

Ultimon Hero

Fatmir Gusani

BACHELOR OF ENGINEERING (HONOURS) IN SOFTWARE AND ELECTRONIC ENGINEERING

Galway-Mayo Institute of Technology

2019/2020

Poster



Game Play



**Declaration**

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

FATMIR GUSANI

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgements**

Use this section to acknowledge anyone, if you wish to, who might have helped during your project.

Mairtin O Conghaile

Tommy Kearns

**Table of Contents**

[1 Summary 9](#_Toc40706834)

[2 Introduction 10](#_Toc40706835)

[3 Intro to Game 11](#_Toc40706836)

[4 Level System 13](#_Toc40706837)

[4.1 Display Level 15](#_Toc40706838)

[5 Health System 18](#_Toc40706839)

[5.1 Hero Health 18](#_Toc40706840)

[5.2 Enemy Health 21](#_Toc40706841)

[5.3 Hero Health Bar 25](#_Toc40706842)

[6 Stats 35](#_Toc40706843)

[6.1 Hero Stats 35](#_Toc40706844)

[6.2 Enemy Stats 39](#_Toc40706845)

[7 Spawn System 41](#_Toc40706846)

[7.1 Hero Spawn 41](#_Toc40706847)

[7.2 Enemy Spawn 43](#_Toc40706848)

[8 Button Functions 45](#_Toc40706849)

[8.1 Button Delay 45](#_Toc40706850)

[8.1 Button New Hero 47](#_Toc40706851)

[9 Secret Hero 48](#_Toc40706852)

[10 Problem Solving 49](#_Toc40706853)

[11 Conclusion 50](#_Toc40706854)

[12 References 51](#_Toc40706855)

[13 Code 52](#_Toc40706856)

[13.1 Fanir\_EneHealth 52](#_Toc40706857)

[13.2 Fanir\_Health 54](#_Toc40706858)

[13.3 Fanir\_HealthBar 56](#_Toc40706859)

[13.4 Lagoon\_EneHealth 63](#_Toc40706860)

[13.5 Lagoon\_Health 65](#_Toc40706861)

[13.6 Lagoon\_HealthBar 67](#_Toc40706862)

[13.7 Panboo\_EneHealth 74](#_Toc40706863)

[13.8 Panboo\_Health 76](#_Toc40706864)

[13.9 Panboo\_HealthBar 78](#_Toc40706865)

[13.10 Secret\_EneHealth 85](#_Toc40706866)

[13.11 Secret\_Health 87](#_Toc40706867)

[13.12 Secret\_HealthBar 89](#_Toc40706868)

[13.13 DisplayLevel 96](#_Toc40706869)

[13.14 LevelSystem 98](#_Toc40706870)

[13.15 Fanir\_EnemySpawner 100](#_Toc40706871)

[13.16 FanirSpawner 101](#_Toc40706872)

[13.17 Lagoon\_EnemySpawner 102](#_Toc40706873)

[13.18 LagoonSpawner 103](#_Toc40706874)

[13.19 Panboo\_EnemySpawner 104](#_Toc40706875)

[13.20 PanbooSpawner 105](#_Toc40706876)

[13.21 SecretEnemySpawner 106](#_Toc40706877)

[13.22 SecretSpawner 107](#_Toc40706878)

[13.23 EnemyLevelStats 108](#_Toc40706879)

[13.24 HeroLevelStats 109](#_Toc40706880)

[13.25 ButtonDelay 111](#_Toc40706881)

[13.26 ButtonNewHero 113](#_Toc40706882)

[13.27 MainMen 114](#_Toc40706883)

# Summary

My project is a recreation of the batting part in the Pokémon Games. Anyone who played any of the games understands the excitement of batting with your Pokémon. My goal for Ultimon Hero is for the player to enjoy the game.

The game allows the player to pick a starter hero from three characters and have a one-versus-one battle with an enemy hero/character. If the player wins the battle, their hero levels up. Once a certain level is reached the final boss battle will be triggered, the player will be battling their hero with a new look and better stats. If the player beats the final boss, they can play as the boss. If the player beats the game with all three starter heroes, they unlock a new secret hero.

A health system, battle system, level system, stats system & spawn system are implemented into the game.

A new Pokémon game was released on 15th November 2019. I thought it would be awesome to create my version of the game.

The main technologies used are Unity, I coded the game in C# as it worked well with Unity’s API. No hardware is used as the project is all software-based.

Creating the game was a success as all the features for it worked, the only thing that didn't work was the animations.

The main conclusion is that I had fun making the game, some parts were stressful but overall it was good.

# Introduction

My motivation for the project is that I always wanted to create my own game, I was a gamer since I was a little boy and I’m still a gamer till this day. I have been a fan of Pokémon for a decade now. When the new Pokémon game come out, I was inspired to create my version of the game. Since it was my game, I can add/remove any parts of the game that I liked or didn’t like, design and add features that the original Pokémon games didn’t have, that’s the beauty of creating your own game.

# Intro to Game

When you start the game, the player is presented with an introduction scene, a play and quit button is used to navigate on what the player wants to do.

The “PLAY” and “QUIT” are buttons that trigger an Onclick() method.

The “QUIT” button will trigger a method called QuitGame() in the MainMenu script.

The code will display a message to the console and end the application.

public void QuitGame ()

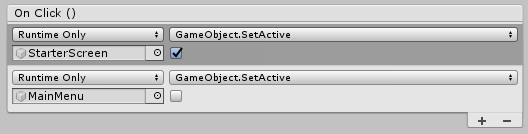
{

Debug.Log("Quiting");

Application.Quit();

}

The “PLAY” button will trigger one of the Unity APIs that disables the object and enables the starter select screen object. The checkbox determines the true or false value.



Once the player is ready to play the game, they will be brought to a starter select screen [1].



This screen is where the player picks their starting hero. The names of the heroes are buttons that open a new scene where the selected hero is battling. The button uses an Onclick() method that calls a function inside the MainMenu[7] script.

using UnityEngine.SceneManagement;

public void FanirStarter()

{

SceneManager.LoadScene("FanirScene");

}

public void LagoonStarer()

{

SceneManager.LoadScene("LagoonScene");

}

public void PanbooStarter()

{

SceneManager.LoadScene("PanbooScene");

}

The SceneManager is a class in Unity’s API and the LoadScene that scene with that name in the build setting. To put the scene in the build setting, you must go to File->Build Settings [, drag the scene in the build setting to put it into it so that the LoadScene can find and display the scene. All the scenes will be handled by the MainMenu[7] script.

# Level System

The Level system is one of the most important features in my game. The game uses the level of the hero to make the player’s hero stronger as they level up, the final battle is triggered once a certain level is reached, new attacks are learned when the player’s hero is on the right level. The player’s hero and the enemy will also be gain stats boosts as they level up.

The enemy hero will always be the same level as the player’s hero, I designed it like this because it will always make the enemy and the player on the same level. No, overpower or underpower heroes. It will be a fair fight.

To start this, I created a script called LevelSystem[2].

This script will be doing the maths for our level system.

I added all the event handlers/triggers to the game, this is important as it will the responsible for detecting a chance on the experience and level when it occurs.

// Start is called before the first frame update

public event EventHandler OnExperienceChange;

public event EventHandler OnLevelChange;

I then add some public/private variables.

* Level variable will be used to do the maths.
* Experience variable will be used to determine how much experience the hero has.
* ExperienceNextLevel variable is the value of experience needed to gain a level.
* KeepExp is the value of experience the hero starts with.
* KeepLevel is the value of the level the hero starts with.

public int Level = KeepLevel;

public int Experience = KeepExp;

private int ExperienceNextLevel = 100;

public static int KeepExp = 0;

public static int KeepLevel = 1;

AddExperience() is a function that takes an integer and will be used to handle the adding of experience. The integer value is gained into Experience. If the Experience is greater than ExperienceNextLevel value, one level is gained, ExperienceNextLevel is subtracted from Experience and OnLevelChange handler is triggered. KeepLevel gets the value from Level.

After the OnExperienceChange handler is triggered. KeepExp gets the value from Experience.

public void AddExperience(int amount)

{

Experience += amount;

if (Experience >= ExperienceNextLevel)

{

Level++;

Experience -= ExperienceNextLevel;

if (OnLevelChange != null) OnLevelChange(this, EventArgs.Empty);

//KeepLevel = Level;

KeepLevel = Level;

}

if (OnExperienceChange != null) OnExperienceChange(this, EventArgs.Empty);

KeepExp = Experience;

}

GetLevelNumber() function just returns the level of the hero.

public int GetLevelNumber()

{

return Level;

}

GetExperienceNormalied() function return the amount of experience the hero has.

public float GetExperienceNormalized()

{

return (float)Experience / ExperienceNextLevel;

}

ReturnExpText() function returns a string of the current experience out of max experience.

public string ReturnExpText()

{

return KeepExp + "/" + ExperienceNextLevel;

}

## Display Level

The DisplayLevel[2] script is used to display values on the screen during the battle scenes that are received from the LevelSystem. The main purpose of this script is to display useful information like the current level of the player’s hero and the amount of experience the player has. I started the class by creating some Unity API.

* Private Text Exptext is a variable that uses the Unity API Text.
* Private Text LevelText is a variable that uses the Unity API Text.
* Private Image ExpBar is a variable that uses the Unity API image.
* Private LevelSystem levelSystem is reference to the LevelSystem script.

private Text ExpText;

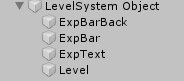
private Text LevelText;

private Image ExpBar;

private LevelSystem levelSystem;



The Awake() function is a function that initializes the variables before the game starts, it is only called once and allows the object to talk to each other once the object is initialized. What I did here is attached the private Text and Image variable to the component in my game called “ExpText”, “Level” and “ExpBar”. I then created an instance of the LevelSystem and call the SetLevelSystem(). How the scripts find the components is by searching the name and type of the component in my object. In this case, the DisplayLevel script is attached to my LevelSystem Object, the text and image that the Awake() function is looking for is inside the object.





private void Awake()

{

ExpText = transform.Find("ExpText").GetComponent<Text>();

LevelText = transform.Find("Level").GetComponent<Text>();

ExpBar = transform.Find("ExpBar").GetComponent<Image>();

Attack5 = transform.Find("ExpBar").GetComponent<Button>();

LevelSystem levelSystem = new LevelSystem();

SetLevelSystem(levelSystem);

}

The Update() function is a Unity API that runs after every frame. In other words, everything inside this function is called/ran every frame. Inside the function, I call MyExpTextChange().

void Update()

{

MyExpTextChange();

}

SetLevelNumber() function is used to display the current level of the player by getting it from the LevelSystem and attaching that LevelNumber onto the LevelText and displaying the level during battles.

private void SetLevelNumber (int LevelNumber)

{

LevelText.text = "Level : " + LevelNumber;

}

MyExpTextChange() is used to display the amount of experience the user currently has during the battle. It does this by getting the return string from the LevelSystem and attaching it to Unity’s text called ExpText.

