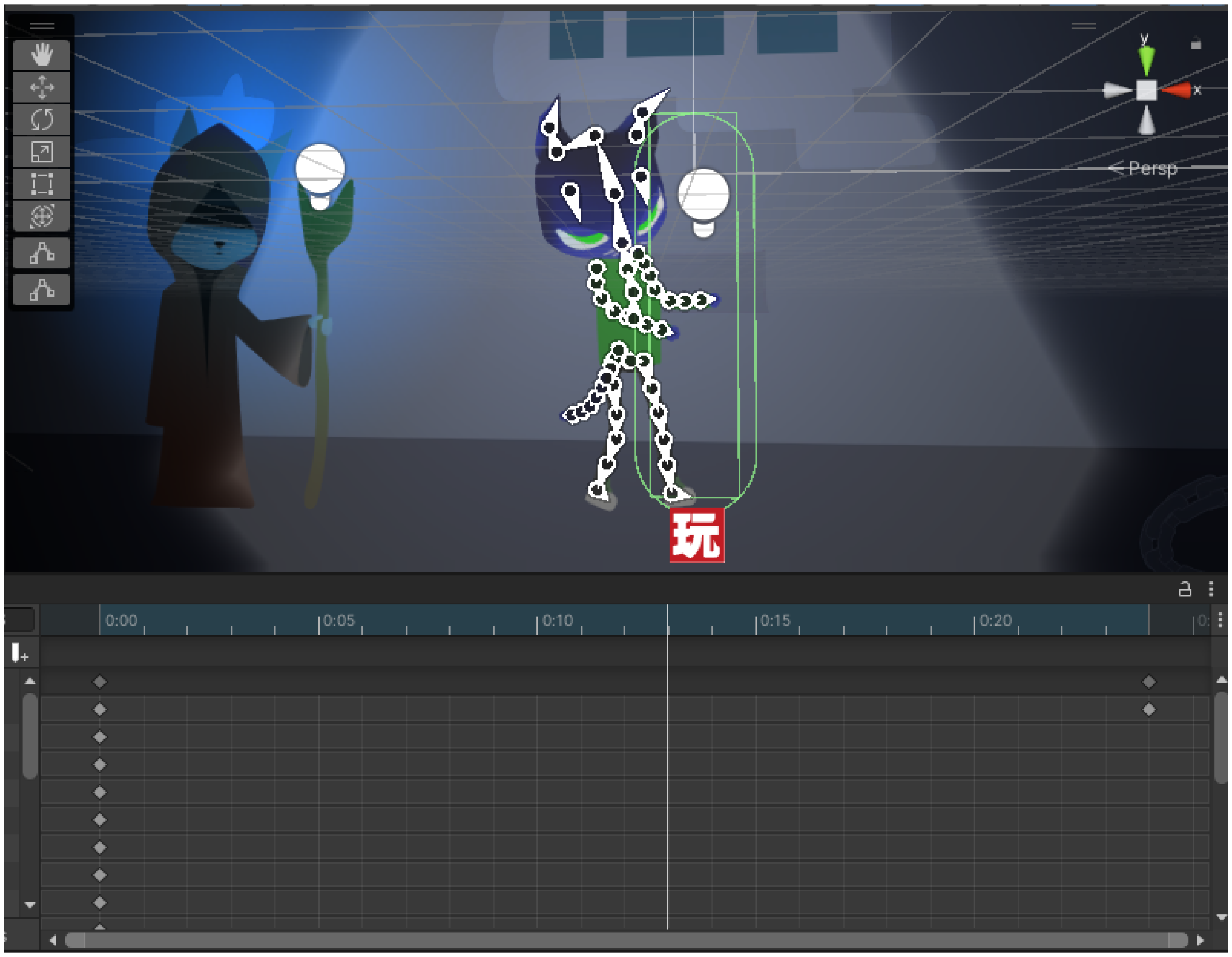
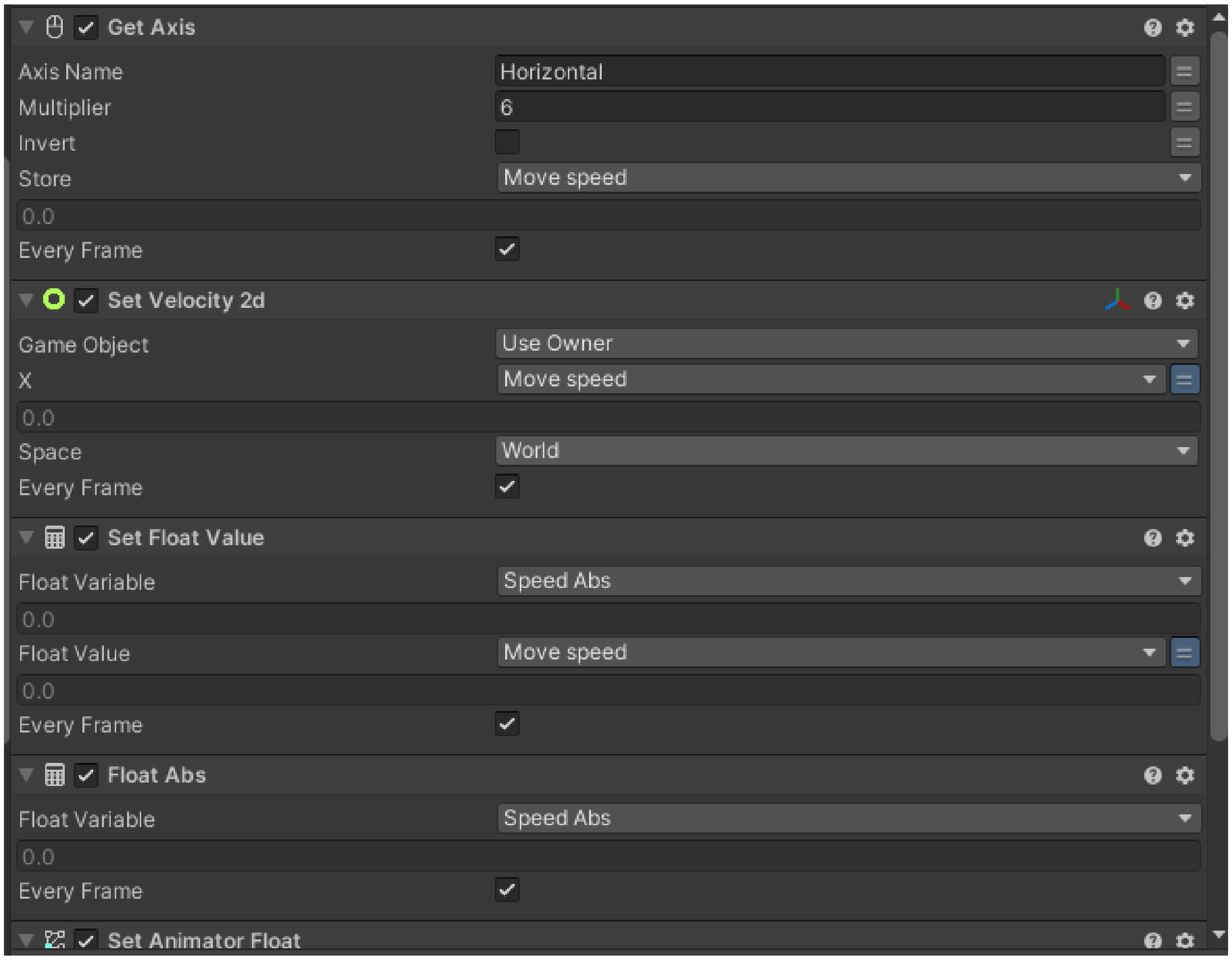


Fig 4.9.3: Hitted Animation

Play when the player gets hitted by the enemy.

4.10.1:Creating the Main Game (Main Character Fsm/Scripts):
The main character has nine local visual scripts inside it. The first one is the walking script, where I collected the axis of the player's movement and then I set it to its velocity to give it force and to the number of forces and converted it to float to trigger the movement animation.



