National College of Ireland

BSc (Honours) in Computing

Software Development

Academic Year 2020/2021

Joshua Hall

x16334501

x16334501@student.ncirl.ie

Nature’s Fury

Technical Report

Contents

Executive Summary

1.0 Introduction

1.1. Background

As we progress into the future it is getting easier and easier to join the behemoth of an industry that is the video game industry. If we look at all the tools and support available to us today, making a video game is as accessible as making breakfast. With tools such as Unreal Engine, Unity, GameMaker, Godot, CryEngine and many more and the hundreds of online tutorials available to you on the web (Brackeys, GDQuest), you can create your first video game in minutes.

From a young age I had an interest in making games. When I was in the cub scouts as a child I made my own board game, like snakes and ladders but with added rules. Around the same time period I was reading a book called 'Island of the Lizard King' by Ian Livingstone. This was a single-player adventure gamebook where the reader was tasked with choosing their own story to reach the end of the book. You would solve puzzles and fight enemies like Dungeons and Dragons. It was after reading this I created my first 'video game'. I found in Microsoft PowerPoint I could hyper link one slide to another, so I was able to create a story where at the end of each slide the reader could choose which slide they would advance to next, allowing for the reader to create their own story. I eventually returned to this concept in my first year at the National College of Ireland. After I was taught how to create websites and tasked with creating a website for my web design project, I decided to create the same choose your path style game as before. It was very basic as I did not understand JavaScript, so I was limited to just navigating pages without creating player statistics. It was not until I started my 3rd year internship that I finally decided to work on the JavaScript end. During the free time in my internship, I created a new choose your path game this time with variables and player stats to make it more engaging (Hall, 2020). Finally, when entering 4th year, I decided to create a game using an appropriate software that was best for game creation.

The game Pokémon was released in 1996 and since then its popularity has only increased over the years as they continue to release the same style of game over and over with new features. As the years have gone on the fans of the earlier games have made complaints about the newer versions of the games as a lot has changed to meet the expectations of new content. This leaves a gap for other creators to make similar games that fill the issues consumers have with current Pokémon games, examples of this are Temtem and Nexomon Extinction. I also played Pokémon a lot as a child and it was one of my favourite games, so I have decided to make something similar due to it being an interest of mine. I intend to have more emphasis on the battling aspect of Pokémon as that is the part I enjoy the most.

1.2. Aims

The aim for this project has changed slightly since the project proposal. My original aim and end goal are to create a fully functional RPG. With the limited time I have been allocated to complete this project that goal is unattainable with my small knowledge of game design. Originally, I decided I would have a limited amount of character movement, exploration, dialogue and one or two battle sequences to set the scene for a playable game in the future. I have decided to work more on the battle sequence side of the project as by doing this I will be creating a game that can be playable without reaching the end goal, a standalone battle simulation game. I am going to create a small number of characters you can battle with and I hope to implement multiplayer so you can battle other people, failing that I will implement an AI. If I can complete this within the time frame, I will have two options to add more content. The first option being improving upon the already existing battle simulator and adding more playable characters each with their own unique abilities and actions. The second option is that I begin creating the world for the RPG. This will be the adventuring side where I build my story.

I hope this project will set me up and help me decide whether I wish to follow the route of game creation for my future occupation. If this game is a success, I will either upload it as a finished project online or I will continue working on it until it reaches the original aims I set for myself.

1.3. Technology

I will be using Unity to create my game. It is one of the largest software out there for game creation on multiple platforms. I find the GUI for unity is very easy to use allowing for an enjoyable and stress-free experience when trying to create games. I have not used any other software for game creation so I cannot compare it to others. Unity also comes with a free/paid for asset store which means I am not restricted to creating my own assets. C# is the primary language used in Unity and it is the one I'll be using to create my game.

I will be writing my scripts for Unity using Visual Studio. I have used visual studio in the past for other college projects and have had no issues with it. It is the default IDE used with Unity.

I will be using Gimp for editing and creating sprites. Gimp is a free to use picture editing software that I have limited experience with. I have no other software to compare it to as it is the only one I have used.

I will use Itch.io to upload my project online for free.

1.4. Structure

Section 1 discusses the reason I chose this project, my aims for this project and the technology I used.

Section 2 discusses the process required for the user to use the project. Step by step guide.

Section 3 discusses the learning outcome for undertaking my project.

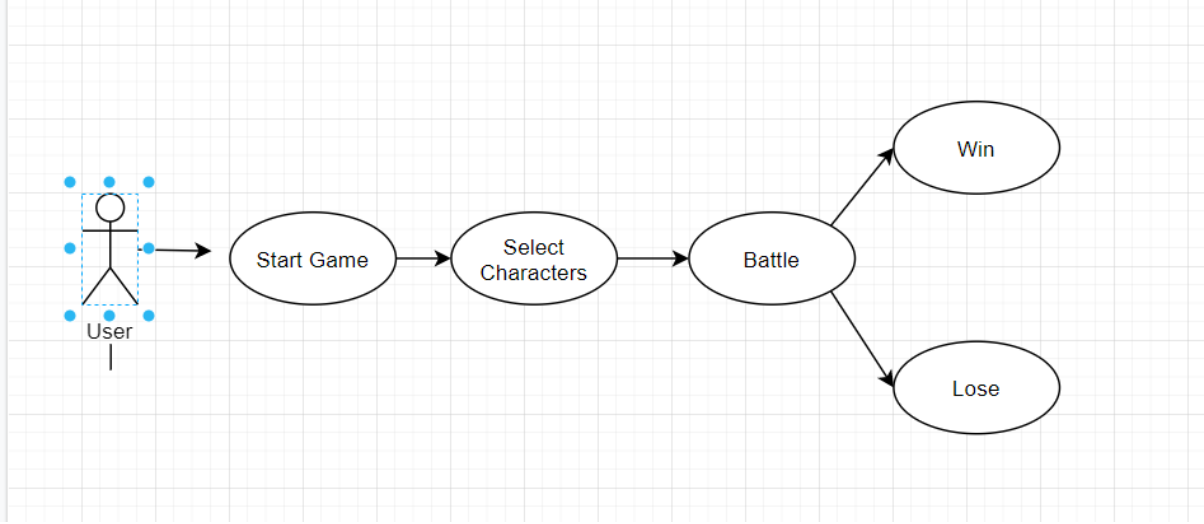
Section 4 discusses the future of my project and where I hope it will go.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1 Character Selection

2.1.1.2.1. Description & Priority

This is an essential aspect of the game as you can't play the game without being able to select your characters. The process involved is looking at the characters available to be chosen and reading through their statistics. After you have decided what characters you like the most you select three of them and continue into the game.

2.1.1.2.2. Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

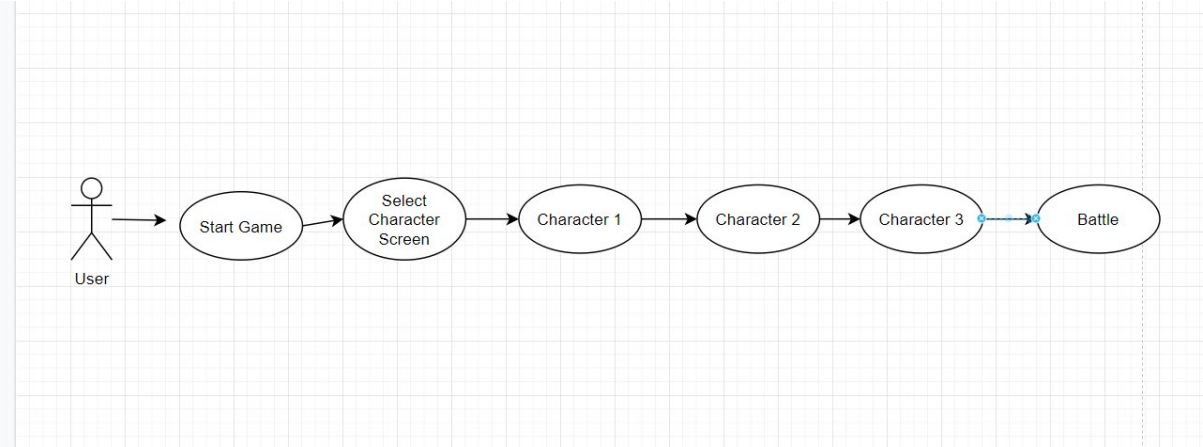
Scope

The scope of this use case is to allow the player to gather information on the units within the game and choose the units they wish to use.

Description

This use case describes the process of how a player chooses their characters.

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode. The game has started and it is waiting for you to choose your characters.

Activation

This use case starts when the user runs the game.

Main flow

1. The user chooses their first character.
2. The user chooses their second character.
3. The user chooses their third character.
4. The user presses the start game button.

Alternate flow

A1 : User doesn't like chosen character.

1. The user chooses their first character.
2. The user doesn't like their first character and chooses a different character.
3. The user chooses their second character.
4. The user chooses their third character.

Exceptional flow

E1 : Error

1. The user cannot progress due to a missing variable.

Termination

Process is terminated when the user has chosen all three of their characters and presses the start game button.

Post condition

You progress onto the battle sequence.

2.1.1.3. Requirement 2 Battle Sequence

2.1.1.3.1. Description & Priority

This is the main aspect of the game, without this section there would be no game. This is where the user is able to use their chosen characters to battle the AI characters.

2.1.1.3.2. Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

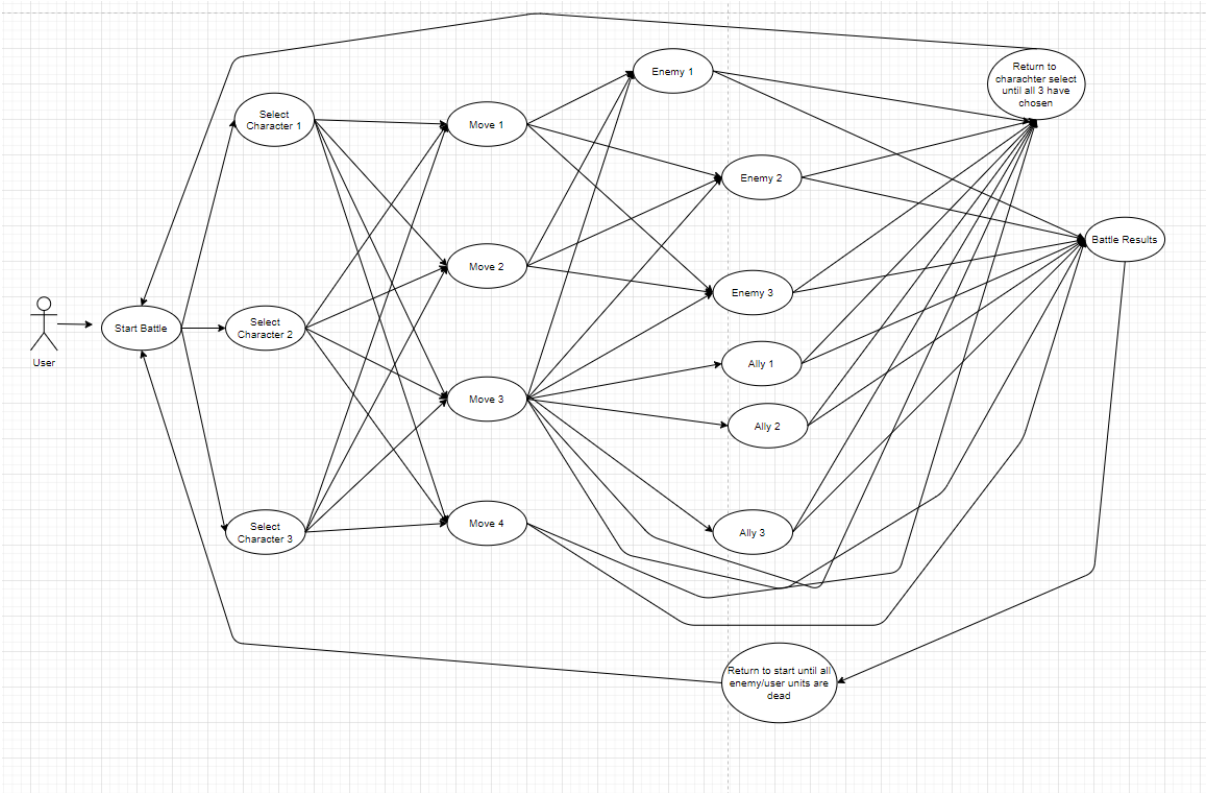
Scope

The scope of this use case is to allow the player to fight the AI.

Description

This use case describes the process of how a player fights the AI.

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode. The fight has begun, the system is awaiting your decision.

Activation

This use case starts when the user chooses a character to attack with.

Main flow

1. The player chooses one of their characters to attack with.
2. The player chooses a move to use.
3. The player chooses an enemy to attack.
4. The player repeats this process until all three of their characters have chosen a move.
5. The battle sequence then starts taking all the player's chosen moves and all the enemies chosen moves and begins playing-out the actions of the selected moves.
6. This will continue until the player is declared the victor.