SetLevelSystem() takes in the LevelSystem. The function sets the LevelSystem object and updates the starting value gained from the LevelSystem.

GetlevelNumber() function is called and the returned level goes into SetLevelNumber, GetExperienceNormalzed() is called and the returned value is set into SetExperinceBarSize. I then subscribe to the LevelSystem event handlers OnExperienceChange and OnLevelChange whenever the Level or Experience is changed.

public void SetLevelSystem(LevelSystem levelSystem)

{

this.levelSystem = levelSystem;

SetLevelNumber(levelSystem.GetLevelNumber());

SetExperienceBarSize(levelSystem.GetExperienceNormalized());

levelSystem.OnExperienceChange += LevelSystem\_OnExperienceChange;

levelSystem.OnLevelChange += LevelSystem\_OnLevelChange;

}

LevelSystem\_OnLevelChange() function sends an empty event argument. This function only updates the LevelText when the level is changed by getting the LevelNumber from the LevelSystem.

private void LevelSystem\_OnLevelChange(object sender, System.EventArgs e)

{

//Level change, update text//

SetLevelNumber(levelSystem.GetLevelNumber());

}

LevelSystem\_OnExperienceChange() function sends an empty event argument. This function is used to update the health bar image when the player gains experience by calling GetExperienceNormalized function inside the LevelSystem.

private void LevelSystem\_OnExperienceChange(object sender, System.EventArgs e)

{

//Experience change, update Bar Size//

SetExperienceBarSize(levelSystem.GetExperienceNormalized());

}

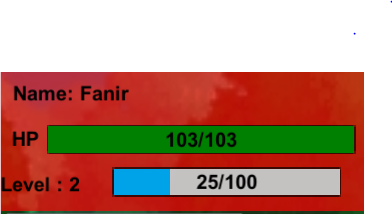
The SetExperienceBarSize() is used to fill the amount of the bar by getting the ExperienceNormalized.

private void SetExperienceBarSize(float ExperienceNormalized)

{

ExpBar.fillAmount = ExperienceNormalized;

}





1. SetLevelNumber displays the Text for the current level
2. MyExpTextChange displays the Text for experience
3. Fills the image bar to the right amount of experience

# Health System

The health system allows the hero to deal damage to the enemy or heal itself, this also applies to the enemy as well. It determines who wins or loses the battle by checking whose health reaches zero first. In my game there are 3 scripts for the health system. MyHealthSystem, EnemyHealthSystem & HealthSystem.

MyHealthSystem deals with the calculation for the damage received by the enemy and the calculation of the hero healing itself.

EnemyHealthSystem deals with the calculation for the damage received by the player’s hero and the calculation of the enemy healing itself.

HealthSystem is reasonable for displaying the results and the attack system.

Each hero has a different health system as they progress differently, the stats and the attacks the gain from leveling up is different. Most of the code is the same for all the heroes so I will only explain one Health system.

## Hero Health

I created a script called Hero\_Health[3], in this script, I created event handlers that will trigger when damaged or healed. The event handlers notify when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

The healthAmount variable is used for the current health of hero and the healthAmountMax is the maximum health for the hero. The amountValue is the amount of damage or heal the hero.

public int healthAmount;

private int healthAmountMax;

public int amountValue;

The Damage() function is how the enemy will be doing damage to the player’s hero. The function takes in an integer value (represent the power/damage of the enemy’s attack) and subtracts the value from the hero’s healthAmount. I then make the amount value equal to the amount value. If the hero’s health is less than or equal to zero, set healthAmount to zero and run the FanirGameOver scene. I created an instance of the LevelSystem and called the addExperience function in the LevelSystem to add the experience points since the player lost the battle, they will only gain 25 experience points. I also called the ReturnExpText() inside the LevelSystem, this function returns the experience gained and displays it as a Text. If OnDamaged is not equal to null, the EventArgs.Empty field is passed to the OnDamaged Method.

public void Damage(int amount)

{

//if the heros's health is less than

or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("FanirGameOver");

LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(25);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

The heal() function allows the player to heal their hero, the hero function takes in an integer value (represents the heal amount) and adds it to the healthAmount. I also make the amountValue equal to the health amount value. I did an error check, if the healthAmount is greater than the healthAmountMax, the healthAmount will equal to the healthAmountMax. This is so that the hero’s health with never go pass the maximum capacity. If OnHealed is not equal to null, the EventArgs.Empty field is passed to the OnHealed Method.

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

MyGetHealthNormalized() function returns the current health from the maximum.

public float MyGetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

MyHPTextReturn() function returns the current health from the maximum in a string format.

public string MyHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

Note: If the player wins the battle, the HeroGameWon scene will appear, this is the same for all heroes.



The “Next Battle” button will start the next battle by calling the Hero fight scene.

The “Main Menu” button will take the player back to the start of the game but also reset the hero’s stats, level, and experience. This is done in the MainMenu script.

public void Main\_Menu()

{

SceneManager.LoadScene("SampleScene");

LevelSystem.KeepLevel = 1;

LevelSystem.KeepExp = 0;

HeroLevelStats.KeepAttackStats = 0;

HeroLevelStats.KeepDefenceStats = 0;

HeroLevelStats.KeepHealthStats = 0;

EnemyLevelStats.EnemyKeepAttackStats = 0;

EnemyLevelStats.EnemyKeepHealthStats = 0;

}

## 5.2 Enemy Health

This script is like Hero Health, but this applies to the enemy’s health. This is where the enemy will be damaged and healed.

To start, I created a script called Hero\_EneHealth[3], I then created the same event handlers, OnDamage, and OnHealed this will notify when the enemy’s health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

I then created the healthAmount and HealthAmountMax variable to represent the health the enemy has and the enemy’s maximum health. amountValue is used to get the damage/heal value.

private int healthAmount;

private int healthAmountMax;

public int amountValue;

The damage function is like the hero’s damage function. The function takes in an integer value (represent the power/damage of the player attack) and subtracts it from the enemy’s healthAmount, the amountValue gets the damage value. If the level is/greater than 14 and the healthAmount reaches zero, it creates a new instance of the ButtonNewHero script, sets the healthAmount to 0. After that ScencManager will load the “GameBeat” scene, Set the Boolean value to true for the FanirBossBeat inside ButtonNewHero.

All the heroes have a BossBeat Boolean value inside the ButtonNewHero script. Whichever starter the player beats the game with, it will set that respected Boolean value to true.

This part of the code is important as it will be used to check if the player beats the game.

Since the final boss battle is triggered once the player reaches 14 and upwards.

Else if the enemy’s health reaches zero, Load the “HeroGameWon” scene using the LoadScene API. I then created an instance of the LevelSysteem and called the function AddExperience(), since the player won the battle, they gain 100 experience points. I also called the ReturnExpText(). If OnDamaged is not equal to null, the EventArgs.Empty field is passed to the OnDamaged Method.

public void Damage(int amount)

{

LevelSystem levelSystem = new LevelSystem();

//if the enemy's health is less than or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (levelSystem.Level >= 15 && healthAmount <= 0)

{

ButtonNewHero newHero = new ButtonNewHero();

healthAmount = 0;

SceneManager.LoadScene("GameBeat");

ButtonNewHero.FanirBossBeat = true;

}

else if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("FanirGameWon");

levelSystem.AddExperience(100);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

The Heal() function is the exact same as Hero’s Heal() function.

public void Heal(int amount)

{

healthAmount += amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

EnemyGetHealthNormalized() function returns the current health from the maximum.

public float EnemyGetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

EnemyHPTextReturn() function returns the current health from the maximum in a string format.

public string EnemyHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

Note: If the player wins the battle, the HeroGameWon scene will appear, this is the same for all heroes.



The “Next Battle” button will start the next battle by calling the Hero fight scene.

The “Main Menu” button will take the player back to the start of the game but also reset the hero’s stats, level, and experience. This is done in the MainMenu script.

public void Main\_Menu()

{

SceneManager.LoadScene("SampleScene");

LevelSystem.KeepLevel = 1;

LevelSystem.KeepExp = 0;

HeroLevelStats.KeepAttackStats = 0;

HeroLevelStats.KeepDefenceStats = 0;

HeroLevelStats.KeepHealthStats = 0;

EnemyLevelStats.EnemyKeepAttackStats = 0;

EnemyLevelStats.EnemyKeepHealthStats = 0;

}

Note: If the hero beats the boss, the GameBeat scene will appear. This applies to all heroes



## 5.3 Hero Health Bar

Hero\_HealthBar[3] script puts the Fanir\_EnemyHealth[3] and Fanir\_Health[3] together, I created some Image & Text variables that will be used to display information during the gameplay. Since the variables are public, I will have to manually drag the required Text into the script that is attached to the object.

*Note:* this script is the same for all the heroes but just with the respected function calls for the right hero. I’m going to show a general HealthBar script.

private Image BarImage;

private Image BarImagemy;

private Image CircleDamage;

public Text MyHPText;

public Text EmeHPText;

public Text DamageText;

public Text Attack5Text;

public Text Attack6Text;

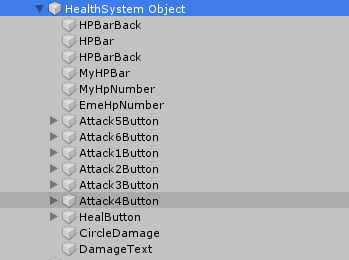


I then reference the required scripts.

private Hero\_EneHealth HeroEnemyHealth;

private Hero\_Health HeroHealth;

Awake() function initializes the HPBar, MyHPBar & CircleDamage by referencing the component type and name inside the HealthSystem object.



public void Awake()

{

BarImage = transform.Find("HPBar").GetComponent<Image>();

BarImagemy = transform.Find("MyHPBar").GetComponent<Image>();

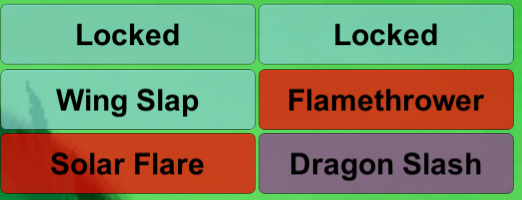
CircleDamage = transform.Find("CircleDamage").GetComponent<Image>();

}

I call Unity’s Update() function to run EneTextChange() & MyTextChange().