Get Axis

Axis Name	Horizontal
Multiplier	6
Invert	
Store	Move speed
	0.0
Every Frame	✓

Set Velocity 2d

Game Object	Use Owner
X	Move speed
	0.0
Space	World
Every Frame	✓

Set Float Value

Float Variable	Speed Abs
	0.0
Float Value	Move speed
	0.0
Every Frame	✓

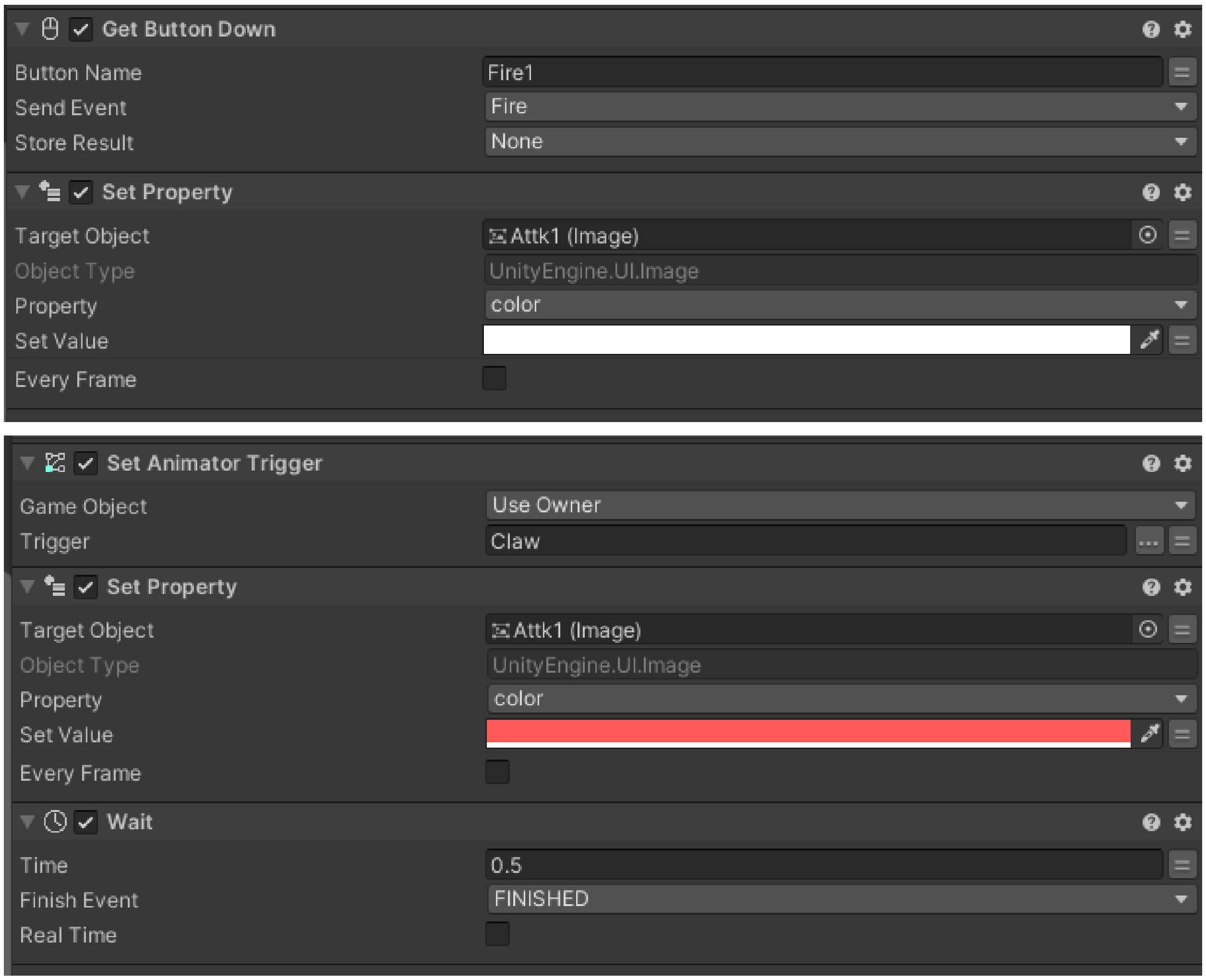
Float Abs

Float Variable	Speed Abs
	0.0
Every Frame	✓

Set Animator Float

Fig 4.10.1: Movement Fsm

The next script is the Attack 1 which get triggered by the mouse 1 button and on button down it plays the attack 1 animation and activates the claw image.



Get Button Down

Button Name	Fire1
Send Event	Fire
Store Result	None

Set Property

Target Object	Attk1 (Image)
Object Type	UnityEngine.UI.Image
Property	color
Set Value	⬜
Every Frame	☐

Set Animator Trigger

Game Object	Use Owner
Trigger	Claw

Set Property

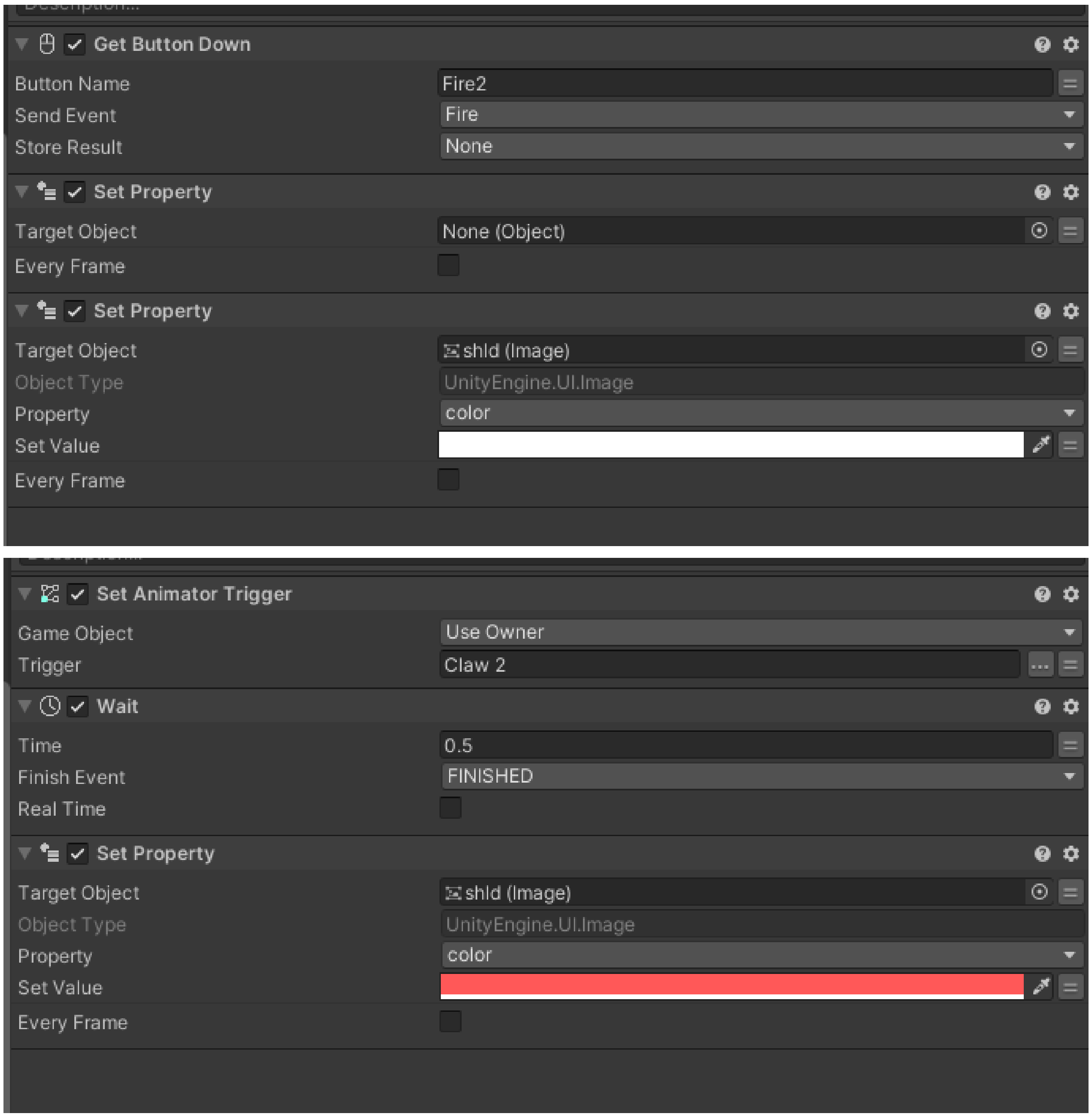
Target Object	Attk1 (Image)
Object Type	UnityEngine.UI.Image
Property	color
Set Value	🟥
Every Frame	☐

Wait

Time	0.5
Finish Event	FINISHED
Real Time	☐

Fig 4.10.1: Attack1 Fsm

The next script is the Shield which gets triggered by the mouse 2 button and on button down it plays the shield animation and activates the shield image.



Get Button Down

Button Name	Fire2
Send Event	Fire
Store Result	None

Set Property

Target Object	None (Object)
Every Frame	☐

Set Property

Target Object	shld (Image)
Object Type	UnityEngine.UI.Image
Property	color
Set Value	⬜
Every Frame	☐

Set Animator Trigger

Game Object	Use Owner
Trigger	Claw 2

Wait

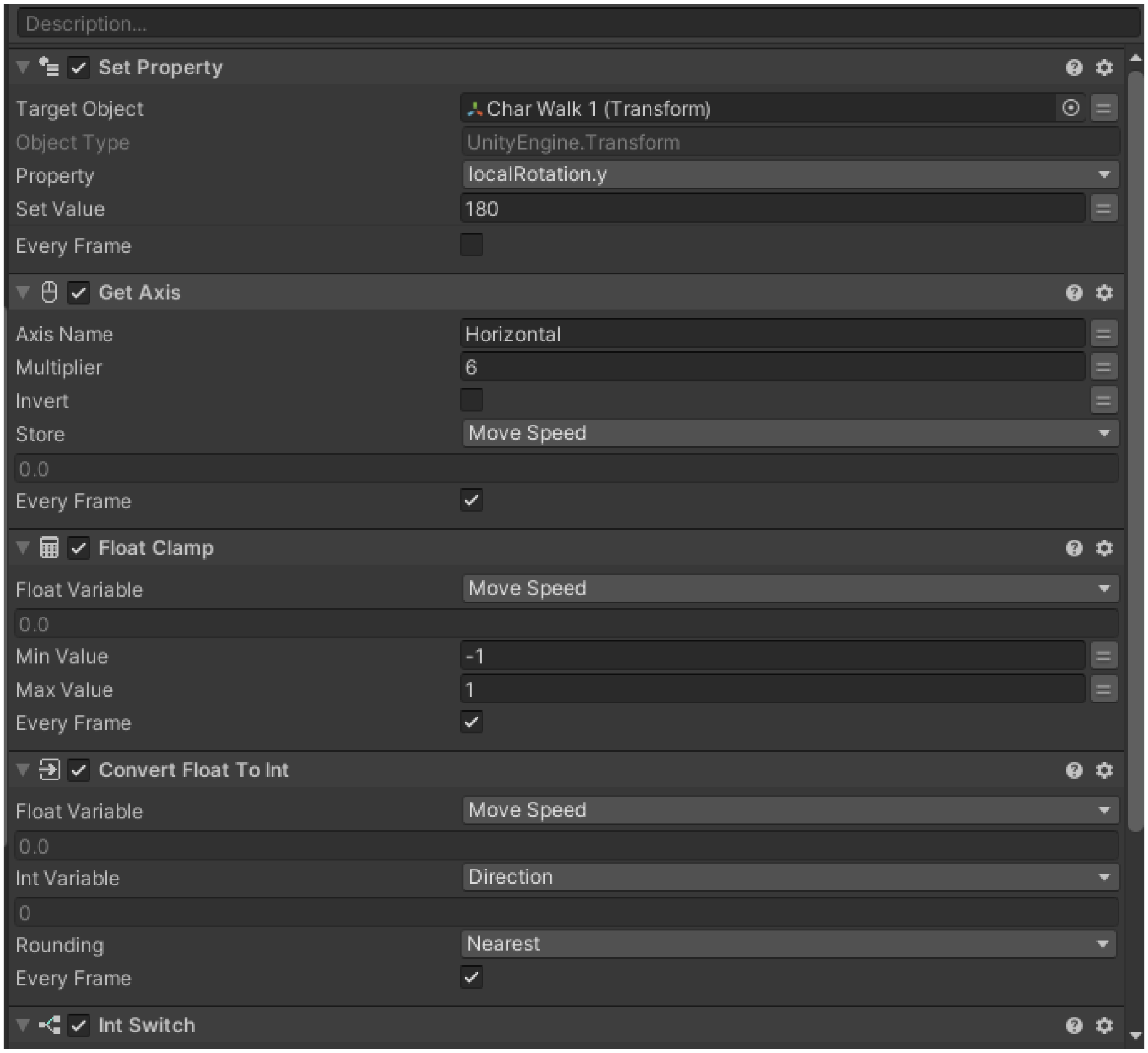
Time	0.5
Finish Event	FINISHED
Real Time	☐

Set Property

Target Object	shld (Image)
Object Type	UnityEngine.UI.Image
Property	color
Set Value	🟥
Every Frame	☐

Fig 4.10.1: Shield Fsm

Next is the flip animation, It flips the character according to the direction it is moving toward.