Alternate flow

A1 : AI wins

1. The player chooses one of their characters to attack with.
2. The player chooses a move to use.
3. The player chooses an enemy to attack.
4. The player repeats this process until all three of their characters have chosen a move.
5. The battle sequence then starts taking all the player's chosen moves and all the enemies chosen moves and begins playing-out the actions of the selected moves.
6. This will continue until the AI is declared the victor.

A2 : Choosing supportive moves.

1. The player chooses one of their characters to attack with.
2. The player chooses a move to use.
3. The player chooses an ally to heal.
4. The player repeats this process until all three of their characters have chosen a move.
5. The battle sequence then starts taking all the player's chosen moves and all the enemies chosen moves and begins playing-out the actions of the selected moves.
6. This will continue until the player is declared the victor.

Exceptional flow

E1 : Denied progression

1. The steps cannot progress as there is a variable missing in the script.

E2 : Wrong move

1. The player chooses one of their characters to attack with.
2. The player chooses a move to use.
3. The player chooses an ally to heal.
4. The player repeats this process until all three of their characters have chosen a move.
5. The battle sequence then starts taking all the player's chosen moves and all the enemies chosen moves and begins playing-out the actions of the selected moves.
6. The wrong actions occur than what was selected.
7. This will continue until the player is declared the victor.

Termination

Process is terminated when player has chosen all their characters, moves and who to attack.

Post condition

The player or AI has won the game.

2.1.2. Data Requirements

The user will need to have access to wireless internet and the website itch.io.

2.1.3. User Requirements

The User will require a PC or Laptop to play the game. The game should not require high PC specs to run.

2.1.4. Environmental Requirements

The only impact on the environment is the mental health of the User. The excessive use of video games can become an addiction. The user should contact an appropriate specialist if you believe you are suffering from this addiction.

2.1.5. Usability Requirements

The game will be easy to use, young children and elderly people should be able to play this game.

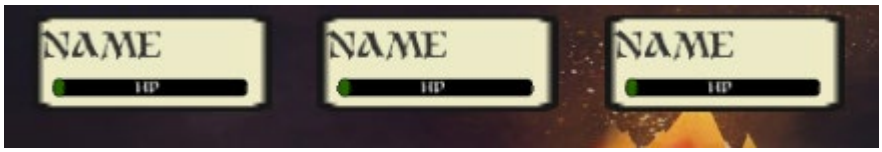
2.2. Design & Architecture

2.2.1. Spawn Points

These are the areas that the sprites can spawn on. There are six spawn points for the active sprites. Three of these are for the player and the other three are for the enemy. When the units are spawned into the game, they are placed on top of the Spawn Point.

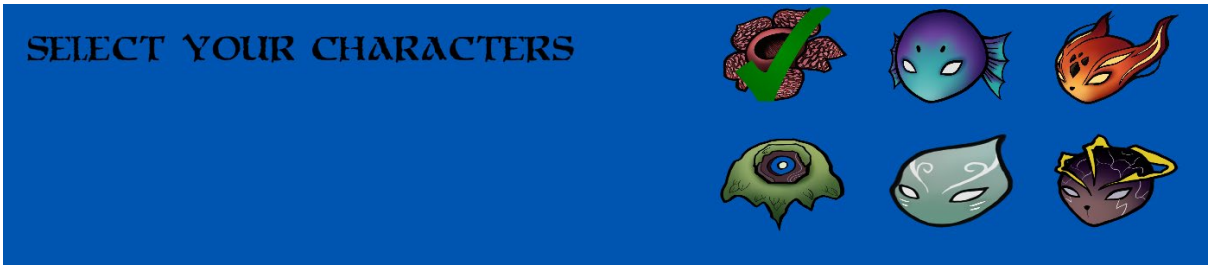
2.2.2. Unit HUDs

The unit's Health is displayed on one of the six unit HUDs. Their health is displayed via a percentage bar and specific numbers. The units name is also displayed on the HUD. These HUDs are updated whenever a unit receives damage or healing.



2.2.3. Character Selection

The user is given the option to choose three different sprites to use before commencing the battle. During this period, they can view the units' statistics prior to making their choice.



2.2.4. Sprites

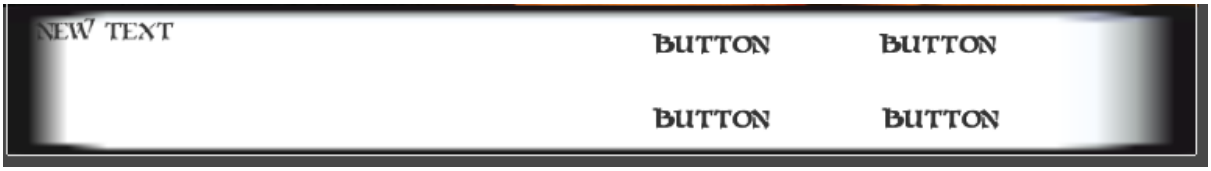
Each sprite has their own unique name and move set for combat along with their own personal base damage, HP bar, speed stat and passive. Each sprite will also have a unique design. The sprites have been modelled off generic aspects of nature.

2.2.5. Sprite Animation

Each sprite has their own animation while fighting, these were created by moving the individual pieces of the art from point A to point B within sixty frames. This method of animation was used as only one sprite had to be drawn instead of frame-by-frame animation with the movement being redrawn every time.

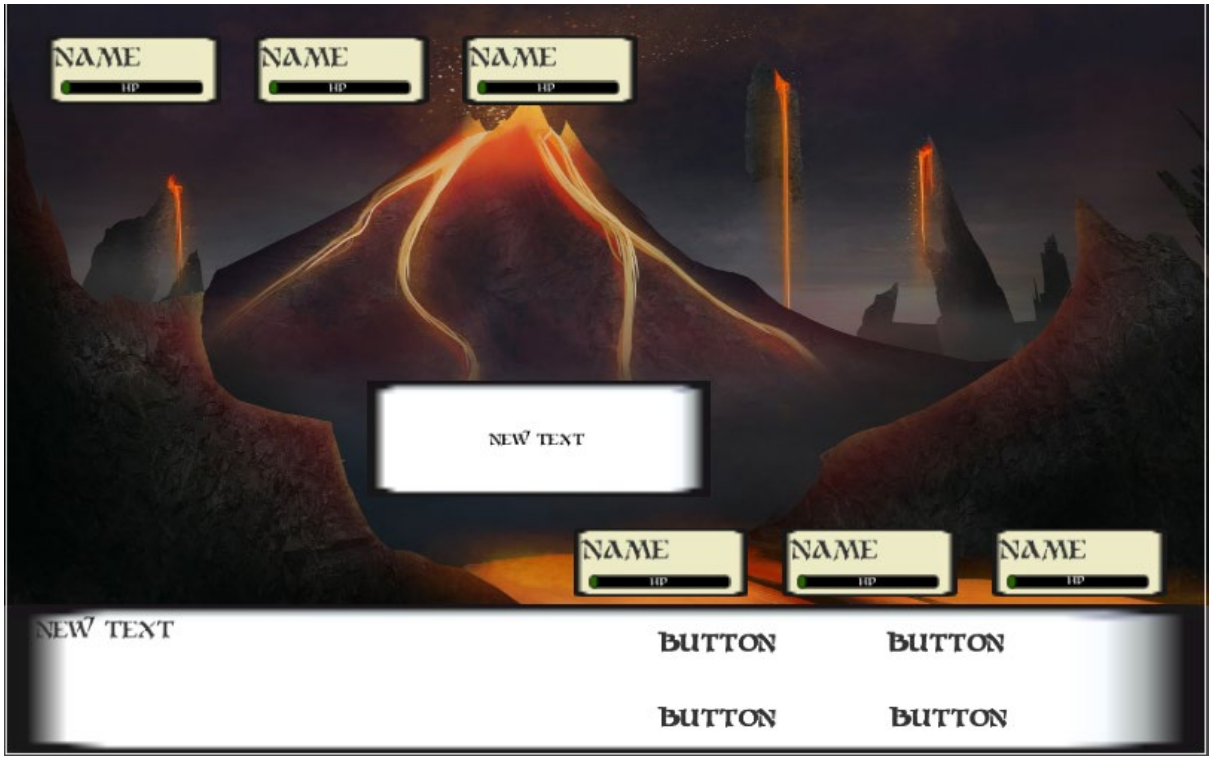
2.2.6. Dialogue Box

This feature tells the user what to do during play time, from here they can make their choices and it will display what actions are currently occurring.



2.2.7. Battle Sequence

This is where all the different components of the game are displayed and where all the actions occur.

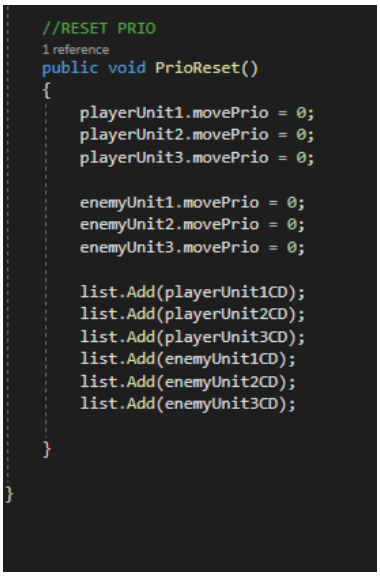


2.3. Implementation

2.3.1. Move Priority

This was the script that began snowballing my entire project. Prior to this script existing I was stuck on how to move forward with my project. It took me almost three weeks of constant dead ends until I eventually figured out how to create the script, the biggest reason for this was due to my limited knowledge of unity.

Every unit has a unique speed stat, this stat controls when a specific unit can use its ability when going through the battle sequence. The script that calculates which unit moves first based off the speed stat goes through the following process. The speed stat of each unit is taken into a list. The max value within the list is then found, a loop then begins randomly choosing the six different units to see if their speed matches the highest. The reason it chooses by random is because that although the speed stats are unique between different units, if the player and enemy AI have the same unit the speed stat would be the same. So, the random generator allows for them to choose randomly if the speed stat is the same. After deciding on a unit, that unit is given priority one (This unit moves first) and then the unit's speed is set to zero and the loop is cut off. This process is then continued until all units have been given a priority. This script occurs before the start of each turn.

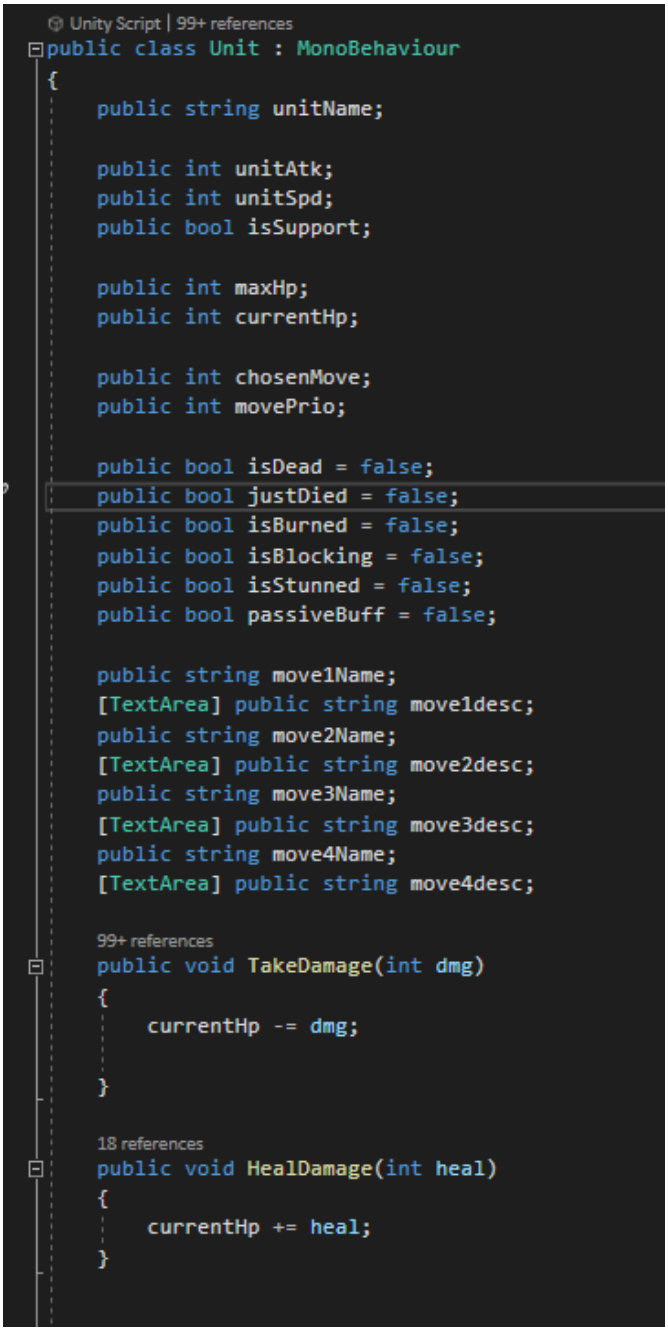


```
    //RESET PRIO
    1 reference
    public void PrioReset()
    {
        playerUnit1.movePrio = 0;
        playerUnit2.movePrio = 0;
        playerUnit3.movePrio = 0;

        enemyUnit1.movePrio = 0;
        enemyUnit2.movePrio = 0;
        enemyUnit3.movePrio = 0;

        list.Add(playerUnit1CD);
        list.Add(playerUnit2CD);
        list.Add(playerUnit3CD);
        list.Add(enemyUnit1CD);
        list.Add(enemyUnit2CD);
        list.Add(enemyUnit3CD);

    }

}
```