I create an instance of the LevelSystem and an if statement. In this case, If the level equals to or greater than 3, Attack5Text is changed to “Fire Blast” else the value for Attack5Text is “Locked”.

If the level equals or greater then 9, Attack6Text is changed to “Eruption” else the value for Attack6Text is “Locked”. This part only for Fanir, the other heroes have different names for their attacks. This part of the code just changes the string inside the Text variable to the right attack name once a certain level is reached otherwise the string will indicate that the player cannot use this attack.



void Update()

{

MyTextChange();

EmeTextChange();

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 3)

Attack5Text.text = "Fire Blast";

else

Attack5Text.text = "Locked";

if (levelSystem.Level >= 9)

Attack5Text.text = "Eruption";

else

Attack6Text.text = "Locked";

}

In the Start() function sets the CircleDamage.enable and DamageText.enable to false, this makes the image and text invisible. I created two instances, HeroLevelStats to call the HeroStatsLevel() and EnemyLevelStats to call the EnemyStatsLevel().

Created an instance of the Hero\_EneHealth and passed the value 100 plus a value inside the EnemyLevelStates script called EnemyKeepHealthState. This line sets the health for the enemy, EnemyKeepHealthState increase the health stats as the enemy levels up. Created another instance of the Hero\_Health and passed the value 100 + a value inside the FanirStates called KeepHealthState. This increases the health of the hero as the player levels up.

*Note:* Fanir’s & Panboo’s starting health is 100 and Lagoon’s starting health is 110.

I then go into the health normalized by looking inside the Fanir\_EnemyHealth script and call GethealthNormalized() function. I also called the player’s hero MyhealthNormalized() from the Fanir\_Health script. SetHealth takes GetHealthNormalized() and MySetHealth takes MyGetHealthNormalized().

I then updated the enemy’s health OnDamaged by calling the event handler along with the player’s hero OnDanaged. This is the same for the OnHealed for both the player and enemy.

private void Start()

{

CircleDamage.enabled = false;

DamageText.enabled = false;

HeroLevelStats Lagoon\_States = new HeroLevelStats();

EnemyLevelStats enemyLevelStats = new EnemyLevelStats();

Lagoon\_States.LagoonStatesLevel();

enemyLevelStats.LagoonEnemyStatesLevel();

lagoonEnemyHealth = new Lagoon\_EneHealth(100 + EnemyLevelStats.EnemyKeepHealthState);

lagoonHealth = new Lagoon\_Health(110 + HeroLevelStats.KeepHealthState);

//Get the healthNormalized

SetHealth(lagoonEnemyHealth.GetHealthNormalized()); MySetHealth(lagoonHealth.MyGetHealthNormalized());

//Update the health on damage.

lagoonEnemyHealth.OnDamaged += EneHealthSystem\_OnDamaged;

lagoonHealth.OnDamaged += MyHealthSystem\_OnDamaged;

//Update the health on heal.

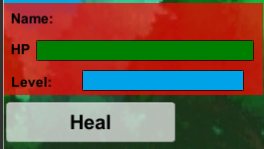
lagoonEnemyHealth.OnHealed += EneHealthSystem\_OnHealed;

lagoonHealth.OnHealed += MyHealthSystem\_OnHealed;

}

The HealHero() function is used to heal the hero when the heal button is pressed. I created an instance of the ButtonDelay script and called the buttonAttackDelay(). After, I get a random value between 15 and 30 and use it as the healing value. I called the DamageBox() function and set the colour of the square to blue. Then I called the ValueToHero() and the Delay() function.





public void HealHero()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

lagoonHealth.Heal(Random.Range(15, 30));

StartCoroutine(DamageBox());

CircleDamage.color = Color.blue;

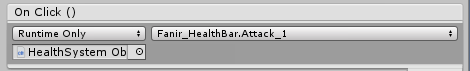
ValueToHero();

Delay();

}

# 

The Attack() function is used to deal damage to the enemy’s health system. The way this is done is by attaching the attack function to a button and when pressed, it to call the function. I started by creating a new instance of ButtonDelay script and calling the ButtonAttackDelay(). I also called the DamageBox(). Once that is done, I called the Damage() from the enemy’s health system and added value, I also called the HeroLevelStats script added the KeepAttackStats variables to the damage amount. Set the colour of the Box to green and called the ValueToEnemy() & Delay(). This is the same for the first four attacks available to the hero at the start of the game. The function is the only difference between the starter heroes, the attack damage is different for all heroes. Fanir has the highest attack damage out of all the heroes.





//Attack 1 – 4 is the same.

public void Attack\_1()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

lagoonEnemyHealth.Damage(5 + HeroLevelStats.KeepAttackState);

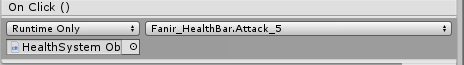
CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

Attack\_5 and Attack\_6 are different. This is because attack 5 and attack 6 are unlocked as the player levels up. The function starts by creating a new instance of the LevelSystem. If the level is/greater than 3 then the attack is enabled by setting the interactable value to true. I create a new instance of ButtonDelay script and calling the ButtonAttackDelay(). I also called the DamageBox(). Once that is done, I called the Damage() from the enemy’s health system and added value, I also called the HeroLevelStats script added the KeepAttackStats to the damage amount. Set the colour of the Box to green and called the ValueToEnemy() & Delay(). If the level is less than 3, the code looks for the button called “Attack5Button” and disables the button by setting the interactable to false. This is the same for Attack\_6. Both Attack\_5 and Attack\_6 are called by attaching the function to the attack button. Once the button is unlocked, the name of the attack appears, otherwise, it stays locked.





The Delay() function is used to make a delay every time the attack/heal button is pressed. The function uses the invoke(), with this I can specify what method/function to call and for how long. In my case, I’m calling the “Enemychoice” & “MyTextChange” and have a 1-second delay for the run time of both functions.

public void Delay()

{

Invoke("Enemychoice", 1);

Invoke("MyTextChange", 1);

}

The DamageBox() is an IEnumerator function. This allows me to stop the process of the function at any point. I’m using this function to handle the DamageBox’s text and image. When an attack/heal button is pressed, this function is called, it makes the CircleDamage and DamageText appear by setting the Boolean value to true, waits for 2 seconds and then hides the CircleDamage and DamageText by setting the Boolean value to false.



IEnumerator DamageBox()

{

CircleDamage.enabled = true;

DamageText.enabled = true;

yield return new WaitForSeconds(2.0f);

CircleDamage.enabled = false;

DamageText.enabled = false;

}

EnemyChoice() function is the main way the enemy attacks the player’s hero. This function is called every time the player attacks the enemy or heals itself. I started this by creating an integer called randomnumber. This will decide what the enemy does. The randomnumber gets a random value from 1 to 8. If the randomnumber has the value 3, the enemy will heal itself by calling the heal function in the Hero\_EneHealth script. The heal value/amount is always 30. The Circledamage is then set to yellow. I call the ValueToEnemy() to display the amount the enemy healed and eneTextChange(). If the randomnumber is any other number. It will call the damage function from the Hero\_Health and the damage amount is a random value from 5 to 25. I also call the EnemyKeepAttackStats variable from EnemyLevelStats script. The variable is used to boost the enemy’s attack as the level goes up and subtracts it from the hero’s defence. The CircleDamage colour is set to red and I call ValueToHero() to show the damage the player received & EneTextChange() to update the health text for the player.

public void Enemychoice()

{

int randomnumnber;

//Generate random number between 1 to 5.

randomnumnber = Random.Range(1, 8);

//If the generate number is 2, the enemy hero would heal.

if (randomnumnber == 3)

{

FanirEnemyHealth.Heal((30));

CircleDamage.color = Color.yellow;

ValueToEnemy();

EmeTextChange();

}

else

{

//otherwise it would attack.

FanirHealth.Damage(Random.Range(5, 25) + EnemyLevelStats.EnemyKeepAttackState - HeroLevelStats.KeepDefenceState);

CircleDamage.color = Color.red;

ValueToHero();

EmeTextChange();

}

}

EmeTextChange() is used to display the enemy’s current health by making the EmeHPText get the return value from Hero\_EneHealth script. This takes the string and puts it into the text to be displayed during battle.

public void EmeTextChange()

{

EmeHPText.text = FanirEnemyHealth.emeHPTextReturn();

}

MyTextChange() is used to display the player’s health by getting the return string value from the Hero\_Health and applying it to MyHPText.

public void MyTextChange()

{

MyHPText.text = FanirHealth.MyHPTextReturn();

}

ValueToHero() gets the variable from the hero’s health and displays in DamageText inside the circleDamage. This value represents the amount of damage the hero received.

public void ValueToHero()

{

DamageText.text = FanirHealth.amountValue.ToString();

}

ValueToEnemy() gets the variable from the enemy’s health and displays in DamageText inside the circleDamage. This value represents the amount of damage the enemy received.

public void ValueToEnemy()

{

DamageText.text = FanirEnemyHealth.amountValue.ToString();

}

EneHealthSystem\_OnDamaged() function takes in a reference to the event and event data.  
When the damage function runs, the event handler will run. This gets the enemy’s current health out of the maximum by calling the getHealthNormalized() and sets that value to SetHealth.

private void EneHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

SetHealth(FanirEnemyHealth.GetHealthNormalized());

}

MyHealthSystem\_OnDamaged() function takes in a reference to the event and event data.

When the damage function runs, the event handler will run. This gets the hero’s current health out of the maximum by calling the getHealthNormalized() and sets that value to MySetHealth.

private void MyHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

MySetHealth(FanirHealth.MyGetHealthNormalized());

}

EneHealthSystem\_OnHealed() function takes in a reference to the event and event data.

When the heal function runs, the event handler will run. This gets the enemy’s current health out of the maximum by calling the getHealthNormalized() and sets that value to SetHealth.

private void EneHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

SetHealth(FanirEnemyHealth.GetHealthNormalized());

}

MyHealthSystem\_OnHealed() function takes in a reference to the event and event data.

When the heal function runs, the event handler will run. This gets the hero’s current health out of the maximum by calling the getHealthNormalized() and sets that value to MySetHealth.

private void MyHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

MySetHealth(FanirHealth.MyGetHealthNormalized());

}

SetHealth() gets the float value from the functions above and changes the BarImage to fit the value. In this case the BarImage is the health bar that says the current health is 62 out of 100, the BarImage will look like the image below. This function displays the health bar for the enemy.



private void SetHealth(float healthNormalized)

{

BarImage.fillAmount = healthNormalized;

}

MySetHealth() gets the float value from the functions above and changes the BarImage to fit the value. In this case the BarImage is the health bar says the current health is 62 out of 100, the BarImagemy will look like the image below. This function displays the health bar for the player.



private void MySetHealth(float MyhealthNormalized)

{

BarImagemy.fillAmount = MyhealthNormalized;

}

# Stats

Stats are important in my game.