Description...

Set Property

Target Object	Char Walk 1 (Transform)
Object Type	UnityEngine.Transform
Property	localRotation.y
Set Value	180
Every Frame	☐

Get Axis

Axis Name	Horizontal
Multiplier	6
Invert	☐
Store	Move Speed
0.0	
Every Frame	✓

Float Clamp

Float Variable	Move Speed
0.0	
Min Value	-1
Max Value	1
Every Frame	✓

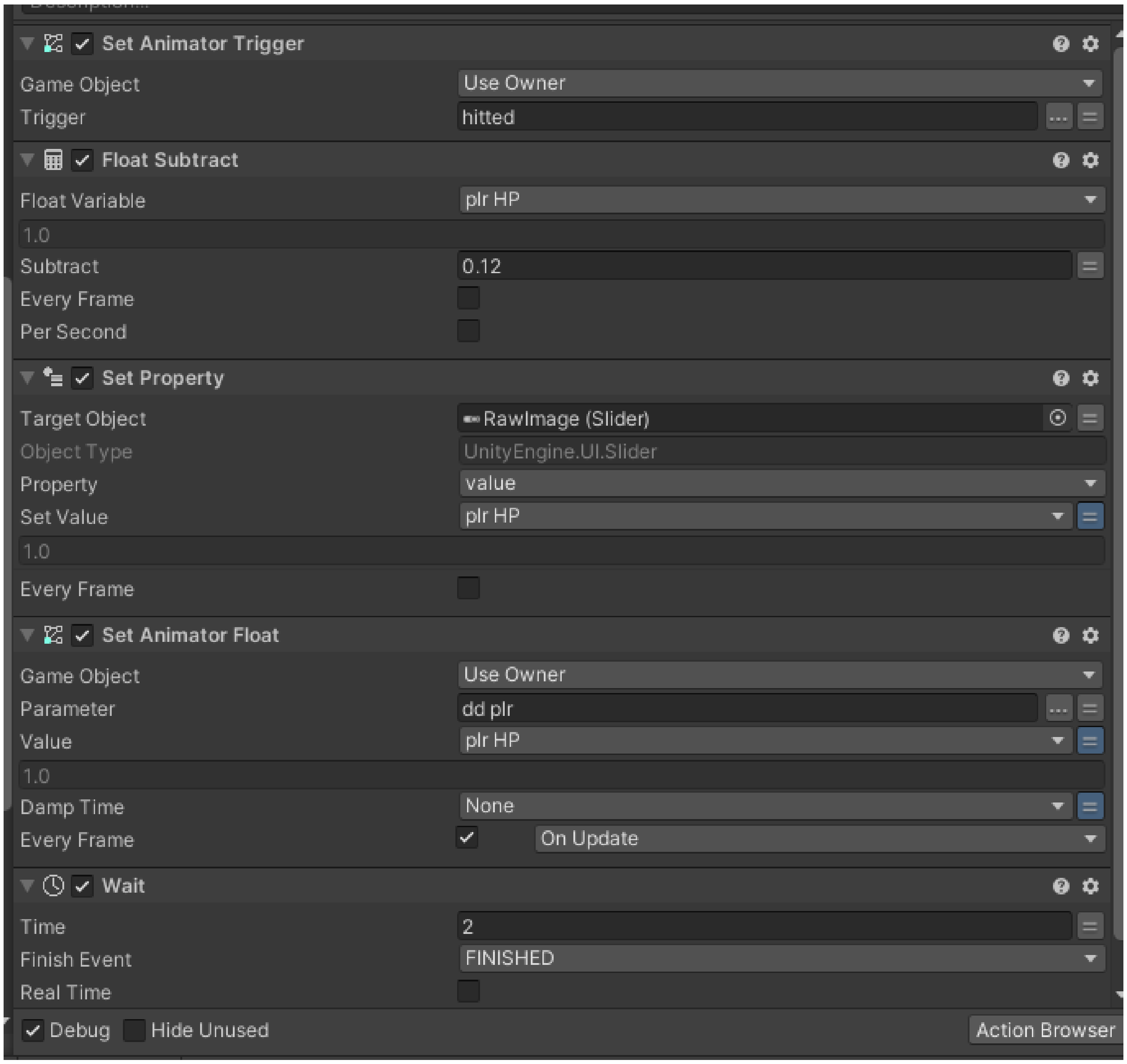
Convert Float To Int

Float Variable	Move Speed
0.0	
Int Variable	Direction
0	
Rounding	Nearest
Every Frame	✓

Int Switch

Fig 4.10.1: Flip Fsm

Next is the hit detection script when a player gets hit by an enemy attack this script gets triggered and plays the hit animation which also reduces the health of the player.



Set Animator Trigger ✓

Game Object	Use Owner
Trigger	hitted

Float Subtract ✓

Float Variable	plr HP
	1.0
Subtract	0.12
Every Frame	☐
Per Second	☐

Set Property ✓

Target Object	RawImage (Slider)
Object Type	UnityEngine.UI.Slider
Property	value
Set Value	plr HP
	1.0
Every Frame	☐

Set Animator Float ✓

Game Object	Use Owner
Parameter	dd plr
Value	plr HP
	1.0
Damp Time	None
Every Frame	✓ On Update

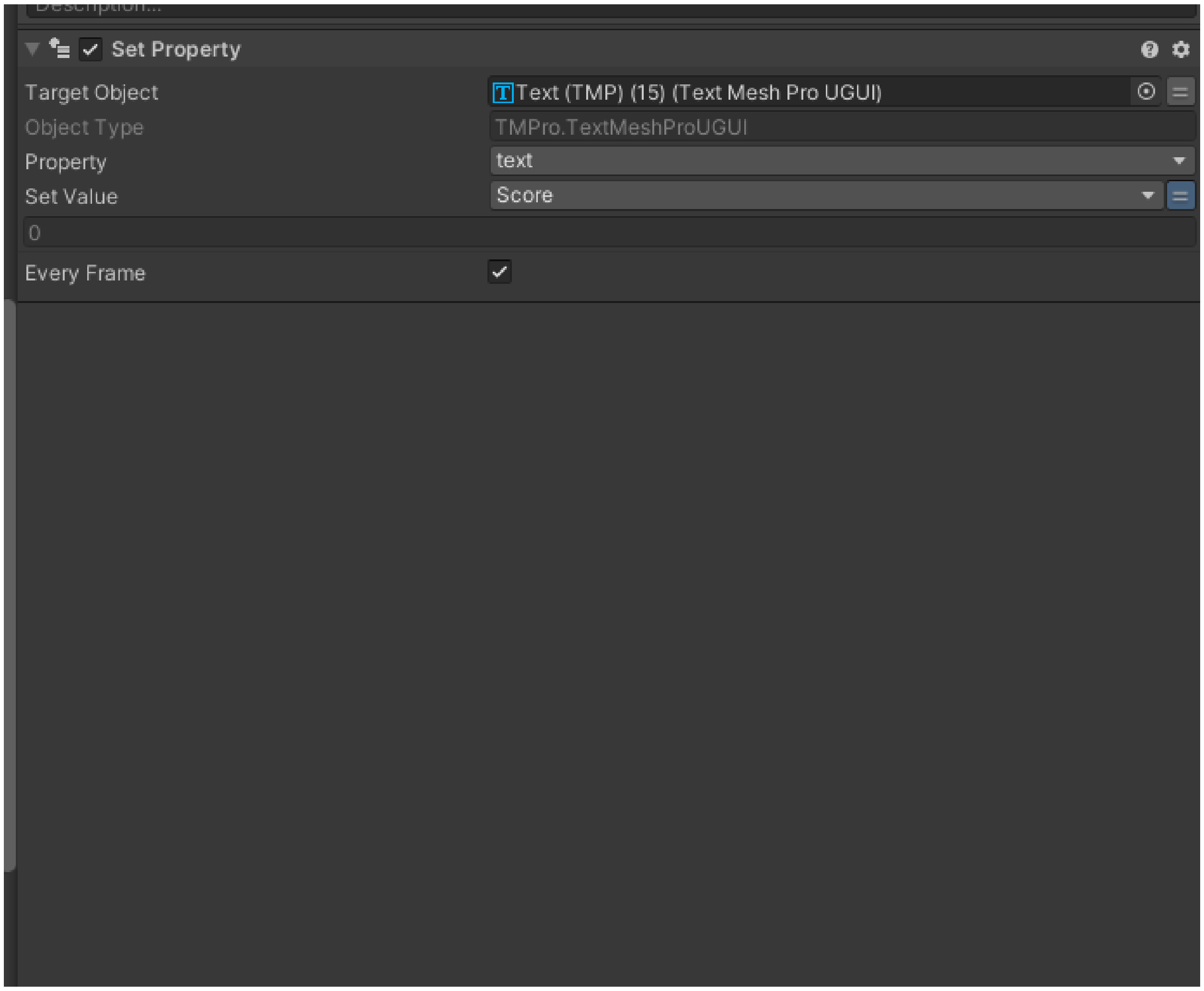
Wait ✓

Time	2
Finish Event	FINISHED
Real Time	☐

✓ Debug ☐ Hide Unused Action Browser

Fig 4.10.1: Hit Fsm

Next is score fsm which updates the score text on the ui when a player kills an enemy.



Description...