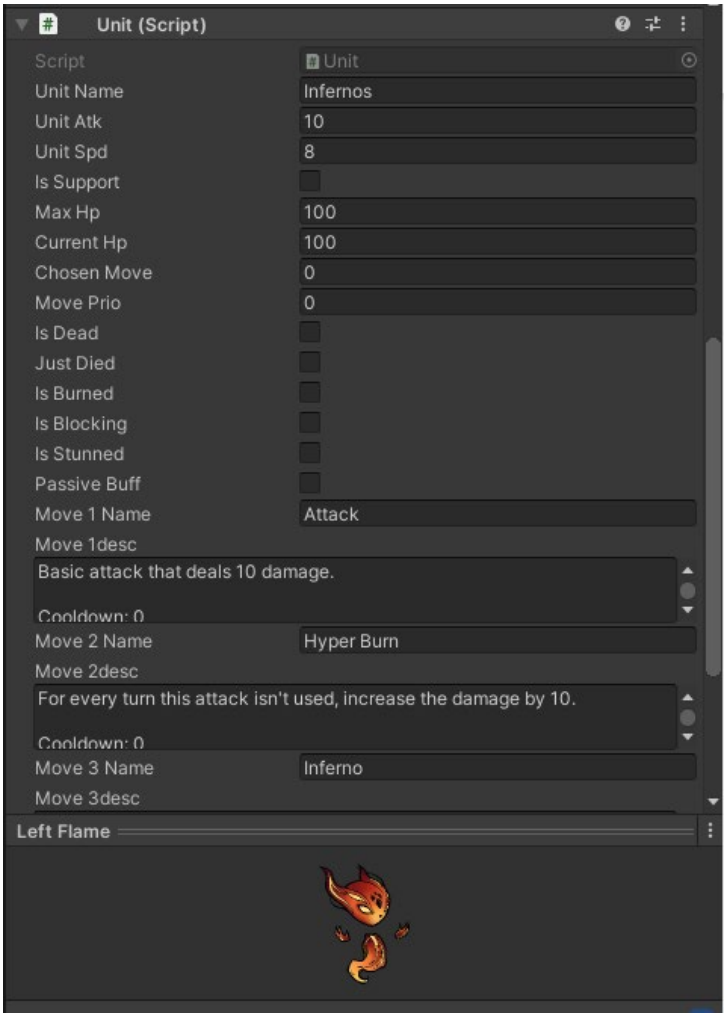
```
//MOVE FIRST FUNCTION
1 reference
public void MovePrio()
{
    max = list.Max();

    //first move check
    for (int i = 0; i < 1000; i++)
    {
        int num = Random.Range(0, 6);

        if (num == 0 && max == playerUnit1CD )
        {
            playerUnit1.movePrio = 1;
            list.Sort();
            list.RemoveAt(5);
            playerUnit1CD = 0;
            i = 1001;
        }
        if (num == 1 && max == playerUnit2CD )
        {
            playerUnit2.movePrio = 1;
            list.Sort();
            list.RemoveAt(5);
            playerUnit2CD = 0;
            i = 1001;
        }
        if (num == 2 && max == playerUnit3CD )
        {
            playerUnit3.movePrio = 1;
            list.Sort();
            list.RemoveAt(5);
            playerUnit3CD = 0;
            i = 1001;
        }
        if (num == 3 && max == enemyUnit1CD )
        {
            enemyUnit1.movePrio = 1;
            list.Sort();
            list.RemoveAt(5);
            enemyUnit1CD = 0;
            i = 1001;
        }
        if (num == 4 && max == enemyUnit2CD )
        {
            enemyUnit2.movePrio = 1;
            list.Sort();
            list.RemoveAt(5);
            enemyUnit2CD = 0;
            i = 1001;
        }
        if (num == 5 && max == enemyUnit3CD )
        {
            enemyUnit3.movePrio = 1;
            list.Sort();
```

2.3.2. Unit Script

Every unit uses the same script for their basic statistics (Speed, health, attack, name etc.) along with the active effects (Stunned, blocking, burning, dead etc.). This script is continuously accessed by the other scripts when needing to alter information about the units during combat. The most used of these is the health stat. This is updated frequently throughout the game. Within this script are also two functions for altering the player's health.



```
Unity Script | 99+ references
public class Unit : MonoBehaviour
{
    public string unitName;

    public int unitAtk;
    public int unitSpd;
    public bool isSupport;

    public int maxHp;
    public int currentHp;

    public int chosenMove;
    public int movePrio;

    public bool isDead = false;
    public bool justDied = false;
    public bool isBurned = false;
    public bool isBlocking = false;
    public bool isStunned = false;
    public bool passiveBuff = false;

    public string move1Name;
    [TextArea] public string move1desc;
    public string move2Name;
    [TextArea] public string move2desc;
    public string move3Name;
    [TextArea] public string move3desc;
    public string move4Name;
    [TextArea] public string move4desc;

    99+ references
    public void TakeDamage(int dmg)
    {
        currentHp -= dmg;

    }

    18 references
    public void HealDamage(int heal)
    {
        currentHp += heal;
    }
```



2.3.3. Character Selection

Before the battle can start a player must select the characters they wish to use. This is done by saving three integers depending on what character a player has chosen. There are currently six characters, so the chosen integers are between one to six. These three integers are then passed into the main script where it matches the chosen integers with the units with corresponding integers. The AI then randomly chooses integers from the optional six.

```
0 references
public void FireSelect()
{
    if(fireChosen == false )
    {
        if(selectedCharacter1 == 0 )
        {
            selectedCharacter1 = 1;
            fireChosen = true;
            fireCheck.enabled = true;
        }
        else if(selectedCharacter1 != 0 && selectedCharacter2 == 0)
        {
            selectedCharacter2 = 1;
            fireChosen = true;
            fireCheck.enabled = true;
        }
        else if(selectedCharacter1 != 0 && selectedCharacter2 != 0 && selectedCharacter3 == 0)
        {
            selectedCharacter3 = 1;
            fireChosen = true;
            fireCheck.enabled = true;
        }
    }
    else if (fireChosen == true && selectedCharacter1 == 1)
    {
        fireChosen = false;
        fireCheck.enabled = false;
        selectedCharacter1 = 0;
    }
    else if (fireChosen == true && selectedCharacter2 == 1)
    {
        fireChosen = false;
        fireCheck.enabled = false;
        selectedCharacter2 = 0;
    }
    else if (fireChosen == true && selectedCharacter3 == 1)
    {
        fireChosen = false;
        fireCheck.enabled = false;
        selectedCharacter3 = 0;
    }
}
```

```
0 references
public void StartGame()
{
    PlayerPrefs.SetInt("selectedCharacter1", selectedCharacter1);
    PlayerPrefs.SetInt("selectedCharacter2", selectedCharacter2);
    PlayerPrefs.SetInt("selectedCharacter3", selectedCharacter3);
    SceneManager.LoadScene("Battle");
}
```

```
//BATTLE SETUP
Unity Message | 0 references
void Start()
{
    state = BattleState.START;

    selectedCharacter1 = PlayerPrefs.GetInt("selectedCharacter1");
    selectedCharacter2 = PlayerPrefs.GetInt("selectedCharacter2");
    selectedCharacter3 = PlayerPrefs.GetInt("selectedCharacter3");

    // selecting first player character
    if(selectedCharacter1 == 1)
    {
        playerPrefab1 = rightFlame;
    }
    else if(selectedCharacter1 == 2)
    {
        playerPrefab1 = rightWater;
    }
    else if(selectedCharacter1 == 3)
    {
        playerPrefab1 = rightGrass;
    }
    else if(selectedCharacter1 == 4)
    {
        playerPrefab1 = rightEarth;
    }
    else if (selectedCharacter1 == 5)
    {
        playerPrefab1 = rightWind;
    }
    else if (selectedCharacter1 == 6)
    {
        playerPrefab1 = rightElectric;
    }
```

```
    //selecting first enemy character
    for (int i=0; i <= 1000; i++)
    {
        int enemy1 = Random.Range(0, 6);

        if (enemy1 == 0 && flameUsed == false)
        {
            enemyPrefab1 = leftFlame;
            flameUsed = true;
            i = 1001;
        }
        if (enemy1 == 1 && waterUsed == false)
        {
            enemyPrefab1 = leftWater;
            waterUsed = true;
            i = 1001;
        }
        if (enemy1 == 2 && grassUsed == false)
        {
            enemyPrefab1 = leftGrass;
            grassUsed = true;
            i = 1001;
        }
        if (enemy1 == 3 && earthUsed == false)
        {
            enemyPrefab1 = leftEarth;
            earthUsed = true;
            i = 1001;
        }
        if (enemy1 == 4 && windUsed == false)
        {
            enemyPrefab1 = leftWind;
            windUsed = true;
            i = 1001;
        }
        if (enemy1 == 5 && electricUsed == false)
        {
            enemyPrefab1 = leftElectric;
            electricUsed = true;
            i = 1001;
        }
    }
```

2.3.4. Main Battle System

Almost every single script within my game refers to the battle system script. This script controls the main flow of the game. It begins with choosing the correct characters based off what the user chose in the previous scene along with randomly choosing three AI units for the player to fight against. Following this it spawns all six units on their spawn points. It also runs the function that creates clones of the units, so the original prefabs created within unity are not altered by what happens within the game (This is to make sure that if a unit loses all its health within the game, it doesn't affect the Unit script outside the game). It also initiates the unit passives that commence at the start of the game (the abilities that increase a unit's max health etc.) It then starts the script that decides the move order of the units, alongside choosing the moves the AI are going to use before the player makes their choices. Following this it activates the dialog panel and buttons so the user can begin selecting the characters they wish to attack with, the abilities they want these characters to use and the targets for the abilities. After all the choices have been made the battle phase script begins.

2.3.5. Battle Phase

This script accesses the units move priority variable which was created in the previous script. Using this it can check which unit moves first along with what move they decided to use. After each successful move it checks to see if a unit has died and whether all the player units or AI units are dead. If either of these options were true it would inform the user that a winner has been chosen and the game is over.

```
IEnumerator MovePrioBattleCoroutine()
{
    //PRIO 0 MOVES

    //first attackers
    if (playerUnit1.movePrio == 1 && playerUnit1.isDead == false)
    {
        battleSystem.gameObject.GetComponent<BattlePhaseMoves>().PU1Moves();
    }
    else if (playerUnit2.movePrio == 1  && playerUnit2.isDead == false)
    {
        battleSystem.gameObject.GetComponent<BattlePhaseMoves>().PU2Moves();
    }
    else if (playerUnit3.movePrio == 1 && playerUnit3.isDead == false)
    {
        battleSystem.gameObject.GetComponent<BattlePhaseMoves>().PU3Moves();
    }
    else if (enemyUnit1.movePrio == 1 && enemyUnit1.isDead == false)
    {
        battleSystem.gameObject.GetComponent<BattlePhaseMoves>().EU1Moves();
    }
    else if (enemyUnit2.movePrio == 1 && enemyUnit2.isDead == false)
    {
        battleSystem.gameObject.GetComponent<BattlePhaseMoves>().EU2Moves();
    }
    else if (enemyUnit3.movePrio == 1 && enemyUnit3.isDead == false)
    {
        battleSystem.gameObject.GetComponent<BattlePhaseMoves>().EU3Moves();
    }

    //check deaths 1
```

```
//check deaths 1
if(playerUnit1.isDead == true && playerUnit1.justDied == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "The player's " + playerUnit1.unitName + " was killed";
    playerUnit1.justDied = false;
}
if (playerUnit2.isDead == true && playerUnit2.justDied == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "The player's " + playerUnit2.unitName + " was killed";
    playerUnit2.justDied = false;
}
if (playerUnit3.isDead == true && playerUnit3.justDied == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "The player's " + playerUnit3.unitName + " was killed";
    playerUnit3.justDied = false;
}
if (enemyUnit1.isDead == true && enemyUnit1.justDied == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "The enemy " + enemyUnit1.unitName + " was killed";
    enemyUnit1.justDied = false;
}
if (enemyUnit2.isDead == true && enemyUnit2.justDied == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "The enemy " + enemyUnit2.unitName + " was killed";
    enemyUnit2.justDied = false;
}
if (enemyUnit3.isDead == true && enemyUnit3.justDied == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "The enemy " + enemyUnit3.unitName + " was killed";
    enemyUnit3.justDied = false;
}
if (playerUnit1.isDead == true && playerUnit2.isDead == true && playerUnit3.isDead == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "Enemy Wins";
    playerUnit1.movePrio = 0;
    playerUnit2.movePrio = 0;
    playerUnit3.movePrio = 0;
    enemyUnit1.movePrio = 0;
    enemyUnit2.movePrio = 0;
    enemyUnit3.movePrio = 0;
}
if (enemyUnit1.isDead == true && enemyUnit2.isDead == true && enemyUnit3.isDead == true)
{
    yield return new WaitForSeconds(5f);
    dialogueText.text = "Player Wins";
    playerUnit1.movePrio = 0;
    playerUnit2.movePrio = 0;
    playerUnit3.movePrio = 0;
    enemyUnit1.movePrio = 0;
    enemyUnit2.movePrio = 0;
    enemyUnit3.movePrio = 0;
}
```