As the player level’s up their hero, the hero gets stronger with all three stats (health, attack & defence), but the stats rate is random for two stats while once stat increase at a constant rate. That stat are the hero’s “specialty”. What I mean by that is the hero is more focused on this stat rate than the others. The ability to increase the hero’s stats is thanks to the HeroLevelStats script.

The other important part of my game is how the enemy’s get stronger as you climb up the levels. Since the enemies are always the same level as the player’s hero. They get the stats to boost at the same time as the heroes. I created a script called EnemyLevelStats that increase the health and attack of the enemy. It does not increase the defence because it will make the enemy too strong and the point of the game is that the player’s hero is meant to be a bit stronger until the boss battle. The boss is always going to be stronger than the hero.

## Hero Stats

Starting the HeroLevelStats[5] script, I created six public variables.

* HealthIncreaseLevel: Increases the health stats as the hero levels up.
* KeepHealthStats: The starting stats.
* AttackIncreaseLevel: Increases the attack stats as the hero levels up.
* KeepAttackStats: The starting stats.
* DefenceIncreaseLevel: Increases the defence stats as the hero levels up.
* KeepDefenceStats: The starting stats.

public int HealthIncreaseLevel = KeepHealthStats;

public static int KeepHealthStats = 0;

public int AttackIncreaseLevel = KeepAttackStats;

public static int KeepAttackStats = 0;

public int DefenceIncreaseLevel = KeepDefenceStats;

public static int KeepDefenceStats = 0;

I called the start(), inside I ran FanirStatsLevel(), LagoonStatsLevel() & PanbooStatsLevel()

functions. This is important to make sure the hero’s stats increase when the condition is met.

void Start()

{

FanirStatsLevel();

LagoonStatsLevel();

PanbooStatsLevel();

}

I started with Fanir’s stat boost. Inside FanirStatsLevel() I created an instance of the LevelSystem to get the level of the hero and put that into an if statement to check if the level can be divided by 3 with nothing remaining. Inside the if statement, 7 is added to AttackIncreaseLevel variable. KeepAttackStats will equal the AttackIncreaseLevel. I also check if the level can be divided into two. If the statement is true, it will increase the HealthIncreaseLevel will increase by a random value between 3 to 8. KeepHealthStats will equal HealthIncreaseLevel. The DefenceIncreaseLevel will also increase by a random value between 3 to 8. KeepDefenceStats will equal DefenceIncreaseLevel. If the level cannot be divided into two or three, it will not increase the stats.

public void FanirStatsLevel()

{

//Every 3 level get more attack boost//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level % 3 == 0)

{

AttackIncreaseLevel += 7;

KeepAttackStats = AttackIncreaseLevel;

}

//every 2 level gets defence and health boost//

if(levelSystem.Level % 2 == 0)

{

HealthIncreaseLevel += Random.Range(3, 8);

KeepHealthStats = HealthIncreaseLevel;

DefenceIncreaseLevel += Random.Range(3, 8);

KeepDefenceStats = DefenceIncreaseLevel;

}

else

{

//otherwise no stats boost//

AttackIncreaseLevel = KeepAttackStats;

HealthIncreaseLevel = KeepHealthStats;

DefenceIncreaseLevel = KeepDefenceStats;

}

}

I then started with the function to increase Lagoon’s stats.

Inside LagoonStatsLevel() I created another instance of the LevelSystem and grabbed the level of the hero. if the level can be divided by 3, I would increase the AttackIncreaseLevel by a random value between 3 to 5. KeepAttackStats will get the value of AttackIncreaseLevel.

If the level can be divided by 2, the healthIncreaseLevel will increase by a constant 10 and that value will be added to KeepHealthStats. A random value between 3 to 8 will be added to DifenceIncreaseLevel. KeepDefenceStats variable will take the value of DifenceIncreaseLevel.

If the level cannot be divided into two or three, it will not increase any stats.

public void LagoonStatsLevel()

{

//Every 3 level get more attack boost//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level % 3 == 0)

{

AttackIncreaseLevel += Random.Range(3, 5);

KeepAttackStats = AttackIncreaseLevel;

}

//every 2 level gets defence and health boost//

if (levelSystem.Level % 2 == 0)

{

HealthIncreaseLevel += 10;

KeepHealthStats = HealthIncreaseLevel;

DefenceIncreaseLevel += Random.Range(3, 8);

KeepDefenceStats = DefenceIncreaseLevel;

}

else

{

//otherwise no stats boost//

AttackIncreaseLevel = KeepAttackStats;

HealthIncreaseLevel = KeepHealthStats;

DefenceIncreaseLevel = KeepDefenceStats;

}

}

The last hero to get a stat boost is Panboo. I inside the PanbooStatsLevel() I created the last instance of the LevelSystem to get the level variable and I put that level into an if statement. If the level can be divided by three, it will increase the AttackIncreaseLevel by a random value between 3 and 5. KeepAttackStats will get the value of AttackIncreaseLevel. If the level can be divided by 2, then the HealthIncreaseLevel will get a random value from 3 to 8. KeepHealthStats will equal to HealthIncreaseLevel. The DefenceIncreaseLevel will increase by a constant 10. KeepDefenceStats will equal to DefenceIncreaseLevel.

If the level cannot into divided by two or three, it will not increase any stats.

public void PanbooStatsLevel()

{

//Every 3 level get more attack boost//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level % 3 == 0)

{

AttackIncreaseLevel += Random.Range(3, 5);

KeepAttackStats = AttackIncreaseLevel;

}

//every 2 level gets defence and health boost//

if (levelSystem.Level % 2 == 0)

{

HealthIncreaseLevel += Random.Range(3, 8);

KeepHealthStats = HealthIncreaseLevel;

DefenceIncreaseLevel += 10;

KeepDefenceStats = DefenceIncreaseLevel;

}

else

{

//otherwise no stats boost//

AttackIncreaseLevel = KeepAttackStats;

HealthIncreaseLevel = KeepHealthStats;

DefenceIncreaseLevel = KeepDefenceStats;

}

}

## Enemy Stats

Starting the EnemyLevelStats[5] script, I have

I created four public variables.

* EnemyHealthIncreaseLevel: Increases the health stats as the hero levels up.
* EnemyKeepHealthStats: The starting stats.
* EnemyDefenceIncreaseLevel: Increases the defence stats as the hero levels up.
* EnemyKeepDefenceStats: The starting stats.

private int EnemyHealthIncreaseLevel = EnemyKeepHealthState;

public static int EnemyKeepHealthState = 0;

private int EnemyAttackIncreaseLevel = EnemyKeepAttackState;

public static int EnemyKeepAttackState = 0;

I called the start() and ran EnemyStatsLevel() functions. This function is used to increase the enemy’s stats when the condition is met

void Start()

{

EnemyStatesLevel();

}

EnemyStatsLevel() has an instance of the LevelSystem to get access to the level variable.

An if statement checks if the level is equal or greater than 14. If this condition is true, it means

that the player is currently battling with the boss. I created an instance of the HeroLevelStats to take the HealthIncreaseLevel variable and added to the enemy’s EnemyHealthIncreaseLevel and multiply by 2. I also got the hero’s AttackIncreaseLevel and added it to the enemy’s EnemyAttackIncreaseLevel variable.

This is important as the boss is a “new look” and a better/stronger than the player’s hero.

If the level can be divided by 2, EnemyAttackIncreaseLevel will increase by a random value between 2 to 5, EnemyHealthIncreaseLevel will also increase by a random value between 5 to 15. If the level cannot be divided by two or is greater then 14, it will not increase any stats.

public void EnemyStatsLevel()

{

//super fanir stats boost

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 14)

{

//gets the health and attack and appliys it to itown//

HeroLevelStats heroLevelStats = new HeroLevelStats();

EnemyHealthIncreaseLevel = heroLevelStats.HealthIncreaseLevel \* 2;

EnemyKeepHealthStats = EnemyHealthIncreaseLevel;

EnemyAttackIncreaseLevel = heroLevelStats.AttackIncreaseLevel \* 2;

EnemyKeepAttackStats = EnemyAttackIncreaseLevel;

}

else if (levelSystem.Level % 2 == 0)

{

//otherwise normal stats boost//

EnemyHealthIncreaseLevel += Random.Range(5, 15);

EnemyKeepHealthStats = EnemyHealthIncreaseLevel;

EnemyAttackIncreaseLevel += Random.Range(2, 5);

EnemyKeepAttackStats = EnemyAttackIncreaseLevel;

}

else

{

//otherwise no stats boost

EnemyHealthIncreaseLevel = EnemyKeepHealthStats;

EnemyAttackIncreaseLevel = EnemyKeepAttackStats;

}

}

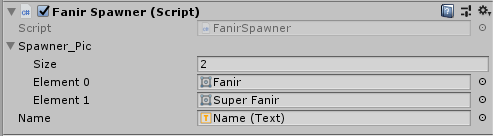
# Spawn System

There is 8 spawn system in my game, one for each hero and one for the enemies.

The code used to create the enemy spawn system are all the same, the difference is that the name and sprite inside the system have different characters and different names. This applies to the hero spawn system too. The Spawn system is used to spawn and enemy for the hero to battle, this is also how the boss battle will be triggered.

## Hero Spawn

I created a game object and attached the HeroSpawner[8] script to it. This is where we spawn the player’s hero. Inside the script I created two arrays, a string array called Namelist which how the names of the hero, and an array of sprite called Spawner\_Pic. I also created a Text called Name. Inside Spawner\_Pic, I put in the hero’s sprite that will spawn when the battle starts.



public Sprite[] Spawner\_Pic;

//array of names

private string[] Namelist = { "Fanir", "Super Fanir", "lagoon", "Super Lagoon"};

public Text Name;

I created a start() then calls the HeroSpawn(), this will run every time the battle starts.

void Start()

{

HeroSpawn();

}

The HeroSpawn() is responsible for selecting the right sprite to spawn. This is done by calling the HeroBossBeat variable inside the ButtonNewHero. If HeroBossBeat is true, it means that the player has beaten the boss and is awarded with the new look of that hero. I put the second element in the Spawner\_Pic array and make the Name. Text equal to the second name from the Namelist array. If HeroBossBeat is false, it means that the player didn’t beat the boss, the sprite will spawn the first element in the Spawrer\_Pic array make Name.text equal to the first element in the Namelist Array.

void HeroSpawn()

{

//checks if player beat game with fanir//

if (ButtonNewHero.FanirBossBeat)

{

//is yes, then spawn super fanir

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[1];

Name.text = " Name : " + Namelist[1];

}

else

{

//if not then spawn normal fanir

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[0];

Name.text = " Name : " + Namelist[0];

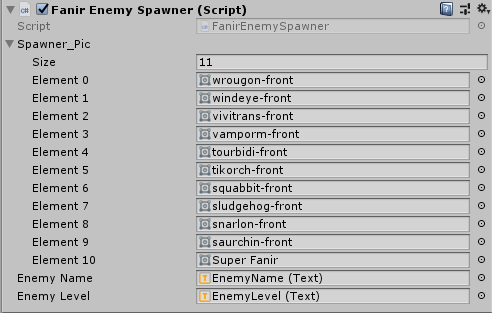
}

}

## 7.2 Enemy Spawn

I created a game object and attached the EnemySpawner[8] script to it. This is where we spawn enemies. Inside the script I created two arrays, a string array called Name which holds the names of the enemies and an array of Sprite called Spawner\_Pic. I also created a two Text called EnemyName, EnemyLevel & a private integer called rand.