Set Property

Target Object: Text (TMP) (15) (Text Mesh Pro UGUI)

Object Type: TMPro.TextMeshProUGUI

Property: text

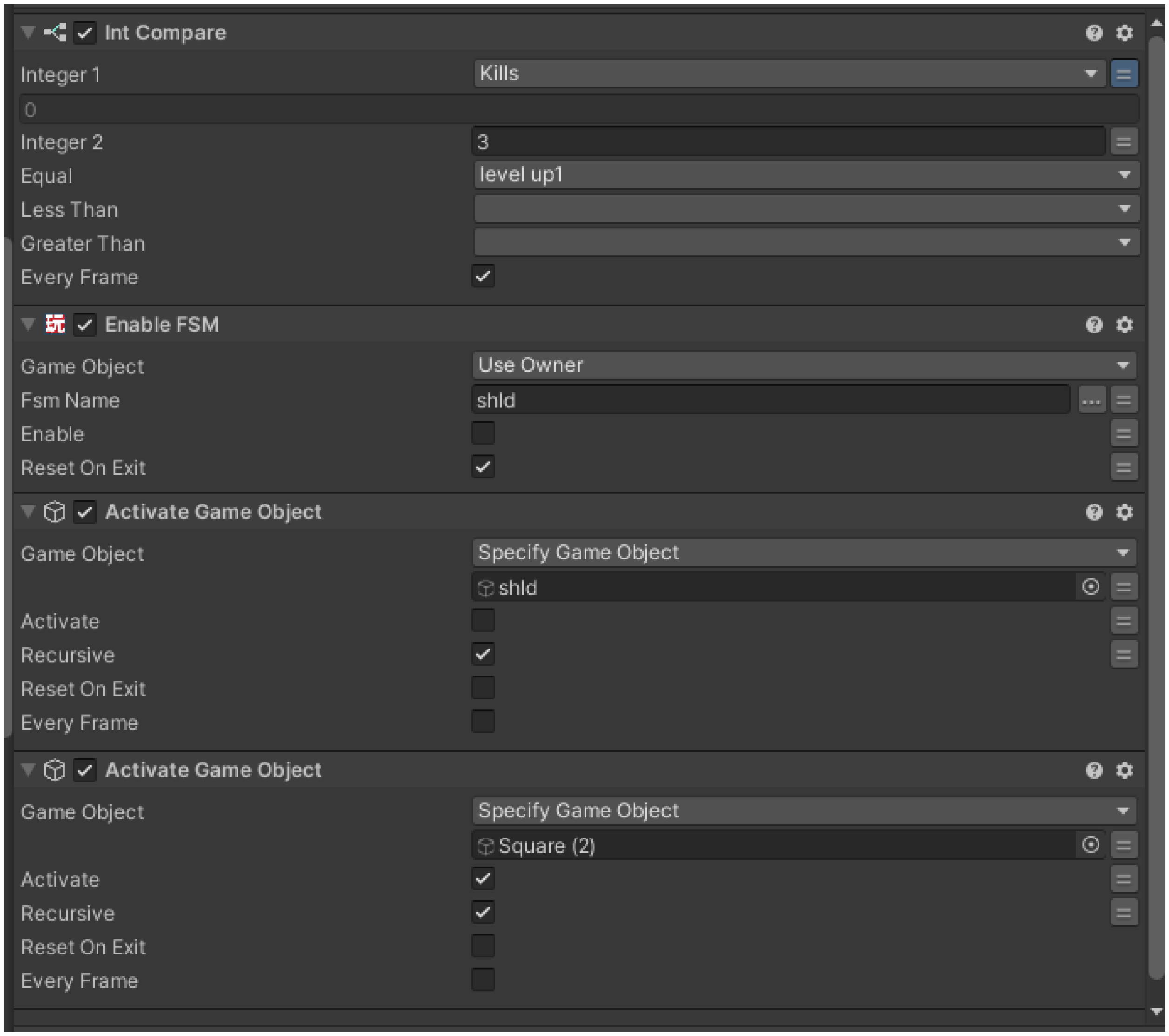
Set Value: Score

0

Every Frame: ✓

Fig 4.10.1: Score Fsm

Next is the level fsm which checks and updates the level number when a player completes the requirement of leveling up and also unlocks new abilities.



Int Compare

Integer 1	Kills
	0
Integer 2	3
Equal	level up1
Less Than	
Greater Than	
Every Frame	✓

Enable FSM

Game Object	Use Owner
Fsm Name	shld
Enable	
Reset On Exit	✓

Activate Game Object

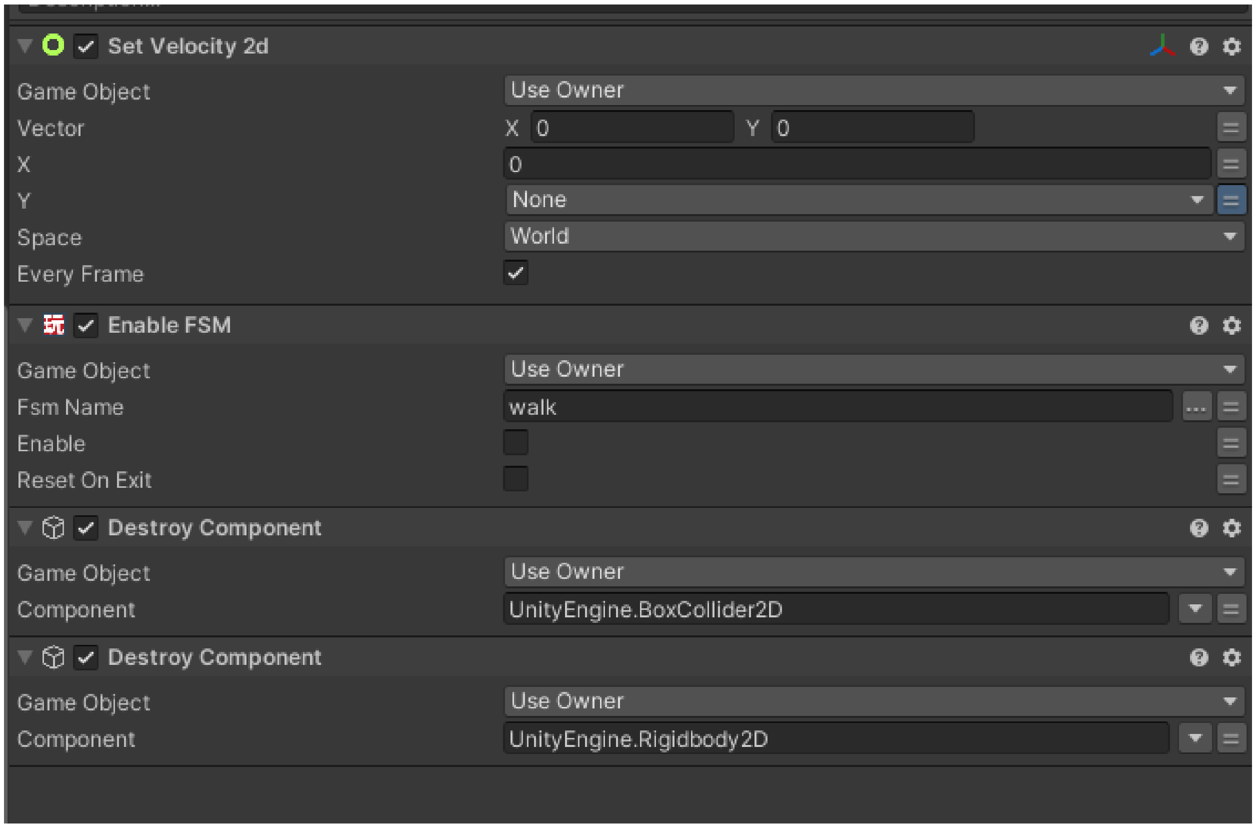
Game Object	Specify Game Object
	shld
Activate	
Recursive	✓
Reset On Exit	
Every Frame	

Activate Game Object

Game Object	Specify Game Object
	Square (2)
Activate	✓
Recursive	✓
Reset On Exit	
Every Frame	

Fig 4.10.1: Level Fsm

Another important fsm inside the player is the death fsm that checks if the player is dead/Hp value is 0 or <1 then it stops all player activity and restarts the stage.



Set Velocity 2d

Game Object	Use Owner
Vector	X 0 Y 0
X	0
Y	None
Space	World
Every Frame	✓

Enable FSM

Game Object	Use Owner
Fsm Name	walk
Enable	
Reset On Exit	

Destroy Component

Game Object	Use Owner
Component	UnityEngine.BoxCollider2D

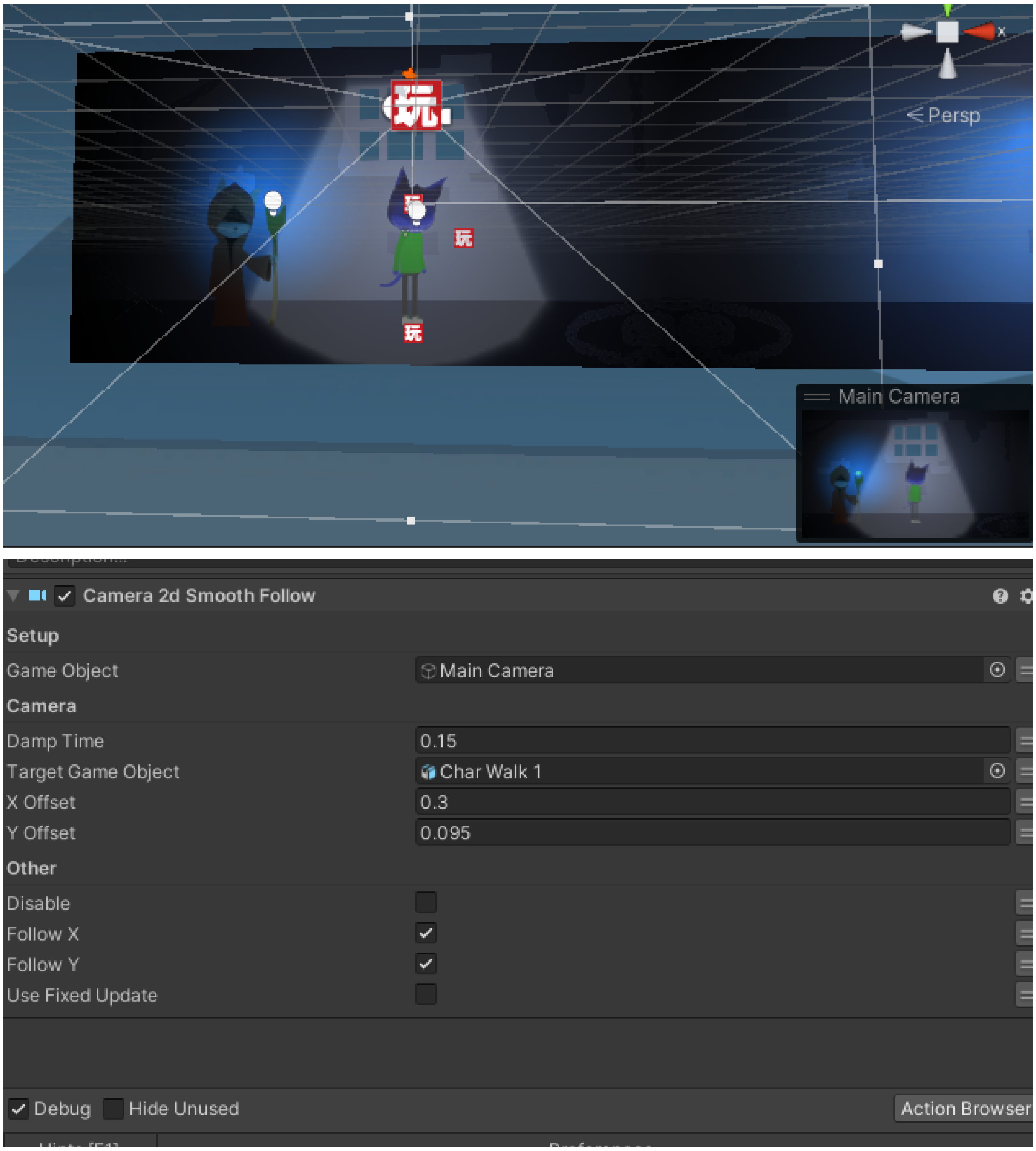
Destroy Component

Game Object	Use Owner
Component	UnityEngine.Rigidbody2D

Fig 4.10.1: Death Fsm

4.10.2:Creating the Main Game (Camera Fsm/Scripts):

The main camera has a script inside it that helps the camera to smoothly follow the player from any direction and placing the camera is a position where the player can have more fields of view to the direction the player is looking at.