2.3.6. Battle Phase Moves Script

This script is the middle ground between the moves and the battle phase script, it extends the selection process, so you know which moves were chosen by which units. I only created this script so less space would be taken up in the battle phase script, it helps with overall organisation and allows for simple scaling.

```
//PLAYER UNIT 1 MOVES
6 references
public void PU1Moves()
{
    if (playerUnit1.tag == "Fire")
    {
        if (playerUnit1.chosenMove == 1)
        {
            battleSystem.gameObject.GetComponent<FireMoves>().PU1FireMove1Btn();
        }
        else if (playerUnit1.chosenMove == 2)
        {
            battleSystem.gameObject.GetComponent<FireMoves>().PU1FireMove2Btn();
        }
        else if (playerUnit1.chosenMove == 3)
        {
            battleSystem.gameObject.GetComponent<FireMoves>().PU1FireMove3Btn();
        }
        else if (playerUnit1.chosenMove == 4)
        {
            battleSystem.gameObject.GetComponent<FireMoves>().PU1FireMove4Btn();
        }
    }

    if (playerUnit1.tag == "Water")
    {
        if (playerUnit1.chosenMove == 1)
        {
            battleSystem.gameObject.GetComponent<WaterMoves>().PU1WaterMove1Btn();
        }
        else if (playerUnit1.chosenMove == 2)
        {
            battleSystem.gameObject.GetComponent<WaterMoves>().PU1WaterMove2Btn();
        }
        else if (playerUnit1.chosenMove == 3)
        {
            battleSystem.gameObject.GetComponent<WaterMoves>().PU1WaterMove3Btn();
        }
        else if (playerUnit1.chosenMove == 4)
        {
            battleSystem.gameObject.GetComponent<WaterMoves>().PU1WaterMove4Btn();
        }
    }
```

2.3.7. Battle Effects

This script controls all the different passives each unit has. Some of these passives activate at the beginning of the battle while others occur at the end of each turn

2.3.8. Units

2.3.8.1. Cloning Data

The unit data needs to be cloned before any changes can be made so as not to alter the original prefabs, this runs before any moves have been selected.

```
//GET CLONE DATA
1 reference
public void GetObjects()
{

    playerUnit1 = GetComponent<BattleSystem>().playerUnit1;
    playerUnit2 = GetComponent<BattleSystem>().playerUnit2;
    playerUnit3 = GetComponent<BattleSystem>().playerUnit3;

    enemyUnit1 = GetComponent<BattleSystem>().enemyUnit1;
    enemyUnit2 = GetComponent<BattleSystem>().enemyUnit2;
    enemyUnit3 = GetComponent<BattleSystem>().enemyUnit3;

    chosenEnemy1 = GetComponent<BattleSystem>().chosenEnemy1;
    chosenEnemy2 = GetComponent<BattleSystem>().chosenEnemy2;
    chosenEnemy3 = GetComponent<BattleSystem>().chosenEnemy3;

    chosenEnemyHUD1 = GetComponent<BattleSystem>().chosenEnemyHUD1;
    chosenEnemyHUD2 = GetComponent<BattleSystem>().chosenEnemyHUD2;
    chosenEnemyHUD3 = GetComponent<BattleSystem>().chosenEnemyHUD3;
}
```

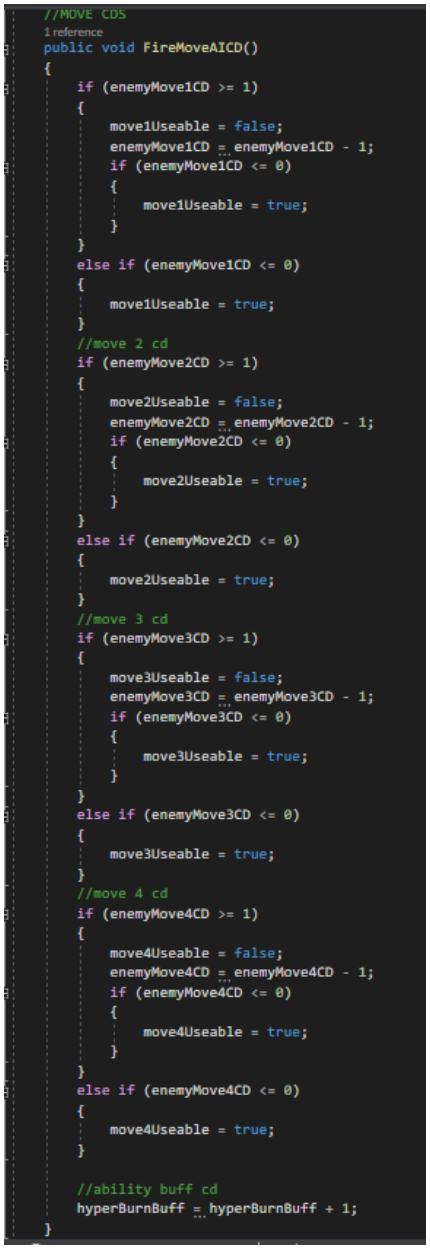
2.3.8.2. Move Cooldowns

Every unit has cooldowns on their moves, whenever a move is used the cooldown timer is increased by X amount. Each time the battle returns to the move selection phase that X will reduce by one until eventually the cooldown reaches zero and a move can be used again.

```
//MOVE CDS
1 reference
public void FireMoveCD()
{
    //p1 move cd
    if (playerUnit1.tag == "Fire")
    {
        //move 1 cd
        if (playerMove1CD >= 1)
        {
            move1btn.enabled = false;
            move1txt.GetComponent<ChangeTextColor>().redColor();
            playerMove1CD = playerMove1CD - 1;
            if (playerMove1CD <= 0)
            {
                move1txt.GetComponent<ChangeTextColor>().greenColor();
                move1btn.enabled = true;
            }
        }
        else if (playerMove1CD <= 0)
        {
            move1txt.GetComponent<ChangeTextColor>().greenColor();
            move1btn.enabled = true;
        }
        //move 2 cd
        if (playerMove2CD >= 1)
        {
            move2btn.enabled = false;
            move2txt.GetComponent<ChangeTextColor>().redColor();
            playerMove2CD = playerMove2CD - 1;
            if (playerMove2CD <= 0)
            {
                move2txt.GetComponent<ChangeTextColor>().greenColor();
                move2btn.enabled = true;
            }
        }
        else if (playerMove2CD <= 0)
        {
            move2txt.GetComponent<ChangeTextColor>().greenColor();
            move2btn.enabled = true;
        }
        //move 3 cd
        if (playerMove3CD >= 1)
        {
            move3btn.enabled = false;
            move3txt.GetComponent<ChangeTextColor>().redColor();
            playerMove3CD = playerMove3CD - 1;
            if (playerMove3CD <= 0)
            {
                move3txt.GetComponent<ChangeTextColor>().greenColor();
                move3btn.enabled = true;
            }
        }
        else if (playerMove3CD <= 0)
        {
            move3txt.GetComponent<ChangeTextColor>().greenColor();
            move3btn.enabled = true;
        }
        //move 4 cd
        if (playerMove4CD >= 1)
        {
            move4btn.enabled = false;
            move4txt.GetComponent<ChangeTextColor>().redColor();
            playerMove4CD = playerMove4CD - 1;
            if (playerMove4CD <= 0)
            {
                move4txt.GetComponent<ChangeTextColor>().greenColor();
                move4btn.enabled = true;
            }
        }
        else if (playerMove4CD <= 0)
        {
            move4txt.GetComponent<ChangeTextColor>().greenColor();
            move4btn.enabled = true;
        }
    }
```

2.3.8.3. AI Move Cooldowns

The AI move cooldown is a much smaller script as it doesn't affect the UI on the player's perspective.



```
//MOVE CDS
1 reference
public void FireMoveAICD()
{
    if (enemyMove1CD >= 1)
    {
        move1Useable = false;
        enemyMove1CD = enemyMove1CD - 1;
        if (enemyMove1CD <= 0)
        {
            move1Useable = true;
        }
    }
    else if (enemyMove1CD <= 0)
    {
        move1Useable = true;
    }
    //move 2 cd
    if (enemyMove2CD >= 1)
    {
        move2Useable = false;
        enemyMove2CD = enemyMove2CD - 1;
        if (enemyMove2CD <= 0)
        {
            move2Useable = true;
        }
    }
    else if (enemyMove2CD <= 0)
    {
        move2Useable = true;
    }
    //move 3 cd
    if (enemyMove3CD >= 1)
    {
        move3Useable = false;
        enemyMove3CD = enemyMove3CD - 1;
        if (enemyMove3CD <= 0)
        {
            move3Useable = true;
        }
    }
    else if (enemyMove3CD <= 0)
    {
        move3Useable = true;
    }
    //move 4 cd
    if (enemyMove4CD >= 1)
    {
        move4Useable = false;
        enemyMove4CD = enemyMove4CD - 1;
        if (enemyMove4CD <= 0)
        {
            move4Useable = true;
        }
    }
    else if (enemyMove4CD <= 0)
    {
        move4Useable = true;
    }

    //ability buff cd
    hyperBurnBuff = hyperBurnBuff + 1;
}
```

2.3.8.4. AI Move Selector

Unlike the player's units which have their moves chosen by the player the AI chooses their moves randomly.

```
//AI MOVE CHOICES
1 reference
public void ChosenFireMoves()
{
    if(enemyUnit1.tag == "Fire")
    {
        for (int i = 0; i < 100; i++)
        {
            int move = Random.Range(0, 4);

            if(move == 0 && move1Useable == true)
            {
                enemyUnit1.chosenMove = 1;
                i = 101;
            }
            else if (move == 1 && move2Useable == true)
            {
                enemyUnit1.chosenMove = 2;
                i = 101;
            }
            else if (move == 2 && move3Useable == true)
            {
                enemyUnit1.chosenMove = 3;
                i = 101;
            }
            else if (move == 3 && move4Useable == true && enemyUnit1.passiveBuff == false && enemyUnit2.passiveBuff == false && enemyUnit3.passiveBuff == false)
            {
                enemyUnit1.chosenMove = 4;
                i = 101;
            }
        }
    }

    if (enemyUnit2.tag == "Fire")
    {
        for (int i = 0; i < 100; i++)
        {
            int move = Random.Range(0, 4);

            if (move == 0 && move1Useable == true)
            {
                enemyUnit2.chosenMove = 1;
                i = 101;
            }
            else if (move == 1 && move2Useable == true)
            {
                enemyUnit2.chosenMove = 2;
                i = 101;
            }
            else if (move == 2 && move3Useable == true)
            {
                enemyUnit2.chosenMove = 3;
                i = 101;
            }
            else if (move == 3 && move4Useable == true && enemyUnit1.passiveBuff == false && enemyUnit2.passiveBuff == false && enemyUnit3.passiveBuff == false)
            {
                enemyUnit2.chosenMove = 4;
                i = 101;
            }
        }
    }
```

2.3.8.5. Infernos

The following scripts are the moves available to the Infernos unit along with the unit's passive.

1. This is a basic attack that can be used every turn.

```
//p1 move 1
1 reference
public void PU1FireMove1Btn()
{
    if(playerUnit1.isStunned == false)
    {
        playerMove1CD = 1;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {
            chosenEnemy1.TakeDamage(playerUnit1.unitAtk);

            if (chosenEnemy1.tag == "Fire")
            {
                chosenEnemy1.isBurned = false;
            }
            else
            {
                chosenEnemy1.isBurned = true;
            }

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
```

2. This move increases by ten damage whenever it is not used.

```
//p1 move 2
1 reference
public void PU1FireMove2Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove2CD = 1;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        hyperBurnBuff = hyperBurnBuff * 10;

        if (chosenEnemy1.isBlocking == false)
        {
            chosenEnemy1.TakeDamage(hyperBurnBuff);

            if (chosenEnemy1.tag == "Fire")
            {
                chosenEnemy1.isBurned = false;
            }
            else
            {
                chosenEnemy1.isBurned = true;
            }

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
```

3. This move deals twenty five damage to the targeted unit and deals fifteen damage back onto the user.

```
//p1 move 3
1 reference
public void PU1FireMove3Btn()
{
    playerMove3CD = 3;

    if (playerUnit1.isStunned == false)
    {
        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {
            chosenEnemy1.TakeDamage(25);

            if (playerUnit1.isBlocking == false)
            {
                playerUnit1.TakeDamage(15);
            }

            if (chosenEnemy1.tag == "Fire")
            {
                chosenEnemy1.isBurned = false;
            }
            else
            {
                chosenEnemy1.isBurned = true;
            }

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }
            else if (playerUnit1.currentHp <= 0)
            {
                playerUnit1.currentHp = 0;
                playerUnit1.isDead = true;
                playerUnit1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
            playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);
```

4. The final move for the Infernos unit increases the damage of your passive by two.

```
//p1 move 4
1 reference
public void PU1FireMove4Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove4CD = 1000;

        playerUnit1.passiveBuff = true;

        dialogueText.text = "Player's " + playerUnit1.unitName + " used their buff increasing burn damage by 2";

    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }

}
```

5. The passive of the Infernos unit is leaving a burn effect on the targeted units that deals two damage at the end of each turn. You can see this effect being applied in the previous scripts when the unit attacks.

```
//fire passive
1 reference
IEnumerator IsBurned()
{

    if (playerUnit1.isBurned == true && playerUnit1.isDead == false)
    {
        if(playerUnit1.isBlocking == false)
        {
            yield return new WaitForSeconds(3f);

            playerUnit1.TakeDamage(2);
            int num = 2;

            if (enemyUnit1.tag == "Fire" && enemyUnit1.passiveBuff == true)
            {
                playerUnit1.TakeDamage(2);
                num = 4;
            }
            if (enemyUnit2.tag == "Fire" && enemyUnit2.passiveBuff == true)
            {
                playerUnit1.TakeDamage(2);
                num = 4;
            }
            if (enemyUnit3.tag == "Fire" && enemyUnit3.passiveBuff == true)
            {
                playerUnit1.TakeDamage(2);
                num = 4;
            }

            if(playerUnit1.currentHp <= 0)
            {
                playerUnit1.currentHp = 0;
                playerUnit1.isDead = true;
            }

            playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);

            dialogueText.text = "Player's " + playerUnit1.unitName + " took "+num+" burn damage";

            if(playerUnit1.isDead == true)
            {
                yield return new WaitForSeconds(3f);
                dialogueText.text = "The player's " + playerUnit1.unitName + " was killed";
            }
        }
        else if (playerUnit1.isBlocking == true)
        {
            dialogueText.text = "Player's " + playerUnit1.unitName + " blocked the burn damage";
        }

    }
```

2.3.8.6. Aquaielle

The following scripts are the moves available to the Aquaielle unit along with the unit's passive.