Inside Spawner\_Pic, I put the enemy sprite that will spawn when the battle starts.



private int rand;

public Sprite[] Spawner\_Pic;

//arrray of names//

private string[] Name = { "Wallabyss", "Toalow", "Penguna", "Mantitar", "Kineling", "Magmelope", "Wintora", "Dracaza", "Quackal", "Staropotamus", "Territe", "Super Fanir" };

public Text EnemyName;

public Text EnemyLevel;

I called the start() and created an instance of the LevelSystem to get access to the level variable and attach it to the EnemyLevel.Text. This will display the enemy’s level and make sure that the hero and enemy are always the same levels. I also call the Change() which is responsible for spawning the right enemy at the right time.

void Start()

{

Change();

LevelSystem levelSystem = new LevelSystem();

EnemyLevel.text = "Level : " + levelSystem.GetLevelNumber();

}

The Change() has an instance of the LevelSystem as well. The rand variable is equal to a random number between 0 and 9. The number represents the enemy that will spawn. Rand is then put into the Spawner\_Pic array and that enemy with Rand number will spawn. Rand is also used to get the right name of the enemy by putting it into the Name array.

The boss sprite will trigger when the player reaches level 14 or greater. It will force the boss to spawn by placing the number of the boss sprite into the Spawner\_Pic array. In my case, the boss’s sprite is 10. I also get the right name for the boss to appear by putting 10 to the Name array and attaching it to the EnemyName.Text.

void Change()

{

LevelSystem levelSystem = new LevelSystem();

//Gets a random number with in 0 to the Srpite array Length

rand = Random.Range(0, 9);

//Displays the sprite that was selected by random

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[rand];

//Displays the name that is accesed with the sprite

EnemyName.text = " Name : " + Name[rand];

//boss sprite//

if (levelSystem.Level >= 14)

{

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[10];

EnemyName.text = " Name : " + Name[11];

}

}

# Button Functions

The button functions are small scripts that are used to disable the buttons and make the new hero button appear when the right condition is met.

## Button Delay

I created a script called ButtonDelay[9], inside the script, I created 7 private button variables. These buttons are the 6 attack moves that the hero knows and the heal button.

private Button Attack1;

private Button Attack2;

private Button Attack3;

private Button Attack4;

private Button Attack5;

private Button Attack6;

private Button Heal;

I then called Unity’s Awake() and got the reference to all the buttons.

void Awake()

{

//Gets the reference//

Attack1 = GetComponent<Button>();

Attack2 = GetComponent<Button>();

Attack3 = GetComponent<Button>();

Attack4 = GetComponent<Button>();

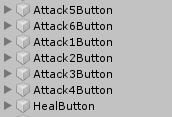
Attack5 = GetComponent<Button>();

Attack6 = GetComponent<Button>();

Heal = GetComponent<Button>();

}

I then created a public IEnumerator function called ButtonAttackDelay(). Inside the function I began to look for a button that I need to disable, I’ve done this by searching for the “Attack1Button” and the component type. Once found I attached it to Attack1. I set Attack1 interactable to false which disables the button and calls the WaitForSecond(2) function. The number inside the WaitForSecond() represents the amount of time the delay is. After the delay, I set Attack1 interactable to true. I’ve done this to all 7 buttons but I’m only going to show you one as it’s the same, but the name is different.



public IEnumerator ButtonAttackDelay()

{

//Disables all the buttons//

Attack1 = GameObject.Find("Attack1Button").GetComponent<Button>();

Attack1.interactable = false;

//Wait 2 second//

yield return new WaitForSeconds(2f);

//Enable all button//

Attack1 = GameObject.Find("Attack1Button").GetComponent<Button>();

Attack1.interactable = true;

}

## 8.1 Button New Hero

This is where we unlock the new hero. In the ButtonNewHero[5] script, the code checks if the player has beaten the game with a Boolean. I start by creating the button called NewHero and three public bools listed below.

* FanirBossBeat: Check is the player beats the game with Fanir.
* LagoonBossBeat: Check is the player beats the game with Lagoon.
* PanbooBossBeat: Check is the player beats the game with Panboo.

I call the Start() and use an if statement, if the FanirBossBeat, LagoonBossBeat, and PanbooBossBeat are true, it will make the NewHero button appear, otherwise, it will stay hidden.

private void Start()

{

//if all the bosses are beat, secret her button appears otherwise it hidden

if (FanirBossBeat && LagoonBossBeat && PanbooBossBeat)

{

NewHero.gameObject.SetActive(true);

}

else

{

NewHero.gameObject.SetActive(false);

}

}

# Secret Hero

The only way to unlock the secret hero is by beating the game with all three characters. The main difference between the secret hero and the starter hero is that there is no boss battle at level 14, the game just keeps going until you get bored, there are 40 different enemies to battle instead of 10. I’m not going to show you code because it’s the same code as the starter heroes in section 5.

# Problem Solving

I encountered a few problems when creating and coding my game.

One of the problems was the level system. When the player wins the battle and levels up, the level will not carry to the next battle. This was a big problem in my game as the level is important. The way I got around this by looking around the web for similar problems and how they fixed it. I then came up with an idea to pass on the level value to another variable and use that as the level, every time the level goes up, it will be added to variables. This fixed the problem but took me a while to figure out.

Another bug was when the player wants to go back to the main menu, the stats, level & experience will still be kept. This means that if the player picks another hero, that hero will be the same level as the previous hero. This is a bug as everything is meant to be reset. I got around this problem by setting resetting the value to zero when the main menu is clicked.

I had a major issue with the characters in my game. When I started to create my game, I used the copyrighted Pokémon characters in my game. This was against the law as I need permission to use Game Freak’s characters (which I didn’t have). I got around this problem by looking for an open-source character. I found this open-source RPG game with a lot of characters called Tuxemon. I tried emailing them to ask for permission to use their character but Tuxemon never got back to me, I’ve read their terms on using their character and before I used them, I made sure I could with my project supervisor. He said it was fine and now they are in my game.

# Conclusion

The outcome of my project is a working game that functions the way I designed it. My game has a fully working level system where when the player wins/loses the battle, they gain some experience. A fully working heath system for both the player’s hero and the enemy. Stats work great and when battling the final boss, it gets the player’s stats and multiply it by 2 and adds it to the boss’s stats. The original plan is to make the final boss a better version of you. I got the awards sorted out as well, the game can tell if you beat the boss and give your hero a new look. What didn’t work is the animations. I tried it but was a bit harder than I expected, and I wouldn’t have enough time to do it, but I did add something instead of animations, I added a challenge. If the player beats the challenge, the player unlocks a new hero. My game was a success because my brother beta tested it and he enjoyed it; I accomplished my goal for the game.

The overall conclusion is that I enjoyed designing and creating my game, it turns from doing work to a hobby.

I’m not done with the game; I do plan on adding some animations and some new features like saving/loading the game.

# References

[1] Tuxemon, “characters”, <https://github.com/Tuxemon/Tuxemon>

[2] Code Monkey, “Level System in Unity”, <https://www.youtube.com/watch?v=kKCLMvsgAR0>, 2019.

[3] Code Monkey, “Health Bar Damage Taken Effects (Unity Tutorial)“, <https://www.youtube.com/watch?v=cR8jP8OGbhM&t=212s>, 2019

[4] Mosh Hamedani, “C# Basics for beginners: Learn C# fundamentals by coding”, <https://www.udemy.com/>

[5] Unity Learn, “Beginner Scripting”, Unity Learn.

[6] Brackeys, “How to make a dialogue System in Unity”, <https://www.youtube.com/watch?v=_nRzoTzeyxU>, 2017

[7] UnityBeginner, “Unity3D how to : Pass variable through Scene”, <https://www.youtube.com/watch?v=kwRlfAtcmOE>, 2014

[8] J.A.Whye, “Unity 2D: Many Game Object from a prefab & array of Sprite”, <https://www.youtube.com/watch?v=LJrc1NHY23w>, 2015

[9] Christopherr\_untiy, “How to make an update() wait for a few seconds before running the codes?”, <https://www.reddit.com/r/Unity3D/comments/82dy4d/how_to_make_an_update_wait_for_a_few_seconds/>, 2018

# Code

## Fanir\_EneHealth

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Fanir\_EneHealth : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

private int healthAmount;

private int healthAmountMax;

public int amountValue;

public Fanir\_EneHealth(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the enemy's health

public void Damage(int amount)

{

LevelSystem levelSystem = new LevelSystem();

//if the enemy's health is less than or equal to 0.

healthAmount -= amount;

amountValue = amount;

//Boss battle trigger//

if (levelSystem.Level >= 14 && healthAmount <= 0)

{

//if the button is beaten//

ButtonNewHero newHero = new ButtonNewHero();

healthAmount = 0;

SceneManager.LoadScene("GameBeat");

ButtonNewHero.FanirBossBeat = true;

}

else if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("FanirGameWon");

levelSystem.AddExperience(100);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the enemy's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns enemy's current health

public float GetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string emeHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Fanir\_Health

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class Fanir\_Health : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

public int healthAmount;

private int healthAmountMax;

public int amountValue;

public Fanir\_Health(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the hero's health

public void Damage(int amount)

{

//if the heros's health is less than or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("FanirGameOver");

LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(25);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the hero's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns hero's current health

public float MyGetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string MyHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Fanir\_HealthBar

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.Threading.Tasks;

public class Fanir\_HealthBar : MonoBehaviour

{

//Reference to both heathbar images

private Image BarImage;

private Image BarImagemy;

private Image CircleDamage;

public Text MyHPText;

public Text EmeHPText;

public Text DamageText;

public Text Attack5Text;

public Text Attack6Text;

//Links the two hero classes.

private Fanir\_EneHealth FanirEnemyHealth;

private Fanir\_Health FanirHealth;

public void Awake()

{

//Initzilse the components to the name before game start//

BarImage = transform.Find("HPBar").GetComponent<Image>();

BarImagemy = transform.Find("MyHPBar").GetComponent<Image>();

CircleDamage = transform.Find("CircleDamage").GetComponent<Image>();

}

void Update()

{

//Update the my health and enemy health change//

MyTextChange();

EmeTextChange();

//This handles unlocking the new attacks name//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 2)

Attack5Text.text = "Fire Blast";

else

Attack5Text.text = "Locked";

if (levelSystem.Level >= 9)

Attack6Text.text = "Eruption";

else

Attack6Text.text = "Locked";

}

private void Start()

{

//game starts and hids the square and text

CircleDamage.enabled = false;

DamageText.enabled = false;

//Creates and instance of Fanir stats

HeroLevelStats Fanir\_States = new HeroLevelStats();

EnemyLevelStats enemyLevelStats = new EnemyLevelStats();

//calls function every time the game start//

Fanir\_States.FanirStatsLevel();

enemyLevelStats.EnemyStatsLevel();

//creates an instant of the health for both characters and sets the health//

FanirEnemyHealth = new Fanir\_EneHealth(100 + EnemyLevelStats.EnemyKeepHealthStats);

FanirHealth = new Fanir\_Health(100 + HeroLevelStats.KeepHealthStats);

//Get the healthNormalized

SetHealth(FanirEnemyHealth.GetHealthNormalized());

MySetHealth(FanirHealth.MyGetHealthNormalized());

//Update the health on damage.