Setup

Game Object	Main Camera

Camera

Damp Time	0.15
Target Game Object	Char Walk 1
X Offset	0.3
Y Offset	0.095

Other

Disable	☐
Follow X	✓
Follow Y	✓
Use Fixed Update	☐

Fig 4.10.1: Camera Fsm

4.10.2.1:Creating the Main Game (Enemy Workflow):

The enemy has several workarounds and some of them are complicated. Enemy has the ability to follow the player and attack it and damage dealing.
The enemy has its own rigs and animation as following:

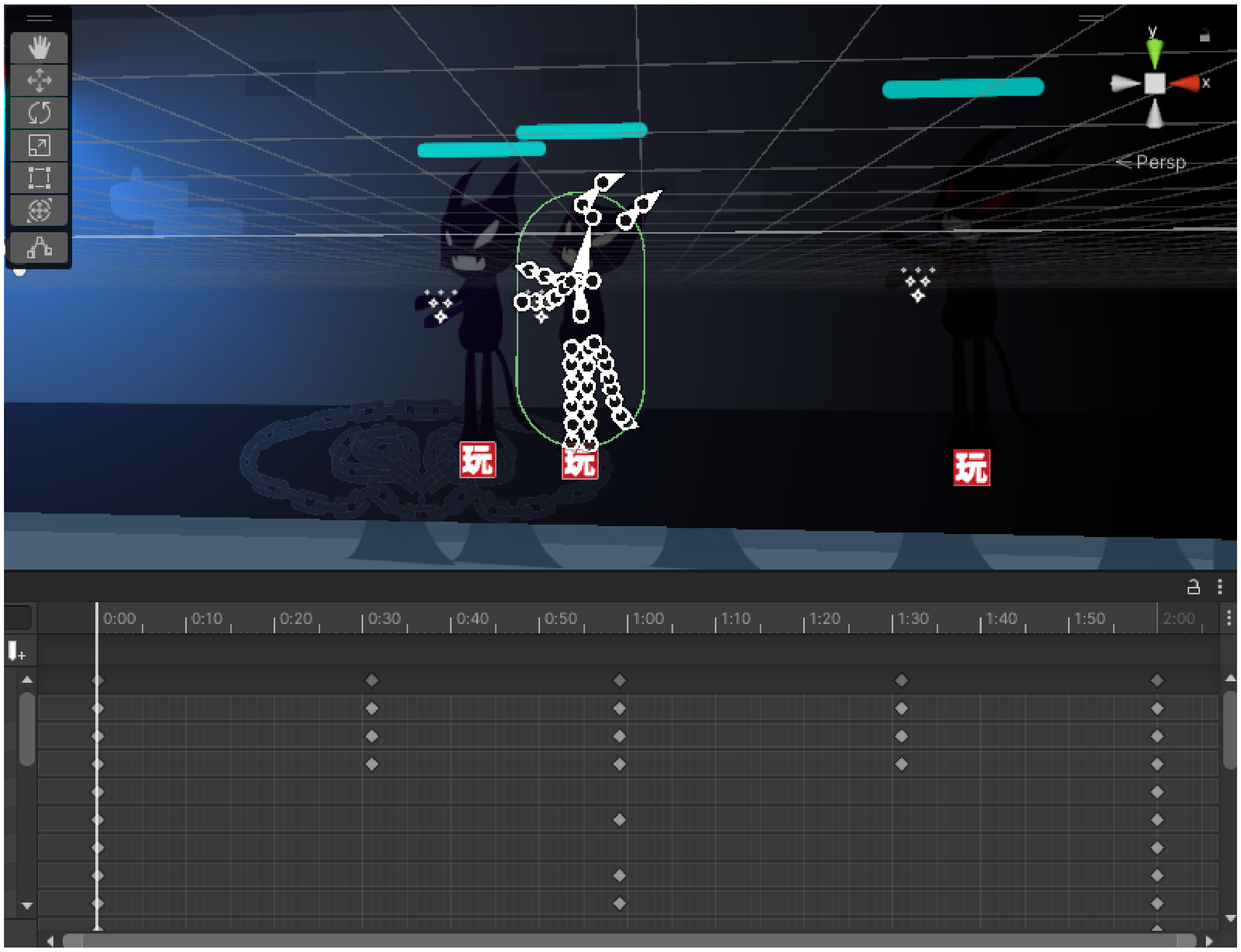
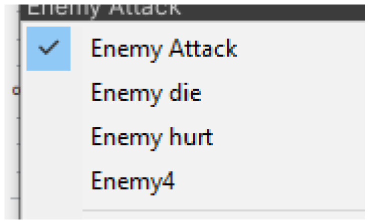


Fig 4.10.2: Enemy Rig and Animation

It also has colliders and tags to detect the nearby players. And has 4 animations inside it as following:



Enemy Attack

✓ Enemy Attack
Enemy die
Enemy hurt
Enemy4

The first animation is the enemy follow the player animation:

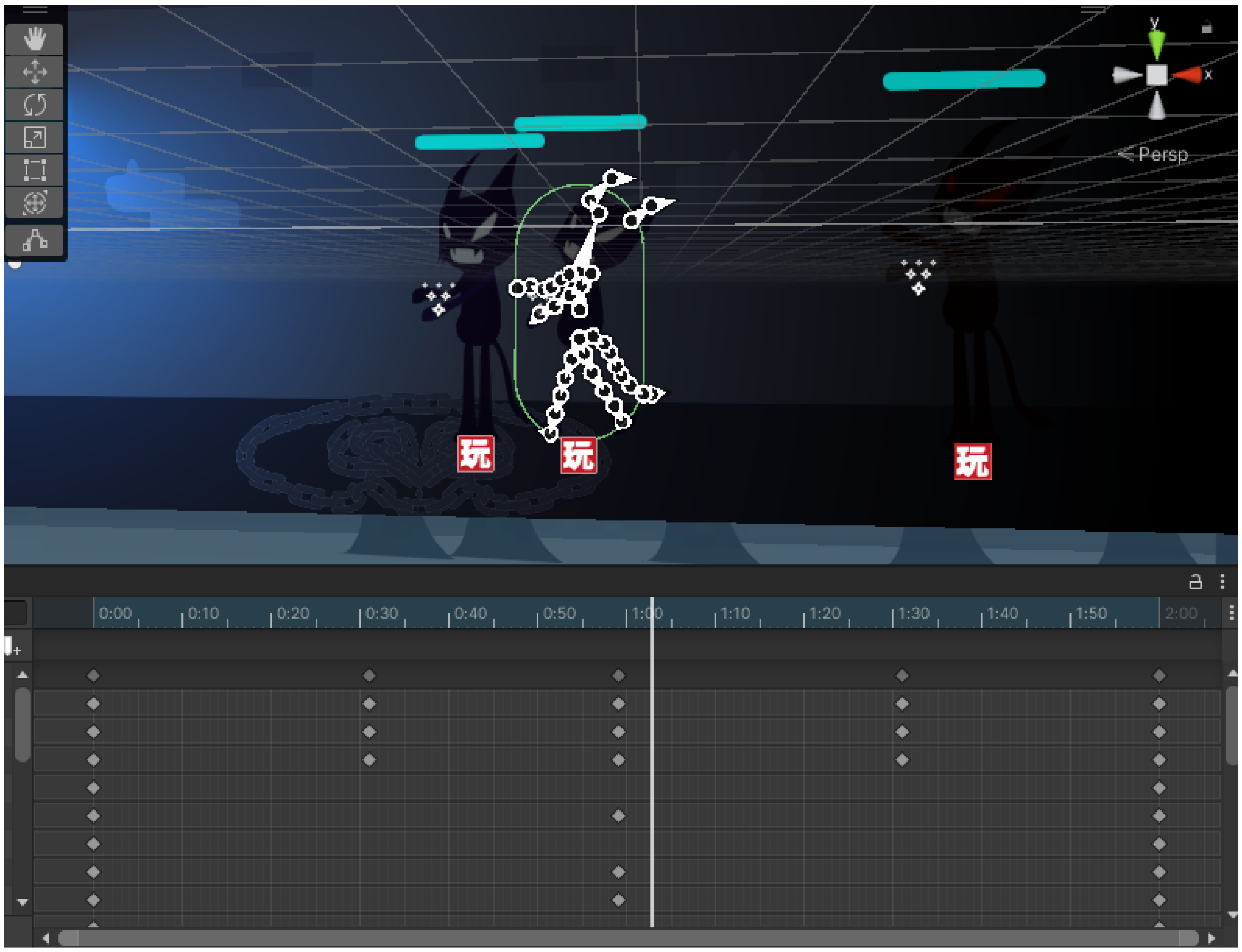


Fig 4.10.2: Enemy Follow Animation

The second animation is the enemy attack animation:

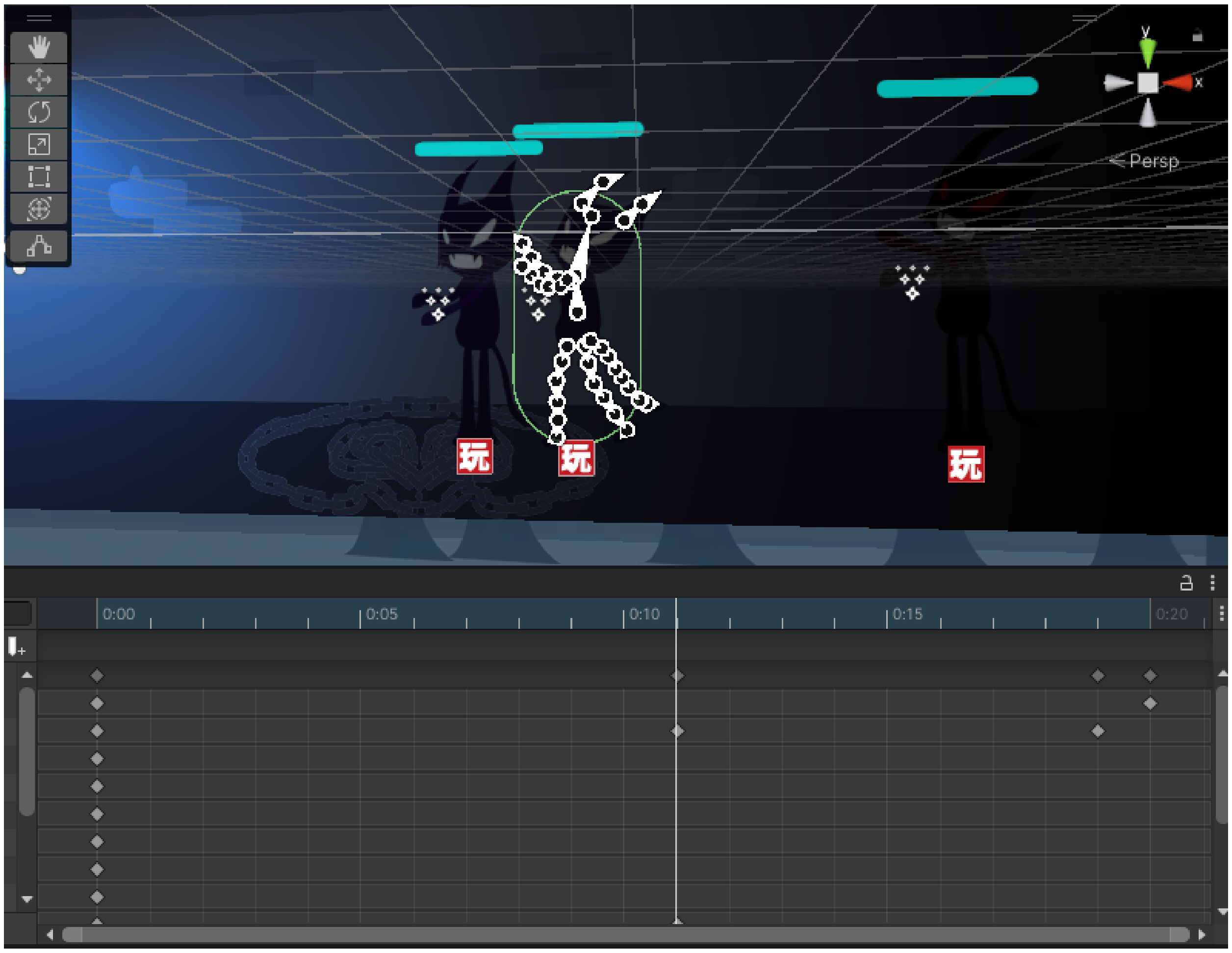


Fig 4.10.2: Enemy Attack Animation

The third animation is the enemy gets hurt animation which has particles included:

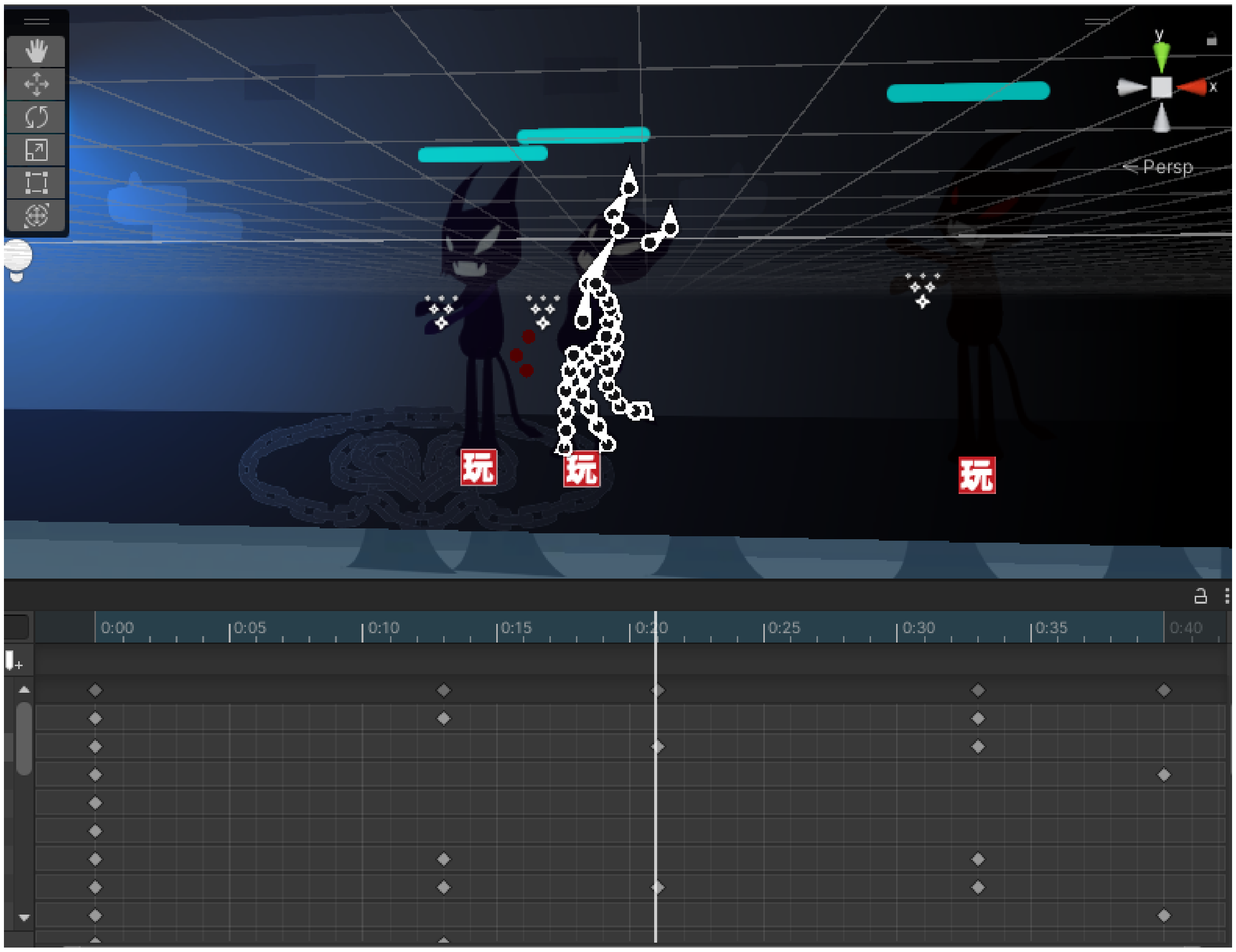


Fig 4.10.2: Enemy Get Hurt Animation

The fourth animation is the enemy death animation. It plays when the enemy dies:

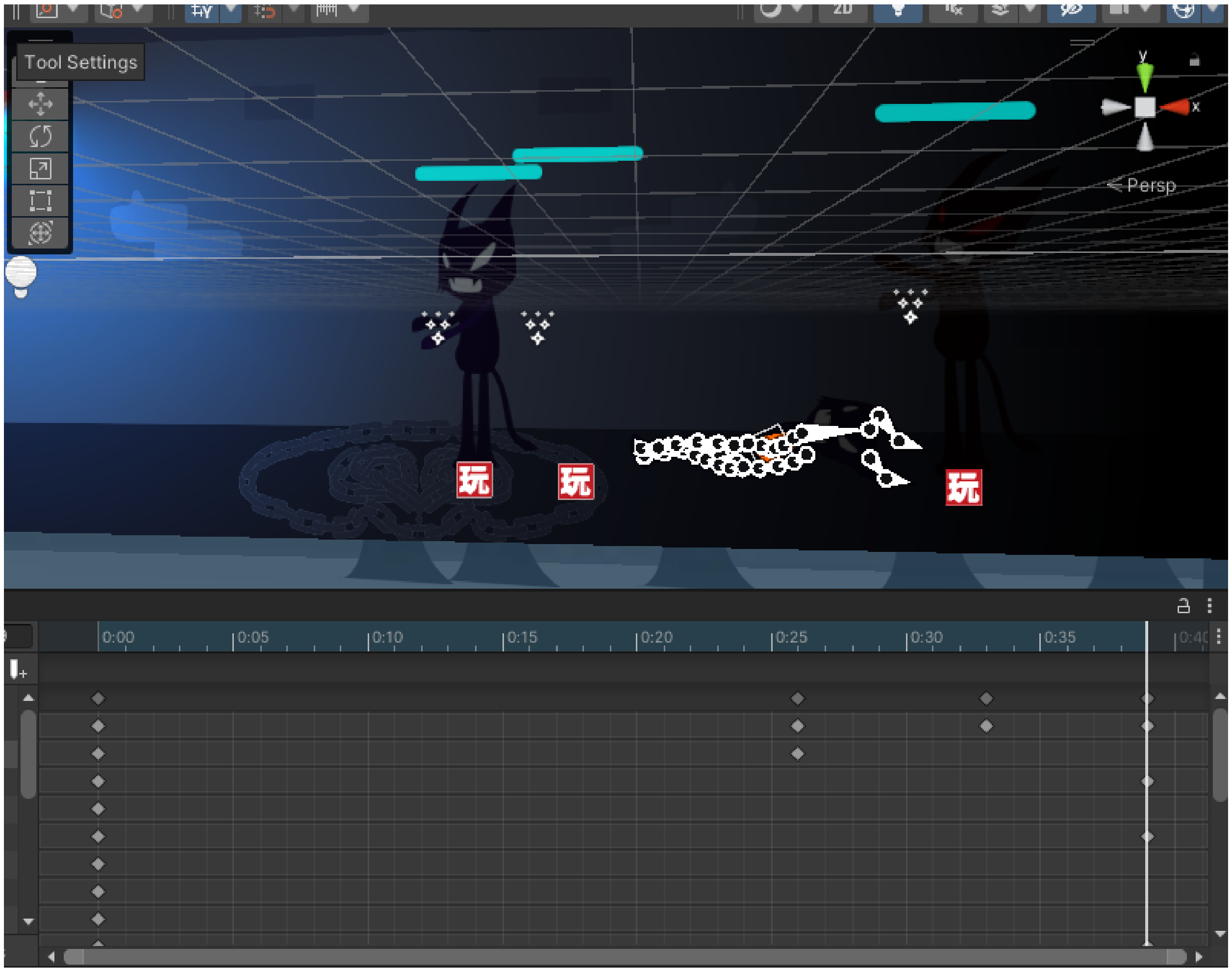
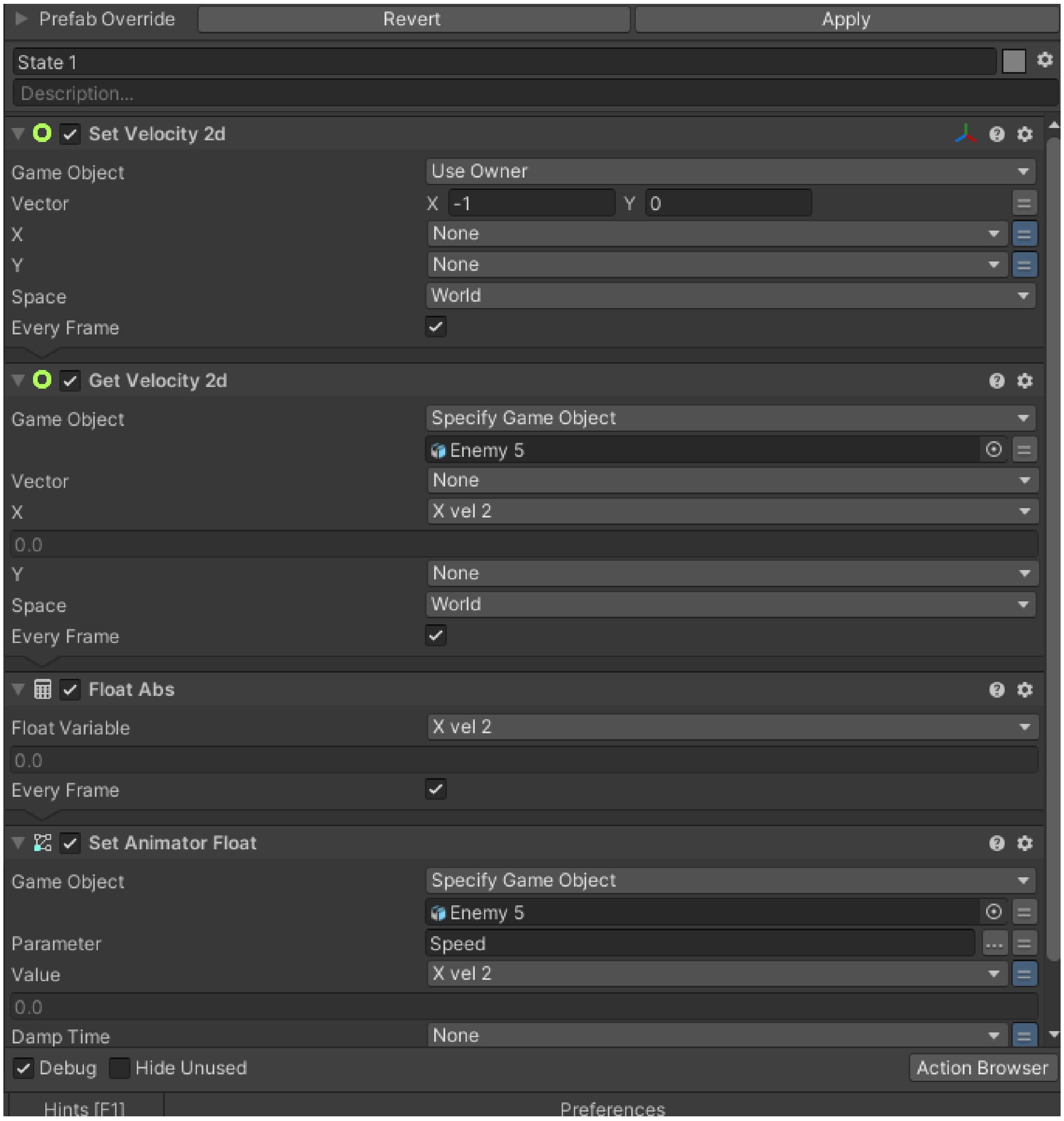