1. This unit uses the same basic attack script as the other units
2. This move deals fifteen damage to the targeted unit and five damage to all other enemy units.

```
//p1 move 2
1 reference
public void PU1WaterMove2Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove2CD = 4;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (enemyUnit1.isBlocking == false || enemyUnit2.isBlocking == false || enemyUnit3.isBlocking == false)
        {
            if (chosenEnemy1.isBlocking == false)
            {
                chosenEnemy1.TakeDamage(10);
            }
            if (enemyUnit1.isBlocking == false)
            {
                enemyUnit1.TakeDamage(5);
            }
            if (enemyUnit2.isBlocking == false)
            {
                enemyUnit2.TakeDamage(5);
            }
            if (enemyUnit3.isBlocking == false)
            {
                enemyUnit3.TakeDamage(5);
            }

            if (enemyUnit1.currentHp <= 0 && enemyUnit1.isDead == false)
            {
                enemyUnit1.currentHp = 0;
                enemyUnit1.isDead = true;
                enemyUnit1.justDied = true;
            }
            if (enemyUnit2.currentHp <= 0 && enemyUnit2.isDead == false)
            {
                enemyUnit2.currentHp = 0;
                enemyUnit2.isDead = true;
                enemyUnit2.justDied = true;
            }
            if (enemyUnit3.currentHp <= 0 && enemyUnit3.isDead == false)
            {
                enemyUnit3.currentHp = 0;
                enemyUnit3.isDead = true;
                enemyUnit3.justDied = true;
            }

            enemyHUD1.SetHP(enemyUnit1.currentHp, enemyUnit1);
            enemyHUD2.SetHP(enemyUnit2.currentHp, enemyUnit2);
            enemyHUD3.SetHP(enemyUnit3.currentHp, enemyUnit3);
```

3. This move heals a chosen ally unit by thirty health points.

```
//p1 move 3
1 reference
public void PU1WaterMove3Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove3CD = 5;

        if (chosenPlayer1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && playerUnit1.isDead == false)
                {
                    chosenPlayer1 = playerUnit1;
                    chosenPlayerHUD1 = playerHUD1;
                    i = 101;
                }
                if (num == 1 && playerUnit2.isDead == false)
                {
                    chosenPlayer1 = playerUnit2;
                    chosenPlayerHUD1 = playerHUD2;
                    i = 101;
                }
                if (num == 2 && playerUnit3.isDead == false)
                {
                    chosenPlayer1 = playerUnit3;
                    chosenPlayerHUD1 = playerHUD3;
                    i = 101;
                }
            }
        }

        chosenPlayer1.HealDamage(30);

        if (chosenPlayer1.currentHp > chosenPlayer1.maxHp)
        {
            chosenPlayer1.currentHp = chosenPlayer1.maxHp;
        }
```

4. The final move for this unit increases the healing from their passive by two points.

```
//p1 move 4
1 reference
public void PU1WaterMove4Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove4CD = 1000;

        playerUnit1.passiveBuff = true;

        dialogueText.text = "Player's " + playerUnit1.unitName + " used their buff increasing healing rain heal by 2";
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }


}
```

5. The passive for this unit is that is heals all its team’s units by two health at the end of each turn.

```
//water passive
1 reference
IEnumerator HealingRain()
{

    if ((playerUnit1.tag == "Water" && playerUnit1.isDead == false) || (playerUnit2.tag == "Water" && playerUnit2.isDead == false) || (playerUnit3.tag == "Water" && playerUnit3.isDead == false))
    {
        yield return new WaitForSeconds(3f);
        int num = 2;

        if (playerUnit1.isDead == false)
        {
            playerUnit1.currentHp = playerUnit1.currentHp + 2;


            if (playerUnit1.tag == "Water" && playerUnit1.passiveBuff == true)
            {
                playerUnit1.currentHp = playerUnit1.currentHp + 2;
                num = 4;
            }
            if (playerUnit2.tag == "Water" && playerUnit2.passiveBuff == true)
            {
                playerUnit1.currentHp = playerUnit1.currentHp + 2;
                num = 4;
            }
            if (playerUnit3.tag == "Water" && playerUnit3.passiveBuff == true)
            {
                playerUnit1.currentHp = playerUnit1.currentHp + 2;
                num = 4;
            }

            if (playerUnit1.currentHp > playerUnit1.maxHp)
            {
                playerUnit1.currentHp = playerUnit1.maxHp;
            }
            playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);
        }
```

2.3.8.7. Rafflesia

The following scripts are the moves available to the Rafflesia unit along with the unit's passive.

1. This unit uses the same basic attack script as the other units.
2. This move deals fifteen damage to an enemy unit and heals the user ten health.

```
//p1 move 2
1 reference
public void PU1GrassMove2Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove2CD = 4;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {
            chosenEnemy1.TakeDamage(15);
            playerUnit1.HealDamage(10);

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }

            if (playerUnit1.currentHp > playerUnit1.maxHp)
            {
                playerUnit1.currentHp = playerUnit1.maxHp;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
            playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);
```

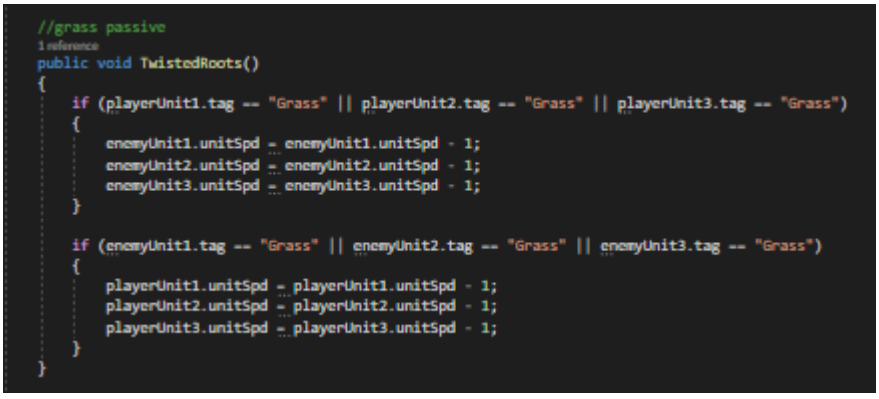
3. This move heals the user forty health.

```
public void PU1GrassMove3Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove3CD = 5;

        playerUnit1.HealDamage(40);

        if (playerUnit1.currentHp > playerUnit1.maxHp)
        {
            playerUnit1.currentHp = playerUnit1.maxHp;
        }

        playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);

        dialogueText.text = "Player's " + playerUnit1.unitName + " used Photosynthesis healing 40 health";
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }


}
```

4. The final move for this unit increases the passive enemy slow by two.

```
//p1 move 4
public void PU1GrassMove4Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove4CD = 1000;

        playerUnit1.passiveBuff = true;

        enemyUnit1.unitSpd = enemyUnit1.unitSpd - 2;
        enemyUnit2.unitSpd = enemyUnit2.unitSpd - 2;
        enemyUnit3.unitSpd = enemyUnit3.unitSpd - 2;

        dialogueText.text = "Player's " + playerUnit1.unitName + " used their buff increasing twisted roots slow by 2";
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }


}
```

5. The passive for this unit reduces the speed of all enemy units by one.



```
//grass passive
public void TwistedRoots()
{
    if (playerUnit1.tag == "Grass" || playerUnit2.tag == "Grass" || playerUnit3.tag == "Grass")
    {
        enemyUnit1.unitSpd = enemyUnit1.unitSpd - 1;
        enemyUnit2.unitSpd = enemyUnit2.unitSpd - 1;
        enemyUnit3.unitSpd = enemyUnit3.unitSpd - 1;
    }

    if (enemyUnit1.tag == "Grass" || enemyUnit2.tag == "Grass" || enemyUnit3.tag == "Grass")
    {
        playerUnit1.unitSpd = playerUnit1.unitSpd - 1;
        playerUnit2.unitSpd = playerUnit2.unitSpd - 1;
        playerUnit3.unitSpd = playerUnit3.unitSpd - 1;
    }
}
```

2.3.8.8. Ganturf

The following scripts are the moves available to the Ganturf unit along with the unit's passive.

1. This unit uses the same basic attack script as the other units.
2. This move deals between ten to thirty damage to an enemy.

```
//p1 move 2
1 reference
public void PU1EarthMove2Btn()
{

    playerUnit1.isBlocking = false;
    playerUnit2.isBlocking = false;
    playerUnit3.isBlocking = false;

    if (playerUnit1.isStunned == false)
    {
        playerMove2CD = 4;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {
            int num = Random.Range(10, 31);

            chosenEnemy1.TakeDamage(num);

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
```

3. This move applies a shield to a friendly unit so they can block the next attack directed at them.

```
//p1 move 3
1 reference
public void PU1EarthMove3Btn()
{
    playerUnit1.isBlocking = false;
    playerUnit2.isBlocking = false;
    playerUnit3.isBlocking = false;

    if (playerUnit1.isStunned == false)
    {
        playerMove3CD = 4;

        if (chosenPlayer1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && playerUnit1.isDead == false)
                {
                    chosenPlayer1 = playerUnit1;
                    i = 101;
                }
                if (num == 1 && playerUnit2.isDead == false)
                {
                    chosenPlayer1 = playerUnit2;
                    i = 101;
                }
                if (num == 2 && playerUnit3.isDead == false)
                {
                    chosenPlayer1 = playerUnit3;
                    i = 101;
                }
            }
        }

        chosenPlayer1.isBlocking = true;

        dialogueText.text = "Player's " + playerUnit1.unitName + " used Rock Shield on the player's " + chosenPlayer1.unitName;
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }

}
```

4. The final move for this unit increases the max health of all your ally units by ten.

```
//p1 move 4
1 reference
public void PU1EarthMove4Btn()
{
    playerUnit1.isBlocking = false;
    playerUnit2.isBlocking = false;
    playerUnit3.isBlocking = false;

    if (playerUnit1.isStunned == false)
    {
        playerMove4CD = 1000;

        playerUnit1.passiveBuff = true;

        playerUnit1.maxHp = playerUnit1.maxHp + 10;
        playerUnit1.currentHp = playerUnit1.currentHp + 10;
        playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);

        playerUnit2.maxHp = playerUnit2.maxHp + 10;
        playerUnit2.currentHp = playerUnit2.currentHp + 10;
        playerHUD2.SetHP(playerUnit2.currentHp, playerUnit2);

        playerUnit3.maxHp = playerUnit3.maxHp + 10;
        playerUnit3.currentHp = playerUnit3.currentHp + 10;
        playerHUD3.SetHP(playerUnit3.currentHp, playerUnit3);

        dialogueText.text = "Player's " + playerUnit1.unitName + " used their buff increasing rock shield health by 10";
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }


}
```

5. The passive for this unit increases the max health of all units by ten at the start of the battle.

```
//earth passive
1 reference
public void EarthHPIncrease()
{
    if(playerUnit1.tag == "Earth" || playerUnit2.tag == "Earth" || playerUnit3.tag == "Earth")
    {
        playerUnit1.maxHp = playerUnit1.maxHp + 10;
        playerUnit1.currentHp = playerUnit1.currentHp + 10;
        playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);

        playerUnit2.maxHp = playerUnit2.maxHp + 10;
        playerUnit2.currentHp = playerUnit2.currentHp + 10;
        playerHUD2.SetHP(playerUnit2.currentHp, playerUnit2);

        playerUnit3.maxHp = playerUnit3.maxHp + 10;
        playerUnit3.currentHp = playerUnit3.currentHp + 10;
        playerHUD3.SetHP(playerUnit3.currentHp, playerUnit3);
    }

    if(enemyUnit1.tag == "Earth" || enemyUnit2.tag == "Earth" || enemyUnit3.tag == "Earth")
    {
        enemyUnit1.maxHp = enemyUnit1.maxHp + 10;
        enemyUnit1.currentHp = enemyUnit1.currentHp + 10;
        enemyHUD1.SetHP(enemyUnit1.currentHp, enemyUnit1);

        enemyUnit2.maxHp = enemyUnit2.maxHp + 10;
        enemyUnit2.currentHp = enemyUnit2.currentHp + 10;
        enemyHUD2.SetHP(enemyUnit2.currentHp, enemyUnit2);

        enemyUnit3.maxHp = enemyUnit3.maxHp + 10;
        enemyUnit3.currentHp = enemyUnit3.currentHp + 10;
        enemyHUD3.SetHP(enemyUnit3.currentHp, enemyUnit3);
    }

}
```

2.3.8.9. Tempest

The following scripts are the moves available to the Tempest unit along with the unit's passive.