FanirEnemyHealth.OnDamaged += EneHealthSystem\_OnDamaged;

FanirHealth.OnDamaged += MyHealthSystem\_OnDamaged;

//Update the health on heal.

FanirEnemyHealth.OnHealed += EneHealthSystem\_OnHealed;

FanirHealth.OnHealed += MyHealthSystem\_OnHealed;

}

//The hero Heal function. Once the button is pressed, this function will run.

public void HealHero()

{

//Disables the buttons buttonAttackDelay

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

//Random Value goes into heal functon for Fanir//

FanirHealth.Heal(Random.Range(15, 30));

//starts the damagebox function//

StartCoroutine(DamageBox());

//changes the box colour + shows the heal value + calls the delay function//

CircleDamage.color = Color.blue;

ValueToHero();

Delay();

}

//Attack move 1. When button is pressed, call this function.

public void Attack\_1()

{

//Disables button

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

//sents damage to Enemy health//

FanirEnemyHealth.Damage(7 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 2. When button is pressed, call this function.

public void Attack\_2()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

FanirEnemyHealth.Damage(12 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 3. When button is pressed, call this function.

public void Attack\_3()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

FanirEnemyHealth.Damage(90 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 4. When button is pressed, call this function.

public void Attack\_4()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

FanirEnemyHealth.Damage(12 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

public void Attack\_5()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 3 or geater

if (levelSystem.Level >= 3)

{

//Enables the button///

transform.Find("Attack5Button").GetComponent<Button>().interactable = true;

//same as normal attack//

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

FanirEnemyHealth.Damage(18 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 3, keep button disabled

transform.Find("Attack5Button").GetComponent<Button>().interactable = false;

}

}

public void Attack\_6()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 9 or geater

if (levelSystem.Level >= 9)

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

FanirEnemyHealth.Damage(20 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 9, keep button disabled

transform.Find("Attack6Button").GetComponent<Button>().interactable = false;

}

}

public void Delay()

{

//call invoke api and runs the EnemyChoice for 1 second

Invoke("Enemychoice", 1);

//call invoke api and runs the MyTextChange for 1 second

Invoke("MyTextChange", 1);

}

IEnumerator DamageBox()

{

//makes the button appear

CircleDamage.enabled = true;

DamageText.enabled = true;

//waits for 2 second

yield return new WaitForSeconds(2.0f);

//hids the button//

CircleDamage.enabled = false;

DamageText.enabled = false;

}

//This is the where the enemy deals damage to hero health system//

public void Enemychoice()

{

int randomnumnber;

//Generate random number between 1 to 8.

randomnumnber = Random.Range(1, 8);

//If the generate number is 3, the enemy hero would heal.

if (randomnumnber == 3)

{

FanirEnemyHealth.Heal((30));

CircleDamage.color = Color.yellow;

ValueToEnemy();

EmeTextChange();

}

else

{

//otherwise it would attack.

FanirHealth.Damage(Random.Range(5, 25) + EnemyLevelStats.EnemyKeepAttackStats - HeroLevelStats.KeepDefenceStats);

CircleDamage.color = Color.red;

ValueToHero();

EmeTextChange();

}

}

//changes the number inside the health bar

public void EmeTextChange()

{

EmeHPText.text = FanirEnemyHealth.emeHPTextReturn();

}

//changes the number inside the health bar

public void MyTextChange()

{

MyHPText.text = FanirHealth.MyHPTextReturn();

}

//gets the damage value to hero health

public void ValueToHero()

{

DamageText.text = FanirHealth.amountValue.ToString();

}

//gets the damage value to enemy health

public void ValueToEnemy()

{

DamageText.text = FanirEnemyHealth.amountValue.ToString();

}

//Trigger by an event on the enemy health systyem.

private void EneHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

SetHealth(FanirEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

MySetHealth(FanirHealth.MyGetHealthNormalized());

}

//Trigger by an event on the enemy health systyem.

private void EneHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

SetHealth(FanirEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

MySetHealth(FanirHealth.MyGetHealthNormalized());

}

//Changes the bar image for the enemy's health.

private void SetHealth(float healthNormalized)

{

BarImage.fillAmount = healthNormalized;

}

//Changes the bar image for the our hero health.

private void MySetHealth(float MyhealthNormalized)

{

BarImagemy.fillAmount = MyhealthNormalized;

}

}

## Lagoon\_EneHealth

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Lagoon\_EneHealth : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

private int healthAmount;

private int healthAmountMax;

public int amountValue;

public Lagoon\_EneHealth(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the enemy's health

public void Damage(int amount)

{

LevelSystem levelSystem = new LevelSystem();

//if the enemy's health is less then or equal to 0.

healthAmount -= amount;

amountValue = amount;

//Boss battle trigger//

if (levelSystem.Level >= 14 && healthAmount <= 0)

{

//if the button is beaten//

ButtonNewHero newHero = new ButtonNewHero();

healthAmount = 0;

SceneManager.LoadScene("GameBeat");

ButtonNewHero.LagoonBossBeat = true;

}

else if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("LagoonGameWon");

//LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(100);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the enemy's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns enemy's current health

public float GetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string emeHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Lagoon\_Health

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Lagoon\_Health : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

public event EventHandler OnDeath;

public int healthAmount;

public int amountValue;

private int healthAmountMax;

public Lagoon\_Health(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the enemy's health

public void Damage(int amount)

{

//if the enemy's health is less then or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("LagoonGameOver");

LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(25);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the enemy's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns enemy's current health

public float MyGetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string MyHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Lagoon\_HealthBar

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.Threading.Tasks;

public class Lagoon\_HealthBar : MonoBehaviour

{

//Reference to both heathbar images

private Image BarImage;

private Image BarImagemy;

private Image CircleDamage;

public Text MyHPText;

public Text EmeHPText;

public Text DamageText;

public Text Attack5Text;

public Text Attack6Text;

//Links the two hero classes.

private Lagoon\_EneHealth lagoonEnemyHealth;

private Lagoon\_Health lagoonHealth;

public void Awake()

{

//Initzilse the components to the name before game start//

BarImage = transform.Find("HPBar").GetComponent<Image>();

BarImagemy = transform.Find("MyHPBar").GetComponent<Image>();

CircleDamage = transform.Find("CircleDamage").GetComponent<Image>();

}

void Update()

{

//Update the my health and enemy health change//

MyTextChange();

EmeTextChange();

//This handles unlocking the new attacks name//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 2)

Attack5Text.text = "Tsunami";

else

Attack5Text.text = "Locked";

if (levelSystem.Level >= 9)

Attack6Text.text = "Icicle";

else

Attack6Text.text = "Locked";

}

private void Start()

{

//game starts and hids the square and text

CircleDamage.enabled = false;

DamageText.enabled = false;

//Creates and instance of Fanir stats

HeroLevelStats Lagoon\_States = new HeroLevelStats();

EnemyLevelStats enemyLevelStats = new EnemyLevelStats();

//calls function every time the game start//

Lagoon\_States.LagoonStatsLevel();

enemyLevelStats.EnemyStatsLevel();

//creates an instant of the health for both characters and sets the health//

lagoonEnemyHealth = new Lagoon\_EneHealth(100 + EnemyLevelStats.EnemyKeepHealthStats);

lagoonHealth = new Lagoon\_Health(110 + HeroLevelStats.KeepHealthStats);

//Get the healthNormalized

SetHealth(lagoonEnemyHealth.GetHealthNormalized());

MySetHealth(lagoonHealth.MyGetHealthNormalized());

//Update the health on damage.

lagoonEnemyHealth.OnDamaged += EneHealthSystem\_OnDamaged;

lagoonHealth.OnDamaged += MyHealthSystem\_OnDamaged;

//Update the health on heal.

lagoonEnemyHealth.OnHealed += EneHealthSystem\_OnHealed;

lagoonHealth.OnHealed += MyHealthSystem\_OnHealed;

}

//The hero Heal function. Once the button is pressed, this function will run.

public void HealHero()

{

//Disables the buttons buttonAttackDelay

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

lagoonHealth.Heal(Random.Range(10, 30));

//starts the damagebox function//

StartCoroutine(DamageBox());

//changes the box colour + shows the heal value + calls the delay function//

CircleDamage.color = Color.blue;

ValueToHero();

Delay();

}

//Attack move 1. When button is pressed, call this function.

public void Attack\_1()

{

//Disables button

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

//sents damage to Enemy health//

lagoonEnemyHealth.Damage(5 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 2. When button is pressed, call this function.

public void Attack\_2()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

lagoonEnemyHealth.Damage(10 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 3. When button is pressed, call this function.

public void Attack\_3()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

lagoonEnemyHealth.Damage(15 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 4. When button is pressed, call this function.

public void Attack\_4()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

lagoonEnemyHealth.Damage(10 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

public void Attack\_5()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 3 or geater

if (levelSystem.Level >= 3)

{

//Enables the button///

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

lagoonEnemyHealth.Damage(17 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 3, keep button disabled

transform.Find("Attack5Button").GetComponent<Button>().interactable = false;

}

}

public void Attack\_6()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 9 or geater

if (levelSystem.Level >= 9)