Fig 4.10.2: Enemy Death Animation

4.10.3:Creating the Main Game (Enemy Fsm):

The enemy has several fsm inside them, the first one is the fsm that helps the enemy to walk towards the player.



Prefab Override | Revert | Apply

State 1

Description...

Set Velocity 2d

Game Object	Use Owner
Vector	X -1 Y 0
X	None
Y	None
Space	World
Every Frame	✓

Get Velocity 2d

Game Object	Specify Game Object
	Enemy 5
Vector	None
X	X vel 2
	0.0
Y	None
Space	World
Every Frame	✓

Float Abs

Float Variable	X vel 2
	0.0
Every Frame	✓

Set Animator Float

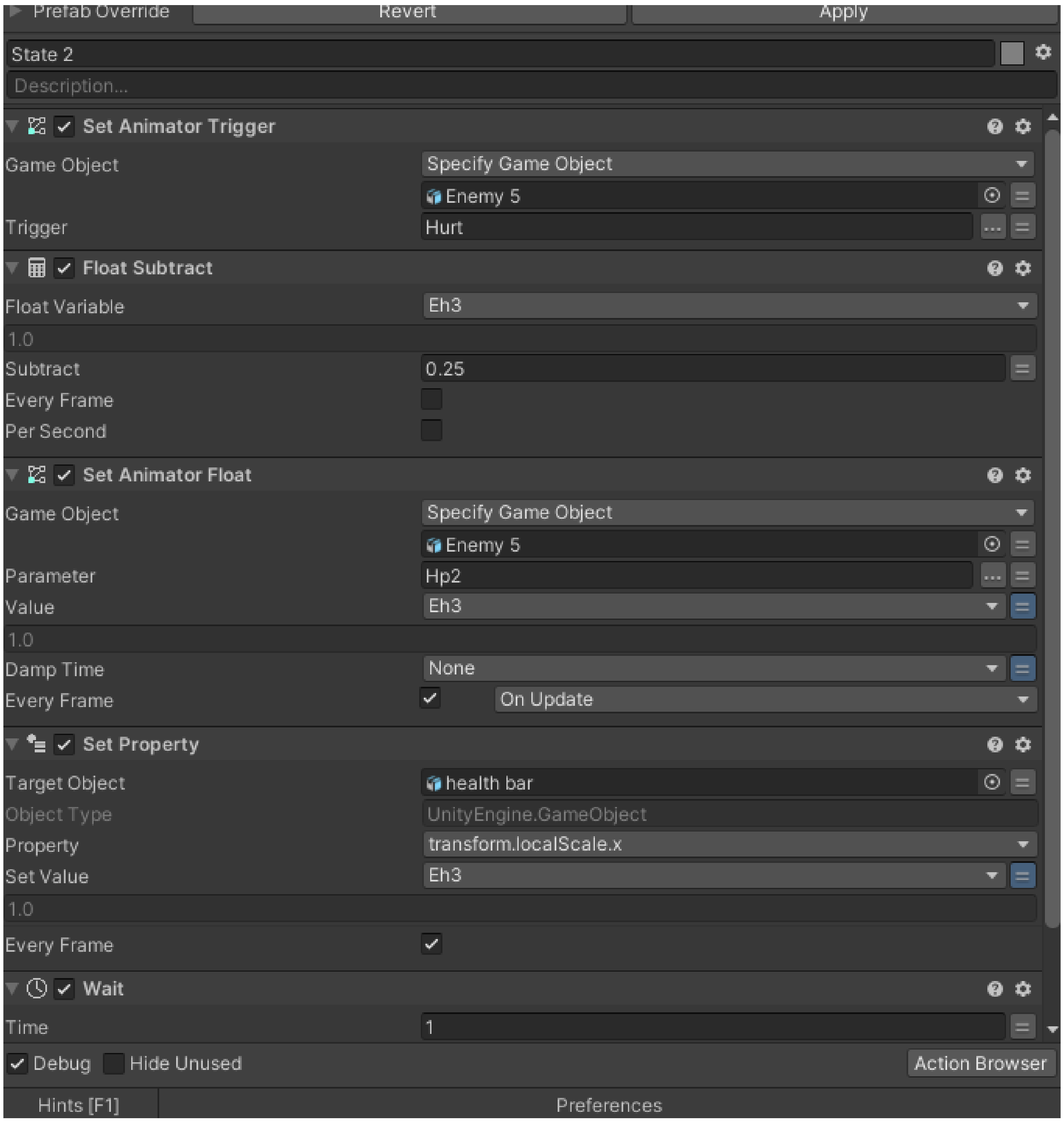
Game Object	Specify Game Object
	Enemy 5
Parameter	Speed
Value	X vel 2
	0.0
Damp Time	None

✓ Debug ☐ Hide Unused | Action Browser

Hints [F1] | Preferences

Fig 4.10.3: Enemy Walkto Fsm

Next is the hurt fsm that determines when the enemy get attack by the player it will reduce the enemy health and will play the hurt animation:



Prefab Override | Revert | Apply

State 2

Description...

Set Animator Trigger

Game Object	Specify Game Object
	Enemy 5
Trigger	Hurt

Float Subtract

Float Variable	Eh3
	1.0
Subtract	0.25
Every Frame	☐
Per Second	☐

Set Animator Float

Game Object	Specify Game Object
	Enemy 5
Parameter	Hp2
Value	Eh3
	1.0
Damp Time	None
Every Frame	✓ On Update

Set Property

Target Object	health bar
Object Type	UnityEngine.GameObject
Property	transform.localScale.x
Set Value	Eh3
	1.0
Every Frame	✓