1. This unit uses the same basic attack script as the other units.
2. This move deals five damage and increases by five damage every time it is used.

```
//p1 move 2
1 reference
public void PU1WindMove2Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove2CD = 2;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                else if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                else if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {

            chosenEnemy1.TakeDamage(whirlingTornadoBuff);

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
```

3. This move deals twenty damage to the enemy unit and has a fifty percent chance to deal twenty recoil damage to the user.

```
//p1 move 3
1 reference
public void PU1WindMove3Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove3CD = 3;

        bool selfDamage = false;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {
            chosenEnemy1.TakeDamage(20);

            int recoil = Random.Range(0, 2);

            if (recoil == 0)
            {
                selfDamage = true;
            }

            if (playerUnit1.isBlocking == false && selfDamage == true)
            {
                playerUnit1.TakeDamage(20);
            }

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }
            else if (playerUnit1.currentHp <= 0)
            {
                playerUnit1.currentHp = 0;
                playerUnit1.isDead = true;
                playerUnit1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
            playerHUD1.SetHP(playerUnit1.currentHp, playerUnit1);
```

4. The final move for this unit increases the passive speed increase of all ally units by two.

```
//p1 move 4
1 reference
public void PU1WindMove4Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove4CD = 1000;

        playerUnit1.passiveBuff = true;

        playerUnit1.unitSpd = playerUnit1.unitSpd + 2;
        playerUnit2.unitSpd = playerUnit2.unitSpd + 2;
        playerUnit3.unitSpd = playerUnit3.unitSpd + 2;

        dialogueText.text = "Player's " + playerUnit1.unitName + " used their buff increasing Hurricane Tailwind speed by 2";
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }


}
```

5. The passive for this unit increases the speed of all allies by one.

```
//wind passive
1 reference
public void HurricaneWinds()
{
    if (playerUnit1.tag == "Wind" || playerUnit2.tag == "Wind" || playerUnit3.tag == "Wind")
    {
        playerUnit1.unitSpd = playerUnit1.unitSpd + 1;
        playerUnit2.unitSpd = playerUnit2.unitSpd + 1;
        playerUnit3.unitSpd = playerUnit3.unitSpd + 1;
    }

    if (enemyUnit1.tag == "Wind" || enemyUnit2.tag == "Wind" || enemyUnit3.tag == "Wind")
    {
        enemyUnit1.unitSpd = enemyUnit1.unitSpd + 1;
        enemyUnit2.unitSpd = enemyUnit2.unitSpd + 1;
        enemyUnit3.unitSpd = enemyUnit3.unitSpd + 1;
    }
}
```

2.3.8.10. Voltage

The following scripts are the moves available to the Voltage unit along with the unit's passive.

1. This unit's basic attack differs from that of the other units. This unit's base attack damage increases by randomly selecting a fellow ally unit and adding their attack damage onto this unit's attack damage. This is also the passive for this unit.

```
//p1 move 1
1 reference
public void PU1ElectricMove1Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove1CD = 1;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy1 = enemyUnit1;
                    chosenEnemyHUD1 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && enemyUnit2.isDead == false)
                {
                    chosenEnemy1 = enemyUnit2;
                    chosenEnemyHUD1 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy1 = enemyUnit3;
                    chosenEnemyHUD1 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {
            int num = Random.Range(0, 2);

            int playerAtk = playerUnit1.unitAtk;

            if(num == 0)
            {
                playerAtk = playerUnit1.unitAtk + playerUnit2.unitAtk;
            }
            else if (num == 1)
            {
                playerAtk = playerUnit1.unitAtk + playerUnit3.unitAtk;
            }

            if(playerUnit1.passiveBuff == true)
            {
                playerAtk = playerUnit1.unitAtk + playerUnit2.unitAtk + playerUnit3.unitAtk;
            }

            chosenEnemy1.TakeDamage(playerAtk);

            if (chosenEnemy1.currentHp <= 0)
            {
                chosenEnemy1.currentHp = 0;
                chosenEnemy1.isDead = true;
                chosenEnemy1.justDied = true;
            }

            chosenEnemyHUD1.SetHP(chosenEnemy1.currentHp, chosenEnemy1);
```

2. This move stuns the targeted unit for one turn.

```
//p1 move 2
1 reference
public void PU1ElectricMove2Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove2CD = 4;

        if (chosenEnemy1.isDead == true)
        {
            for (int i = 0; i < 100; i++)
            {
                int num = Random.Range(0, 3);

                if (num == 0 && enemyUnit1.isDead == false)
                {
                    chosenEnemy3 = enemyUnit1;
                    chosenEnemyHUD3 = enemyHUD1;
                    i = 101;
                }
                if (num == 1 && playerUnit2.isDead == false)
                {
                    chosenEnemy3 = enemyUnit2;
                    chosenEnemyHUD3 = enemyHUD2;
                    i = 101;
                }
                if (num == 2 && enemyUnit3.isDead == false)
                {
                    chosenEnemy3 = enemyUnit3;
                    chosenEnemyHUD3 = enemyHUD3;
                    i = 101;
                }
            }
        }

        if (chosenEnemy1.isBlocking == false)
        {

            chosenEnemy1.isStunned = true;

            dialogueText.text = "Player's " + playerUnit1.unitName + " used Lightning Chamber on the enemy " + chosenEnemy1.unitName + " stunning them";
        }
        else if (chosenEnemy1.isBlocking == true)
        {
            dialogueText.text = "Enemy " + chosenEnemy1.unitName + " blocked the attack";
        }
    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }

}
```

3. This move deals ten damage to all other units.

```
//p1 move 3
1 reference
public void PU1ElectricMove3Btn()
{
    if (playerUnit1.isStunned == false)
    {
        playerMove3CD = 5;

        if(playerUnit2.isBlocking == false)
        {
            playerUnit2.TakeDamage(10);
        }
        if (playerUnit3.isBlocking == false)
        {
            playerUnit3.TakeDamage(10);
        }
        if (enemyUnit1.isBlocking == false)
        {
            enemyUnit1.TakeDamage(10);
        }
        if (enemyUnit2.isBlocking == false)
        {
            enemyUnit2.TakeDamage(10);
        }
        if (enemyUnit3.isBlocking == false)
        {
            enemyUnit3.TakeDamage(10);
        }

        if (playerUnit2.currentHp <= 0 && playerUnit2.isDead == false)
        {
            playerUnit2.currentHp = 0;
            playerUnit2.isDead = true;
            playerUnit2.justDied = true;
        }
        if (playerUnit3.currentHp <= 0 && playerUnit3.isDead == false)
        {
            playerUnit3.currentHp = 0;
            playerUnit3.isDead = true;
            playerUnit3.justDied = true;
        }
        if (enemyUnit1.currentHp <= 0 && enemyUnit1.isDead == false)
        {
            enemyUnit1.currentHp = 0;
            enemyUnit1.isDead = true;
            enemyUnit1.justDied = true;
        }
        if (enemyUnit2.currentHp <= 0 && enemyUnit2.isDead == false)
        {
            enemyUnit2.currentHp = 0;
            enemyUnit2.isDead = true;
            enemyUnit2.justDied = true;
        }
        if (enemyUnit3.currentHp <= 0 && enemyUnit3.isDead == false)
        {
            enemyUnit3.currentHp = 0;
            enemyUnit3.isDead = true;
            enemyUnit3.justDied = true;
        }

        playerHUD2.SetHP(playerUnit2.currentHp, playerUnit2);
        playerHUD3.SetHP(playerUnit3.currentHp, playerUnit3);
        enemyHUD1.SetHP(enemyUnit1.currentHp, enemyUnit1);
        enemyHUD2.SetHP(enemyUnit2.currentHp, enemyUnit2);
        enemyHUD3.SetHP(enemyUnit3.currentHp, enemyUnit3);
```

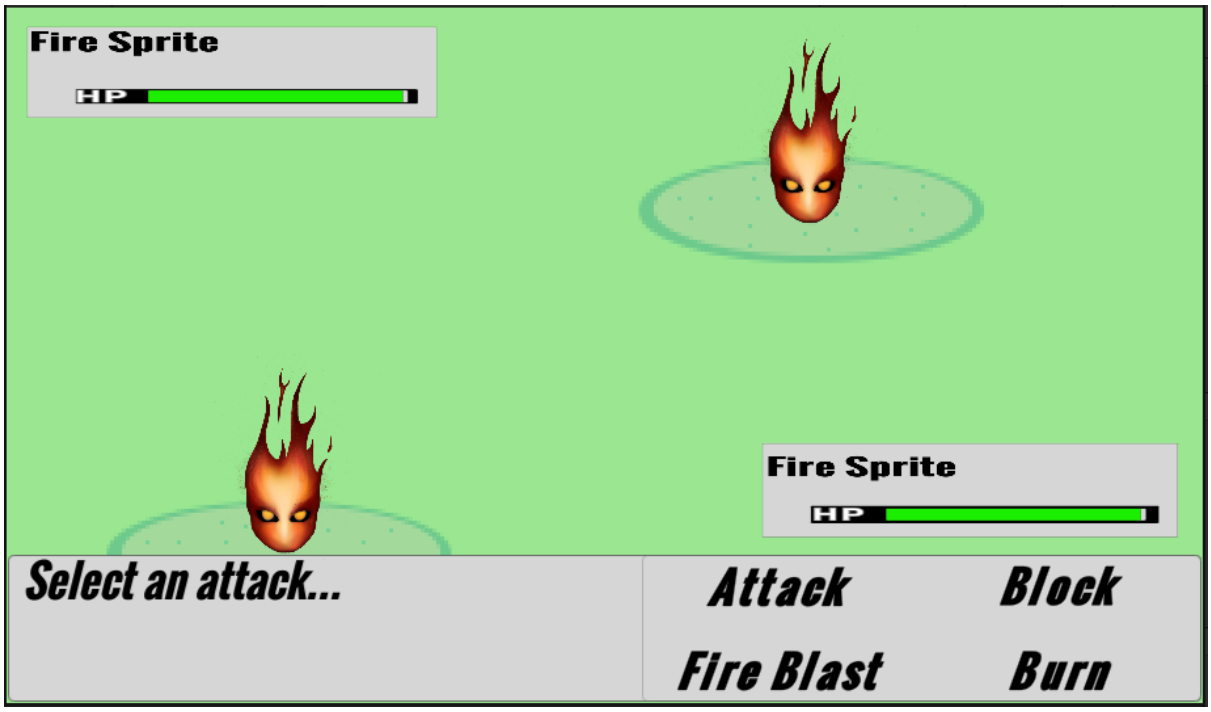
4. The final move for this unit increases the combined attack damage passive so that when this unit basic attacks it combines the attack damage of both ally units with it’s own.

```
//p1 move 4
1 reference
public void PU1ElectricMove4Btn()
{

    if (playerUnit1.isStunned == false)
    {
        playerMove4CD = 1000;

        playerUnit1.passiveBuff = true;

        dialogueText.text = "Player's " + playerUnit1.unitName + " used their buff increasing electrical surges combined allys used to 2";

    }
    else
    {
        dialogueText.text = "Player's " + playerUnit1.unitName + " is stunned";
        playerUnit1.isStunned = false;
    }

}
```