Wait

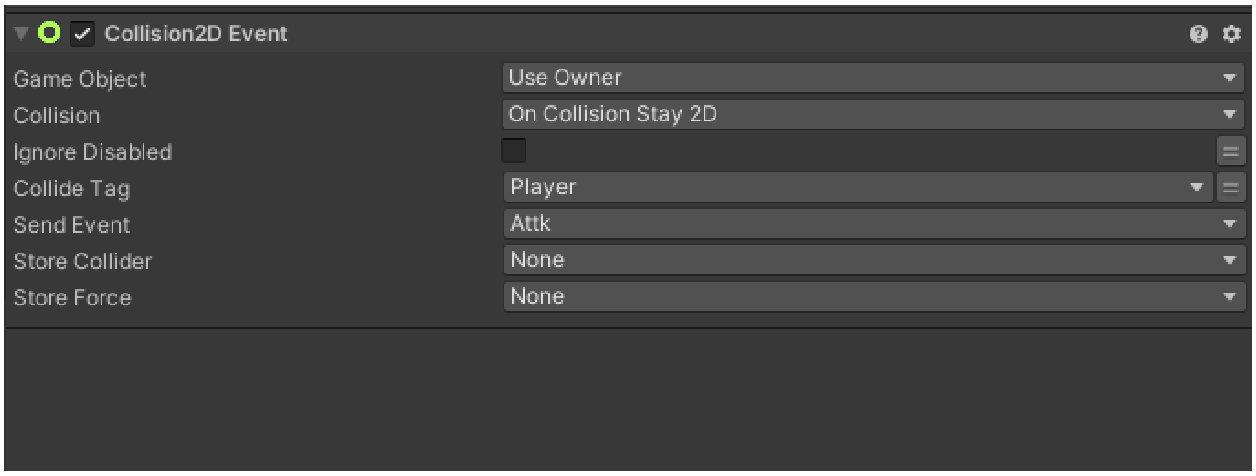
Time	1

✓ Debug ☐ Hide Unused | Action Browser

Hints [F1] | Preferences

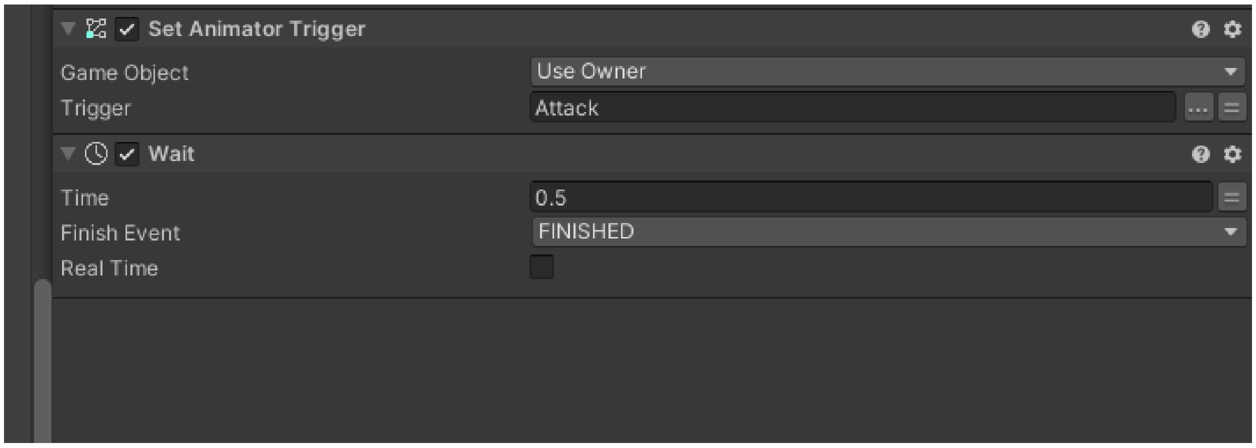
Fig 4.10.3: Enemy Hurt Fsm

Next is the attack fsm that get triggered when a player is in the range of the enemy attack zone ant the enemy attacks the player and play the attack animation:



Collision2D Event

Game Object	Use Owner
Collision	On Collision Stay 2D
Ignore Disabled	☐
Collide Tag	Player
Send Event	Attk
Store Collider	None
Store Force	None



Set Animator Trigger

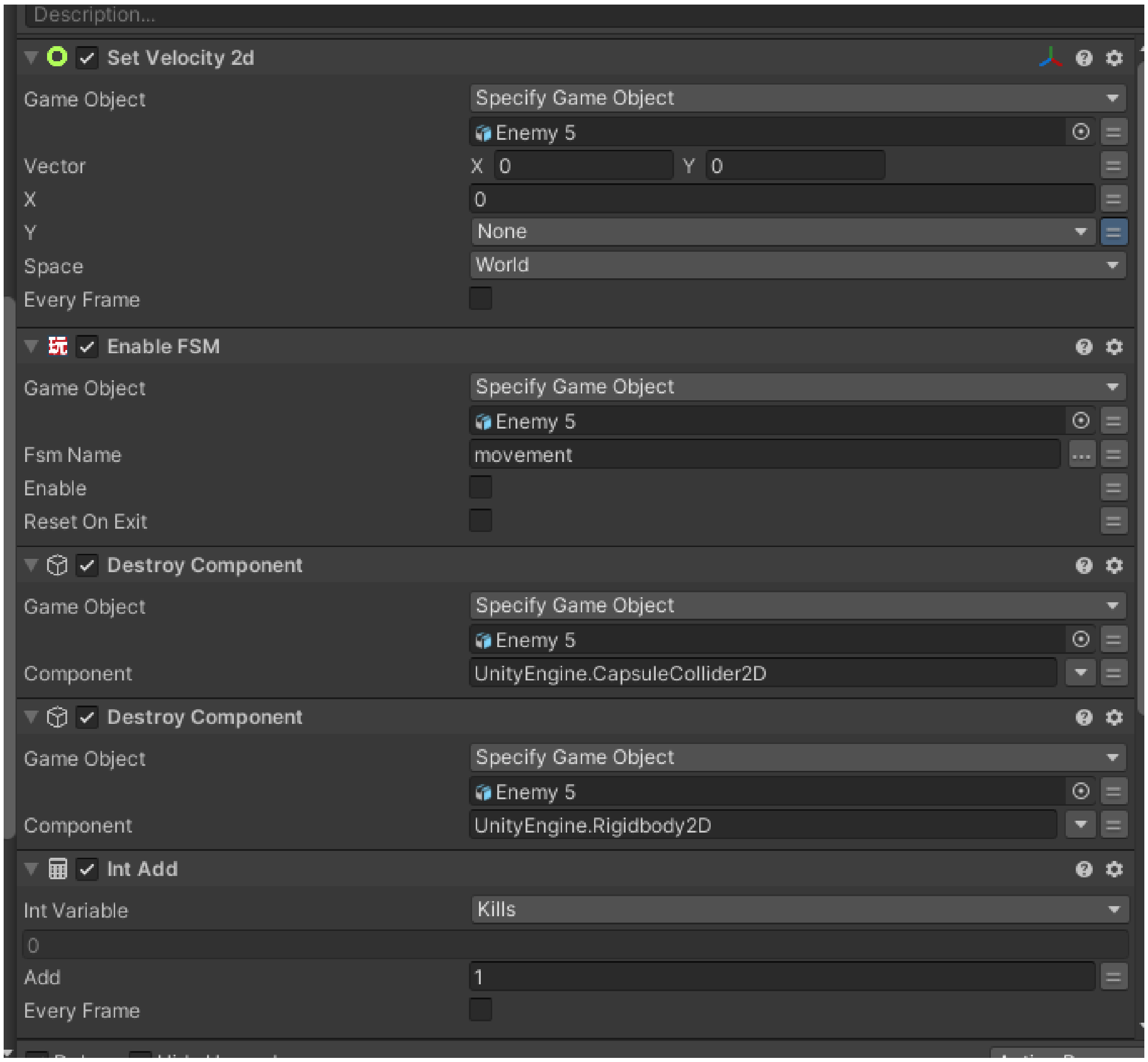
Game Object	Use Owner
Trigger	Attack

Wait

Time	0.5
Finish Event	FINISHED
Real Time	☐

Fig 4.10.3: Enemy Attack Fsm

The last fsm is the death fsm, it disables all the properties of the enemy on death:



Description...

Set Velocity 2d

Game Object	Specify Game Object
	Enemy 5
Vector	X 0 Y 0
X	0
Y	None
Space	World
Every Frame	

Enable FSM

Game Object	Specify Game Object
	Enemy 5
Fsm Name	movement
Enable	
Reset On Exit	

Destroy Component

Game Object	Specify Game Object
	Enemy 5
Component	UnityEngine.CapsuleCollider2D

Destroy Component

Game Object	Specify Game Object
	Enemy 5
Component	UnityEngine.Rigidbody2D

Int Add

Int Variable	Kills
0	
Add	1
Every Frame	

Fig 4.10.3: Enemy Death Fsm

CHAPTER 5

LIMITATION AND CHALLENGES

5.1: Limitation:

Making a game takes a lot of time and sanity and we were lacking both. We had to struggle everyday to pull out the best from our project, Our supervisor has supported us everytime we were facing problems but me and my teammates is completely amature on game development and except for the design, part that why we had to use visual scripts and it is a huge limitation because you can not do everything with the visual scripts. I did not have enough scopes to use it the way I wanted it to. Also, game development requires powerful computers and both of us were also lacing on it since we were working on our low end laptop. And because of that, the process of our work was very slow and once we lost almost all of our works of the project because our computer couldn't handle the work and lost windows. There were many other limitations and challenges.

5.1: Challenges:

Making the game was also super challenging I explained in the limitation that we didn't have the proper equipment to produce what we thought of it but we did our best to pull out as much as we could out of what was available for us. The bigger challenge was the distance between me and my teammate. We had to work online because we live in a separate district and far from each other so there weren't many chances for us to work without being online.

CHAPTER 6

CONCLUSION

6.1: Conclusion:

Making this game has given me the opportunity to learn a lot of things about game development. It has allowed me to explore several development zones and I have learned a lot of unique techniques.
I have learned new ways of doing things.
The world of game development is vast. and Since I used visual scripting to make this game I have gained an understanding of a lot of game logic and started learning LuaU to work on a game development platform called Roblox.
Overall it was a pain more gain kind of journey which changed the way I used to see the multimedia world.

Reference:

[1] “Setting up the plot of the game from the information about the virus”Mayoclinic,[Online].Available: **https://www.mayoclinic.org/diseases-conditions/toxoplasmosis/symptoms-causes/syc-20356249#:~:text=Overview,to%2Dchild%20transmission%20during%20pregnancy.** [Accessed: 2/7/2022]

[2] “Game movement mechanism,” Medium.[Online].Available: **https://medium.com/@chamo.wijetunga/movement-of-a-2d-player-in-unity-aff2b8f02fb5** [Accessed: 5/7/2022]

[3] “Playmaker help,” Hutonggames. [Online].Available : **https://hutonggames.fogbugz.com/** [Accessed: 10/7/2022]

[4] “Getting in-game help,” Unity.Com. [Online]. Available: **https://unity.com/how-to/beginner-video-game-resources** [Accessed: 12/7/2022]

[5] “Unity Assets,” Assetstore [Online]. Available: **https://assetstore.unity.com/** [Accessed: 15/7/2022]

[6] “Playmaker walk nodes,” Youtube [Online]. Available: **https://youtu.be/1Aozzq2Hsko** [Accessed: 22/7/2022]

[7] “Playmaker Player damage system,” Youtube. [Online]. Available:: **https://youtu.be/NMS94nVz4EA** [Accessed: 2/8/2022]

[8] “2D game environment design and placement,” Youtube. [Online]. Available: **https://youtu.be/6XDPkGzt7Yk** [Accessed: 6/8/2022]

[9] “2D game lighting setup” Youtube. [Online]. Available: **https://youtu.be/6XDPkGzt7Yk** [Accessed: 12/8/2022]

[10] “2D game shader system,” Massivemonster. [Online], Available: **https://massivemonster.co/blog/intro-to-2d-unity-shaders** [Accessed: 14/8/2022]

[11] “2D game enemy system,” Youtube. [Online]. Available:
https://youtu.be/siS0Sboz5E8 [Accessed: 18/8/2022]

[12] “Unity build guide,” Unity3D. [Online]. Available:
https://docs.unity3d.com/Manual/PublishingBuilds.html [Accessed: 23/8/2022]