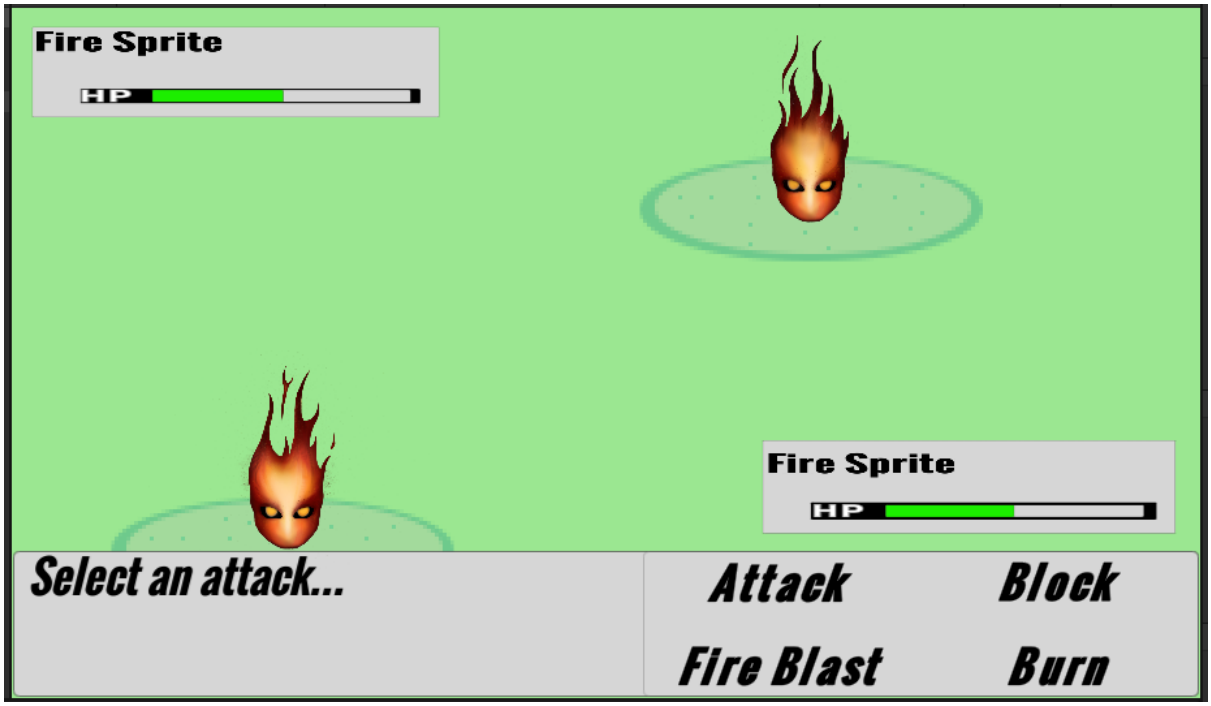
2.4. Graphical User Interface (GUI)

2.4.1. First Look

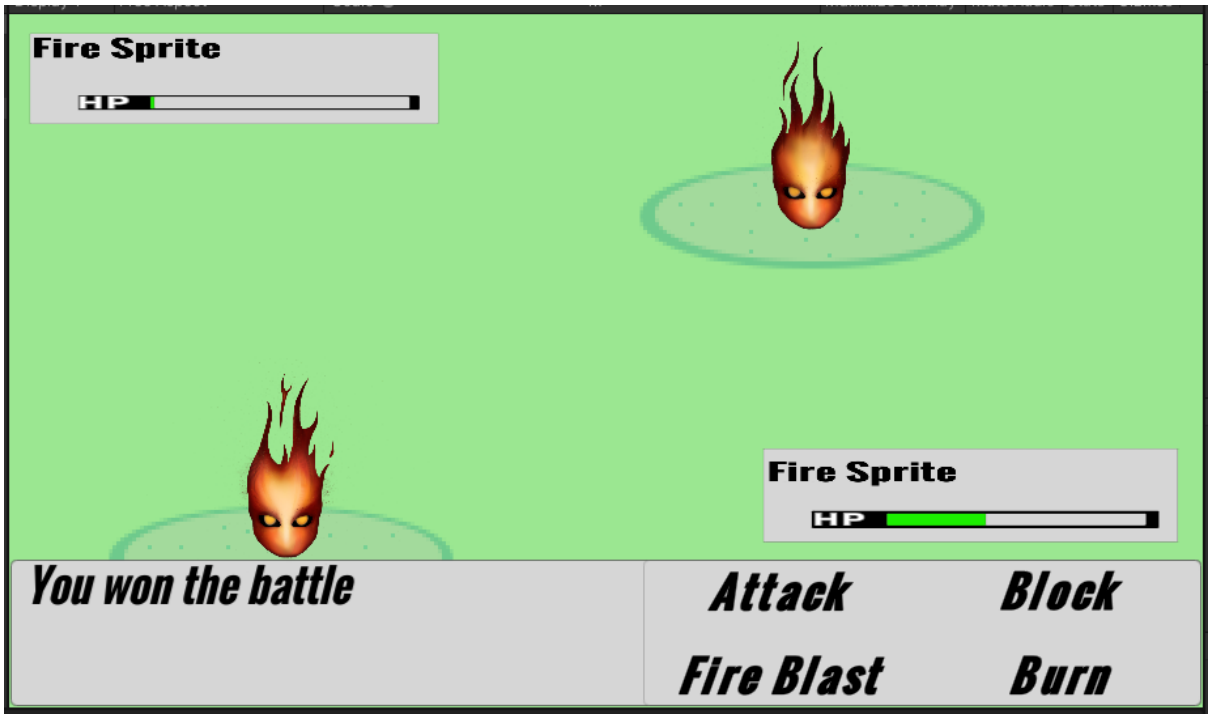
The following screenshots are from the midpoint and the first look at what the game was going to be.



Battle sequence has commenced and you are asked to select your first attack.



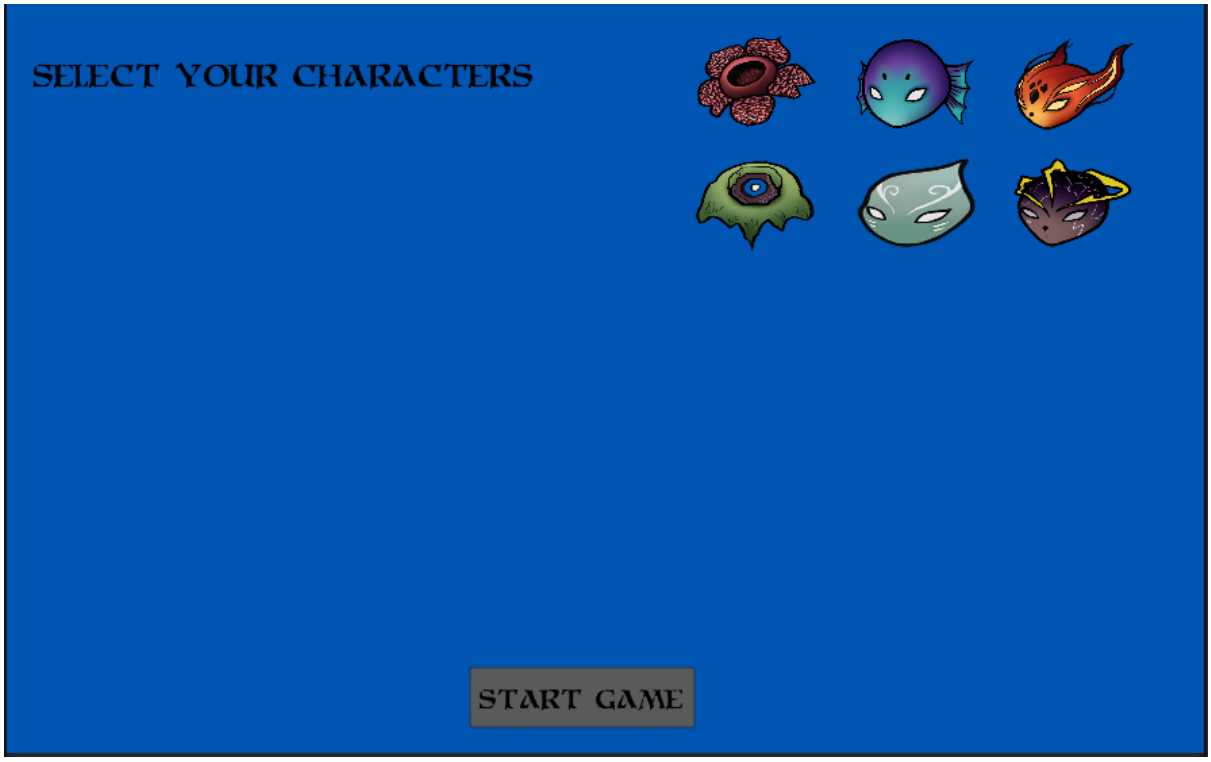
HP bar has changed as you go through the attack sequences.



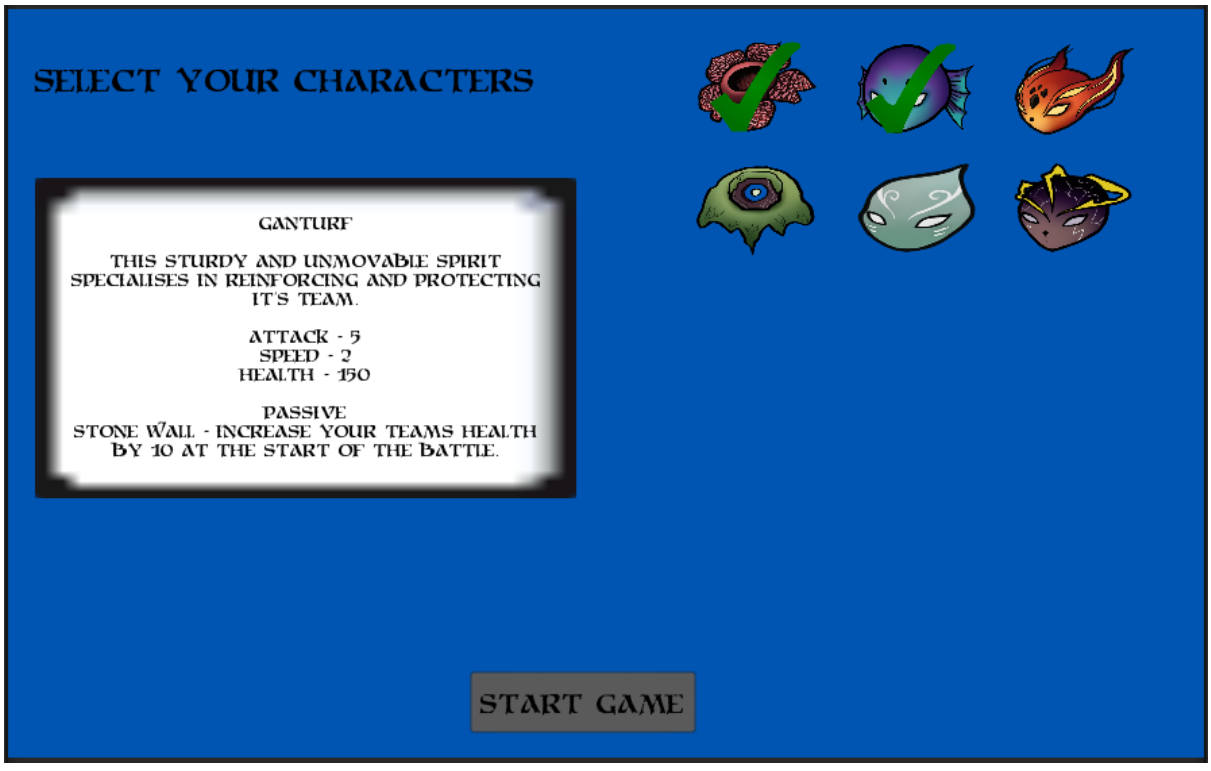
When HP reaches 0 winner is chosen.

2.4.2. Final Product

The following screenshots are the current state of the GUI.



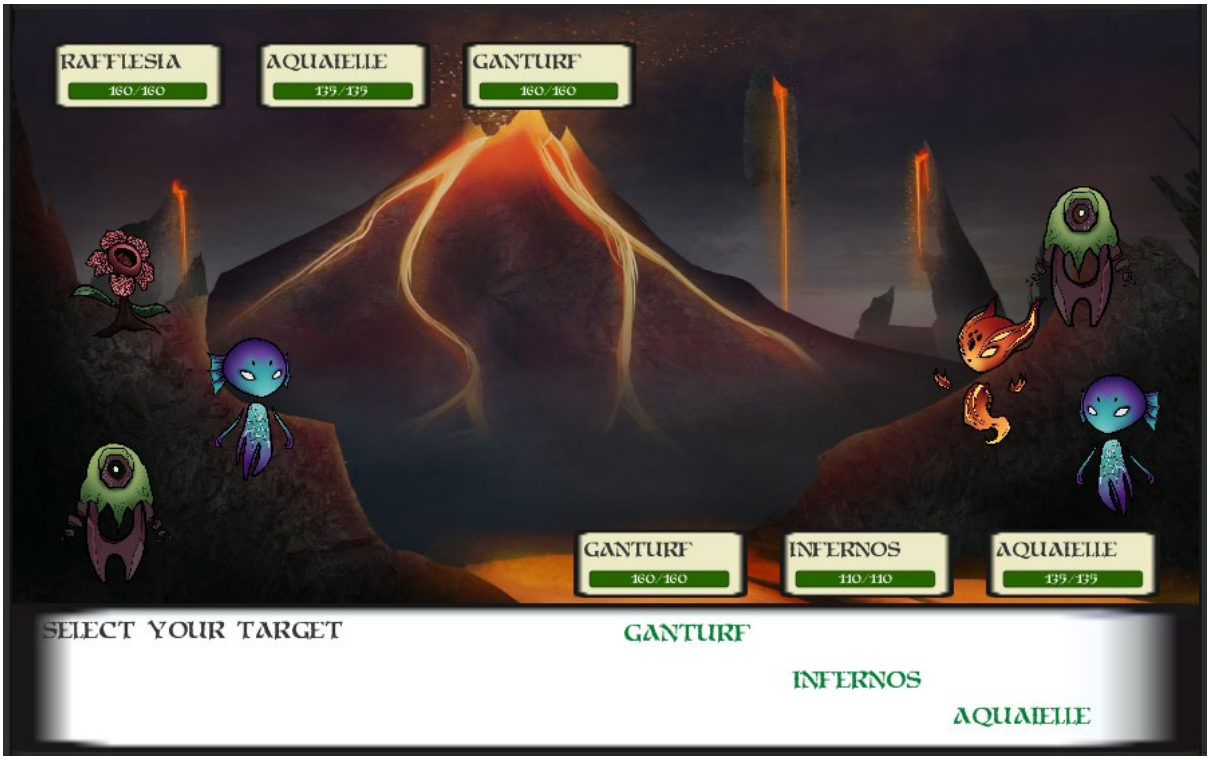
This is the character selection screen which is the first screen the player is shown, here the player can choose the three characters they wish to use.



An example of two selected characters and the information panel when hovering a specific character.

The beginning of the battle sequence, you can see all the selected units and their corresponding names and health. The dialogue bar is showing the available characters that can attack.

The dialogue bar is showing the list of moves for a specific character.



The dialogue bar is showing the optional characters your attack can target.

The game state after multiple turns.

Game state after all the units have died on a specific side.

2.5. Testing

2.5.1. Developer Testing

As I created the game, I tested it frequently. Every time I implemented a new game function, I made sure it worked properly before advancing onto the next stage. In some situations, the part I created may not have worked up to the standard I had originally intended so I just made sure that it didn't impact the overall flow of the game and I left notes to remind myself to correct it later in the project timeline.

2.5.2. Pre-Release User Testing

Prior to uploading my game, I asked a user to test it out. They found multiple bugs with the system after trying to complete the game. From their feedback I was able to improve the system dramatically before uploading it live online. The issues they found with the game are as follows:

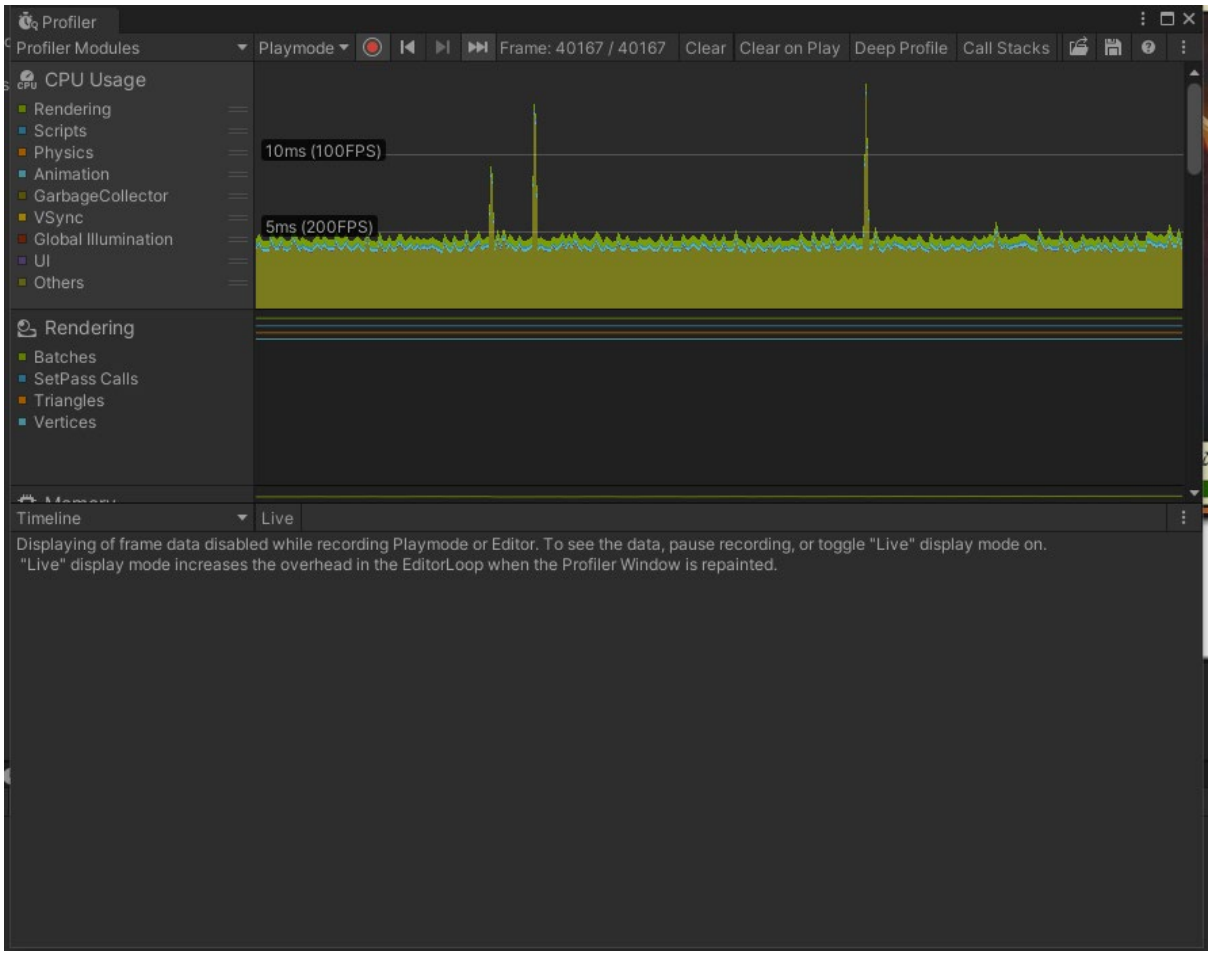
- The win dialogue showing up but then the game continuing.
- The game would stop working after a unit dies.
- Some of the abilities did not match their descriptions.
- Sometimes after a unit died the game would continue to run their script but the dialogue and system wouldn't update, so the timer that exists so the user can read the information on time was extended (five seconds to ten seconds).

2.5.3. Live User Testing

Multiple users gave their feedback after I posted the game online. They pointed out bugs and important things they would like added to the game to improve the overall quality and functionality. I have taken note of these comments made by users and added them into my games description to entice more users to give their feedback so I can improve the game in the future. I can also use the information I have gathered in future games I build if I don't continue this game after college.

2.6. Evaluation

I used the inbuilt unity profiler to analyse that fps of my game. This showed me I had an almost constant 200fps with a few small drops to 60fps.These drops are not an issue as 60fps is the standard average fps for video games.



I also got feedback from users after I posted the game live. They enjoyed the idle animation for the sprites and thought it looked nice. They also enjoyed the game and said it did exactly what it set out to do, it was a battle simulator vs AI. The users that played the game varied in age but none of them found it difficult to grasp, and some players lost to the AI so it wasn't without a small amount of difficulty.

3.0 Conclusions

Advantages:

- Improve my knowledge of game development.
- Improve my overall coding capabilities.
- Increase my project portfolio with a what I hope to be a very impressive project.

Disadvantages:

- Time limit is a big disadvantage.
- My lack of knowledge of unity is a disadvantage.
- I have not studied game dev as a module during my 4 years in NCI so all this work is my own with no college background.

Strengths:

- I have been messing around with game creation since my first year of college.
- I have contacts who have created games and they are guiding me on what tutorials to follow and how best to approach a project.
- I have a passion for playing games and this comes across when creating them. My work drive is higher for this project than any other.

Limitations:

- I have limited knowledge of unity and I am learning as I go so this slows and limits the creation process.
- The time limit prevents may prevent me from finishing what I intend to finish as I can only guess how long it will take due to lack of experience.

4.0 Further Development or Research

I would like this game to become a fully-fledged RPG but that could take a few years to put together as a solo developer, if I had a small team and more time, I believe I could reach this goal.

On the other hand, I could also implement multiplayer into this game and have the core game play being the battle simulator. In its current state it is very easy for me to keep adding new content to the game.

5.0 References

Hall, J., 2020. *GitHub - joshuahall98/Adventure-Game: Game*. [online] GitHub. Available at: <https://github.com/joshuahall98/Adventure-Game> [Accessed 2 August 2021].

Brackeys, 2019. *Before you continue to YouTube*. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=_1pz_ohupPs> [Accessed 2 August 2021].

6.0 Appendices

6.1. Project Plan

- ### Objectives

The end goal for this project is to have a fully fleshed role-playing game where you follow the story of a young boy/girl and their adventure across a world searching for creatures. Using these creatures, they will battle other adventurers (AI) to reach the final battle and save their world. I do not expect to have built an entire game come the end of my 4th year, so I have set myself two possible short-term goals. One being the beginning of the game with some select options, movement, and dialogue with one or two battle sequences, this will be a good start to reach the end goal. The other option is to extract the battle sequence and concentrate completely on that and make it its own game. I will make my decision as I go along with this project when I find out which objective is most realistic to achieve come the end of this year. I will be using Unity and a yet to be chosen app for creating sprites.

- ### Background

As of right now the only game creating experience I have is on scratch and choose your path-based games I made using JavaScript. I also made a small reaction-based game within a phone app. These games I built in the past differ completely from the task I will be undertaking this year; this is due to the fact I will be using unity. As of right now my experience with unity is limited, I have followed a few tutorials and made some small games with very basic objectives. If I wish to undertake the creation of the game I described above, I will need to continue learning how to use Unity alongside my other college modules.

As well as creating the game code I will also have to create the sprites for my game. I am currently looking into the process for creating sprites and it seems 2D is the most realistic option with my limited art skills. The only art experience I have is from the junior cert cycle.

- ### Technical Approach

Up until semester two I intend to spend most of my time learning how to use unity and create 2D sprites, I will use You Tube tutorials and online forums to help give me the knowledge to create my own game. In semester two I hope to use less tutorials and more of my own knowledge to build my game.

- ### Special Resources Required

Photoshop was recommended for sprite creation; I am currently looking for a cheap alternative.

- Project Plan

TASK	ASSIGNED TO	PROGRESS	START	END
Learning Unity and 2D animation			11/8/20	31/12/20
Basic Unity Interface tutorials				
Game Creation Tutorials				
2D Sprite Creation Tutorials				
Creating Sprites			1/1/21	31/1/2021
Creating Base Sprite				
Sprite Animation				
Building Game			1/2/21	31/3/21
Code the Game				
Testing and Bug Fixing			1/4/21	30/4/21
Get Fellow Students to Test Game				
Find and Fix Bugs				

- Technical Details

I used Unity build my game. The scripts I wrote were in visual studio using C#. I uploaded the game as a WEBGL onto itch.io. I used gimp to edit some of the sprites.

- Evaluation

I hope to implement an AI to evaluate my system to make sure everything works. Failing that I will have the game tested by multiple people to help find any bugs in the game.

6.2. Reflective Journals

Monthly Reflective Journal October

We have only just submitted our project ideas and as of yet have received no feedback, whether I can go ahead with my project is still unknown. Other than that, I have been

learning how to use unity in my spare time in preparation for my project idea.

Monthly Reflective Journal November

I have finally chosen the application to create my assets for unity, I will be using Gimp. I have learned how to animate drawings but after trying to draw the assets I have found I lack the artistic flare I desire. I am going to seek outside help for the art of my project so it can reach my expectations. I have also started coding the game itself, as of right now I have begun creating a menu screen.

Monthly Reflective Journal December

I have created a simple battle simulator in preparation for my actual game. I intend to overhaul this but for now it is a good start in the right direction. I also learned how to animate sprites I have implemented a simple animation within my game.

Monthly Reflective Journal January-February

I was unhappy with how I had created the prototype for my game. I learned a lot from my first attempt and figured out I placed a wall in front of myself. I decided to start again from scratch to improve the game I had created, and I got back to where I was before Christmas and am now pushing ahead with new updates to the game.

Monthly Reflective Journal March

Ran into a road bump where my unity project stopped working and could not roll it back to a previous version. Had to start the project all over again, I can re-use the code though.

Monthly Reflective Journal April

Fixed the issue with my Unity application not working on the 04/05/2021, with the help of a fellow student we figured out the issue was not with Unity but instead an error with my PC

which I had to go fix. I have not been able to work on my project since the error arose, so no progress has been made.

Monthly Reflective Journal May

Following the fix I made to my PC, I was finally able to progress with my project. I returned to the work I had done previously and tried to find some new tutorials so I could approach my project differently but found nothing was helping me with what I wanted to do. I was still stuck with a dysfunctional version of my mid-point upload.

Monthly Reflective Journal June

After working on a weeklong game jam with another student he taught me the best techniques when it came to creating games. With his assistance I learned how to build a fully functioning game from the ground up. With my new knowledge I was able to approach my final year project properly and shortly after we finished our game jam game, I figured out the algorithm I needed to create the unit movement priority. This allowed for me to move forward at high speed with the rest of the project, building the entire battle system.

Monthly Reflective Journal July

Progress continued at a high pace throughout this month, and I was able to create all the unit's stats and attacks along with the AI. I also started filling in my technical report.