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

lagoonEnemyHealth.Damage(20 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 9, keep button disabled

transform.Find("Attack6Button").GetComponent<Button>().interactable = false;

}

}

public void Delay()

{

//call invoke api and runs the EnemyChoice for 1 second

Invoke("Enemychoice", 1);

//call invoke api and runs the MyTextChange for 1 second

Invoke("MyTextChange", 1);

}

IEnumerator DamageBox()

{

//makes the button appear

CircleDamage.enabled = true;

DamageText.enabled = true;

//waits for 2 second

yield return new WaitForSeconds(2.0f);

//hids the button//

CircleDamage.enabled = false;

DamageText.enabled = false;

}

//This is the where the enemy deals damage to hero health system//

public void Enemychoice()

{

int randomnumnber;

//Generate random number between 1 to 8.

randomnumnber = Random.Range(1, 8);

//If the generate number is 3, the enemy hero would heal.

if (randomnumnber == 3)

{

lagoonEnemyHealth.Heal((30));

CircleDamage.color = Color.yellow;

ValueToEnemy();

EmeTextChange();

}

else

{

//otherwise it would attack.

lagoonHealth.Damage(Random.Range(5, 25) + EnemyLevelStats.EnemyKeepAttackStats - HeroLevelStats.KeepDefenceStats);

CircleDamage.color = Color.red;

ValueToHero();

EmeTextChange();

}

}

//changes the number inside the health bar

public void EmeTextChange()

{

EmeHPText.text = lagoonEnemyHealth.emeHPTextReturn();

}

//changes the number inside the health bar

public void MyTextChange()

{

MyHPText.text = lagoonHealth.MyHPTextReturn();

}

//gets the damage value to hero health

public void ValueToHero()

{

DamageText.text = lagoonHealth.amountValue.ToString();

Debug.Log("amountValue :" + lagoonHealth.amountValue);

}

//gets the damage value to enemy health

public void ValueToEnemy()

{

DamageText.text = lagoonEnemyHealth.amountValue.ToString();

Debug.Log("amountValue :" + lagoonEnemyHealth.amountValue);

}

//Trigger by an event on the enemy health systyem.s

private void EneHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

SetHealth(lagoonEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

MySetHealth(lagoonHealth.MyGetHealthNormalized());

}

//Trigger by an event on the enemy health systyem.

private void EneHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

SetHealth(lagoonEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

MySetHealth(lagoonHealth.MyGetHealthNormalized());

}

//Changes the bar image for the enemy's health.

private void SetHealth(float healthNormalized)

{

BarImage.fillAmount = healthNormalized;

}

//Changes the bar image for the our hero health.

private void MySetHealth(float MyhealthNormalized)

{

BarImagemy.fillAmount = MyhealthNormalized;

}

}

## Panboo\_EneHealth

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Panboo\_EneHealth : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

private int healthAmount;

private int healthAmountMax;

public int amountValue;

public Panboo\_EneHealth(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the enemy's health

public void Damage(int amount)

{

LevelSystem levelSystem = new LevelSystem();

//if the enemy's health is less then or equal to 0.

healthAmount -= amount;

amountValue = amount;

//Boss battle trigger//

if (levelSystem.Level >= 14 && healthAmount <= 0)

{

//if the button is beaten//

ButtonNewHero newHero = new ButtonNewHero();

healthAmount = 0;

SceneManager.LoadScene("GameBeat");

ButtonNewHero.PanbooBossBeat = true;

}

else if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("PanbooGameWon");

levelSystem.AddExperience(100);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the enemy's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns enemy's current health

public float GetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string emeHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Panboo\_Health

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Panboo\_Health : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

public int healthAmount;

private int healthAmountMax;

public int amountValue;

public Panboo\_Health(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the hero's health

public void Damage(int amount)

{

//if the enemy's health is less then or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("PanbooGameOver");

LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(25);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the hero's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns enemy's current health

public float MyGetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string MyHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Panboo\_HealthBar

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.Threading.Tasks;

public class Panboo\_HealthBar : MonoBehaviour

{

//Reference to both heathbar images

private Image BarImage;

private Image BarImagemy;

private Image CircleDamage;

public Text MyHPText;

public Text EmeHPText;

public Text DamageText;

public Text Attack5Text;

public Text Attack6Text;

//Links the two hero classes.

private Panboo\_EneHealth PanbooEnemyHealth;

private Panboo\_Health PanbooHealth;

public void Awake()

{

//Initzilse the components to the name before game start//

BarImage = transform.Find("HPBar").GetComponent<Image>();

BarImagemy = transform.Find("MyHPBar").GetComponent<Image>();

CircleDamage = transform.Find("CircleDamage").GetComponent<Image>();

}

void Update()

{

//Update the my health and enemy health change//

MyTextChange();

EmeTextChange();

//This handles unlocking the new attacks name//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 2)

Attack5Text.text = "Tsunami";

else

Attack5Text.text = "Locked";

if (levelSystem.Level >= 5)

Attack6Text.text = "Icicle";

else

Attack6Text.text = "Locked";

}

private void Start()

{

//game starts and hids the square and text

CircleDamage.enabled = false;

DamageText.enabled = false;

//Creates and instance of Fanir stats

HeroLevelStats Panboo\_Stats = new HeroLevelStats();

EnemyLevelStats enemyLevelStats = new EnemyLevelStats();

//calls function every time the game start//

Panboo\_Stats.PanbooStatsLevel();

enemyLevelStats.EnemyStatsLevel();

//creates an instant of the health for both characters and sets the health//

PanbooEnemyHealth = new Panboo\_EneHealth(100 + EnemyLevelStats.EnemyKeepHealthStats);

PanbooHealth = new Panboo\_Health(100 + HeroLevelStats.KeepHealthStats);

//Get the healthNormalized

SetHealth(PanbooEnemyHealth.GetHealthNormalized());

MySetHealth(PanbooHealth.MyGetHealthNormalized());

//Update the health on damage.

PanbooEnemyHealth.OnDamaged += EneHealthSystem\_OnDamaged;

PanbooHealth.OnDamaged += MyHealthSystem\_OnDamaged;

//Update the health on heal.

PanbooEnemyHealth.OnHealed += EneHealthSystem\_OnHealed;

PanbooHealth.OnHealed += MyHealthSystem\_OnHealed;

}

//The hero Heal function. Once the button is pressed, this function will run.

public void HealHero()

{

//Disables the buttons buttonAttackDelay

ButtonDelay buttonDelay = new ButtonDelay();

//Random Value goes into heal functon for Panboo//

StartCoroutine(buttonDelay.ButtonAttackDelay());

PanbooHealth.Heal(Random.Range(10, 30));

//starts the damagebox function//

StartCoroutine(DamageBox());

//changes the box colour + shows the heal value + calls the delay function//

CircleDamage.color = Color.blue;

ValueToHero();

Delay();

}

//Attack move 1. When button is pressed, call this function.

public void Attack\_1()

{

//Disables button

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

//sents damage to Enemy health//

PanbooEnemyHealth.Damage(5 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 2. When button is pressed, call this function.

public void Attack\_2()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

PanbooEnemyHealth.Damage(10 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 3. When button is pressed, call this function.

public void Attack\_3()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

PanbooEnemyHealth.Damage(15 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 4. When button is pressed, call this function.

public void Attack\_4()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

PanbooEnemyHealth.Damage(10 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

public void Attack\_5()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 3 or geater

if (levelSystem.Level >= 3)

{

//Enables the button///

transform.Find("Attack5Button").GetComponent<Button>().interactable = true;

//same as normal attack//

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

PanbooEnemyHealth.Damage(17 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 3, keep button disabled

transform.Find("Attack5Button").GetComponent<Button>().interactable = false;

}

}

public void Attack\_6()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 9 or geater

if (levelSystem.Level >= 9)

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

PanbooEnemyHealth.Damage(20 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 9, keep button disabled

transform.Find("Attack6Button").GetComponent<Button>().interactable = false;

}

}

public void Delay()

{

//call invoke api and runs the EnemyChoice for 1 second

Invoke("Enemychoice", 1);

//call invoke api and runs the MyTextChange for 1 second

Invoke("MyTextChange", 1);

}

IEnumerator DamageBox()

{

//makes the button appear

CircleDamage.enabled = true;

DamageText.enabled = true;

//waits for 2 second

yield return new WaitForSeconds(2.0f);

//hids the button//

CircleDamage.enabled = false;

DamageText.enabled = false;

}

//This is the where the enemy deals damage to hero health system//

public void Enemychoice()

{

int randomnumnber;

//Generate random number between 1 to 8.

randomnumnber = Random.Range(1, 8);

//If the generate number is 3, the enemy hero would heal.

if (randomnumnber == 3)

{

PanbooEnemyHealth.Heal((30));

CircleDamage.color = Color.yellow;

ValueToEnemy();

EmeTextChange();

}

else

{

//otherwise it would attack.

PanbooHealth.Damage(Random.Range(5, 20) + EnemyLevelStats.EnemyKeepAttackStats - HeroLevelStats.KeepDefenceStats);

CircleDamage.color = Color.red;

ValueToHero();

EmeTextChange();

}

}

//changes the number inside the health bar

public void EmeTextChange()

{

EmeHPText.text = PanbooEnemyHealth.emeHPTextReturn();

}

//changes the number inside the health bar

public void MyTextChange()

{

MyHPText.text = PanbooHealth.MyHPTextReturn();

}

//gets the damage value to hero health

public void ValueToHero()

{

DamageText.text = PanbooHealth.amountValue.ToString();

}

//gets the damage value to enemy health

public void ValueToEnemy()

{

DamageText.text = PanbooEnemyHealth.amountValue.ToString();

}

//Trigger by an event on the enemy health systyem.s

private void EneHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

SetHealth(PanbooEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

MySetHealth(PanbooHealth.MyGetHealthNormalized());

}

//Trigger by an event on the enemy health systyem.

private void EneHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

SetHealth(PanbooEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

MySetHealth(PanbooHealth.MyGetHealthNormalized());

}

//Changes the bar image for the enemy's health.

private void SetHealth(float healthNormalized)

{

BarImage.fillAmount = healthNormalized;

}

//Changes the bar image for the our hero health.

private void MySetHealth(float MyhealthNormalized)

{

BarImagemy.fillAmount = MyhealthNormalized;

}

}

## Secret\_EneHealth

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Secret\_EneHealth : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

private int healthAmount;

private int healthAmountMax;

public int amountValue;

public Secret\_EneHealth(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the enemy's health

public void Damage(int amount)

{

LevelSystem levelSystem = new LevelSystem();

//if the enemy's health is less then or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("SecretHeroWin");

//LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(100);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the enemy's health

public void Heal(int amount)

{

healthAmount += amount;

amountValue = amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns enemy's current health

public float GetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string emeHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Secret\_Health

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class Secret\_Health : MonoBehaviour

{

//Events notofiys when the health changes.

public event EventHandler OnDamaged;

public event EventHandler OnHealed;

public int healthAmount;

private int healthAmountMax;

public int amountValue;

public Secret\_Health(int healthAmount)

{

healthAmountMax = healthAmount;

this.healthAmount = healthAmount;

}

//This function damages the hero's health

public void Damage(int amount)

{

//if the heros's health is less then or equal to 0.

healthAmount -= amount;

amountValue = amount;

if (healthAmount <= 0)

{

//Load the game won scene.

healthAmount = 0;

SceneManager.LoadScene("SecretHeroLose");

LevelSystem levelSystem = new LevelSystem();

levelSystem.AddExperience(25);

levelSystem.ReturnExpText();

}

if (OnDamaged != null)

{

OnDamaged(this, EventArgs.Empty);

}

}

//This function heals the enemy's health

public void Heal(int amount)

{

amountValue = amount;

healthAmount += amount;

if (healthAmount > healthAmountMax)

{

healthAmount = healthAmountMax;

}

if (OnHealed != null)

{

OnHealed(this, EventArgs.Empty);

}

}

//Function returns hero's current health

public float MyGetHealthNormalized()

{

return (float)healthAmount / healthAmountMax;

}

//returns the current health out of max health

public string MyHPTextReturn()

{

return healthAmount + "/" + healthAmountMax;

}

}

## Secret\_HealthBar

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.Threading.Tasks;

public class Secret\_HealthBar : MonoBehaviour

{

//Reference to both heathbar images

private Image BarImage;

private Image BarImagemy;

private Image CircleDamage;

public Text MyHPText;

public Text EmeHPText;

public Text DamageText;

public Text Attack5Text;

public Text Attack6Text;

//Links the two hero classes.

private Secret\_EneHealth SecretEnemyHealth;

private Secret\_Health SecretHealth;

public void Awake()

{

//Initzilse the components to the name before game start//

BarImage = transform.Find("HPBar").GetComponent<Image>();

BarImagemy = transform.Find("MyHPBar").GetComponent<Image>();

CircleDamage = transform.Find("CircleDamage").GetComponent<Image>();

}

void Update()

{

//Update the my health and enemy health change//

MyTextChange();

EmeTextChange();

//This handles unlocking the new attacks name//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 2)

Attack5Text.text = "Tidal Wave";

else

Attack5Text.text = "Locked";

if (levelSystem.Level >= 5)

Attack6Text.text = "Fallout";

else

Attack6Text.text = "Locked";

}

private void Start()

{

//game starts and hids the square and text

CircleDamage.enabled = false;

DamageText.enabled = false;

//Creates and instance of Fanir stats

HeroLevelStats Fanir\_States = new HeroLevelStats();

EnemyLevelStats enemyLevelStats = new EnemyLevelStats();

//calls function every time the game start//

Fanir\_States.FanirStatsLevel();

enemyLevelStats.EnemyStatsLevel();

//creates an instant of the health for both characters and sets the health//

SecretEnemyHealth = new Secret\_EneHealth(100 + EnemyLevelStats.EnemyKeepHealthStats);

SecretHealth = new Secret\_Health(110 + HeroLevelStats.KeepHealthStats);

//Get the healthNormalized

SetHealth(SecretEnemyHealth.GetHealthNormalized());

MySetHealth(SecretHealth.MyGetHealthNormalized());

//Update the health on damage.

SecretEnemyHealth.OnDamaged += EneHealthSystem\_OnDamaged;

SecretHealth.OnDamaged += MyHealthSystem\_OnDamaged;

//Update the health on heal.

SecretEnemyHealth.OnHealed += EneHealthSystem\_OnHealed;

SecretHealth.OnHealed += MyHealthSystem\_OnHealed;

}

//The hero Heal function. Once the button is pressed, this function will run.

public void HealHero()

{

//Disables the buttons buttonAttackDelay

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

//Random Value goes into heal functon for hero//

SecretHealth.Heal(Random.Range(10, 30));

//starts the damagebox function//

StartCoroutine(DamageBox());

//changes the box colour + shows the heal value + calls the delay function//

CircleDamage.color = Color.blue;

ValueToHero();

Delay();

}

//Attack move 1. When button is pressed, call this function.

public void Attack\_1()

{

//Disables button

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

//sents damage to Enemy health//

SecretEnemyHealth.Damage(7 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 2. When button is pressed, call this function.

public void Attack\_2()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

SecretEnemyHealth.Damage(12 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 3. When button is pressed, call this function.

public void Attack\_3()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

SecretEnemyHealth.Damage(17 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

//Attack move 4. When button is pressed, call this function.

public void Attack\_4()

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

SecretEnemyHealth.Damage(12 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

public void Attack\_5()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 3 or geater

if (levelSystem.Level >= 3)

{

//Enables the button///

transform.Find("Attack5Button").GetComponent<Button>().interactable = true;

//same as normal attack//

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

SecretEnemyHealth.Damage(17 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{

//if level is lower then 3, keep button disabled

transform.Find("Attack5Button").GetComponent<Button>().interactable = false;

}

}

public void Attack\_6()

{

LevelSystem levelSystem = new LevelSystem();

//checks if level is 9 or geater

if (levelSystem.Level >= 9)

{

ButtonDelay buttonDelay = new ButtonDelay();

StartCoroutine(buttonDelay.ButtonAttackDelay());

StartCoroutine(DamageBox());

SecretEnemyHealth.Damage(20 + HeroLevelStats.KeepAttackStats);

CircleDamage.color = Color.green;

ValueToEnemy();

Delay();

}

else

{ //if level is lower then 9, keep button disabled

transform.Find("Attack6Button").GetComponent<Button>().interactable = false;

}

}

public void Delay()

{

//call invoke api and runs the EnemyChoice for 1 second

Invoke("Enemychoice", 1);

//call invoke api and runs the MyTextChange for 1 second

Invoke("MyTextChange", 1);

}

IEnumerator DamageBox()

{

//makes the button appear

CircleDamage.enabled = true;

DamageText.enabled = true;

//waits for 2 second

yield return new WaitForSeconds(2.0f);

//hids the button//

CircleDamage.enabled = false;

DamageText.enabled = false;

}

//This is the where the enemy deals damage to hero health system//

public void Enemychoice()

{

int randomnumnber;

//Generate random number between 1 to 8.

randomnumnber = Random.Range(1, 8);

//If the generate number is 3, the enemy hero would heal.

if (randomnumnber == 3)

{

SecretEnemyHealth.Heal((30));

CircleDamage.color = Color.yellow;

ValueToEnemy();

EmeTextChange();

}

else

{

//otherwise it would attack.

SecretHealth.Damage(Random.Range(5, 25) + EnemyLevelStats.EnemyKeepAttackStats - HeroLevelStats.KeepDefenceStats);

CircleDamage.color = Color.red;

ValueToHero();

EmeTextChange();

}

}

//changes the number inside the health bar

public void EmeTextChange()

{

EmeHPText.text = SecretEnemyHealth.emeHPTextReturn();

}

//changes the number inside the health bar

public void MyTextChange()

{

MyHPText.text = SecretHealth.MyHPTextReturn();

}

//gets the damage value to hero health

public void ValueToHero()

{

DamageText.text = SecretHealth.amountValue.ToString();

}

//gets the damage value to enemy health

public void ValueToEnemy()

{

DamageText.text = SecretEnemyHealth.amountValue.ToString();

}

//Trigger by an event on the enemy health systyem.s

private void EneHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

SetHealth(SecretEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnDamaged(object sender, System.EventArgs e)

{

MySetHealth(SecretHealth.MyGetHealthNormalized());

}

//Trigger by an event on the enemy health systyem.

private void EneHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

SetHealth(SecretEnemyHealth.GetHealthNormalized());

}

//Trigger by an event on the our health systyem.

private void MyHealthSystem\_OnHealed(object sender, System.EventArgs e)

{

MySetHealth(SecretHealth.MyGetHealthNormalized());

}

//Changes the bar image for the enemy's health.

private void SetHealth(float healthNormalized)

{

BarImage.fillAmount = healthNormalized;

}

//Changes the bar image for the our hero health.

private void MySetHealth(float MyhealthNormalized)

{

BarImagemy.fillAmount = MyhealthNormalized;

}

}

## DisplayLevel

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class DisplayLevel : MonoBehaviour

{

private Text ExpText;

private Text LevelText;

private Image ExpBar;

private LevelSystem levelSystem;

private void Awake()

{

ExpText = transform.Find("ExpText").GetComponent<Text>();

LevelText = transform.Find("Level").GetComponent<Text>();

ExpBar = transform.Find("ExpBar").GetComponent<Image>();

LevelSystem levelSystem = new LevelSystem();

SetLevelSystem(levelSystem);

}

void Update()

{

//Update the experince

MyExpTextChange();

}

//dispays the level//

private void SetLevelNumber (int LevelNumber)

{

LevelText.text = "Level : " + LevelNumber;

}

//gets the current experience out of max experience//

public void MyExpTextChange()

{

ExpText.text = levelSystem.ReturnExpText();

}

private void LevelSystem\_OnLevelChange(object sender, System.EventArgs e)

{

//Level change, update text//

SetLevelNumber(levelSystem.GetLevelNumber());

}

private void LevelSystem\_OnExperienceChange(object sender, System.EventArgs e)

{

//Experience change, update Bar Size//

SetExperienceBarSize(levelSystem.GetExperienceNormalized());

}

//fills the experience bar to the current experience//

private void SetExperienceBarSize(float ExperienceNormalized)

{

ExpBar.fillAmount = ExperienceNormalized;

}

}

## LevelSystem

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class LevelSystem : MonoBehaviour

{

// Start is called before the first frame update

public event EventHandler OnExperienceChange;

public event EventHandler OnLevelChange;

public int Level = KeepLevel;

public int Experience = KeepExp;

private int ExperienceNextLevel = 100;

public static int KeepExp = 0;

public static int KeepLevel = 1;

//this function adds the experince to the hero.

public void AddExperience(int amount)

{

//depends on what the outcome out the battle is, differnt amount value//

//adds the amount to experience//

Experience += amount;

//if the experince is greater then max experience

if (Experience >= ExperienceNextLevel)

{

//add level//

Level++;

//subtract max experience from current experince//

Experience -= ExperienceNextLevel;

if (OnLevelChange != null) OnLevelChange(this, EventArgs.Empty);

KeepLevel = Level;

}

if (OnExperienceChange != null) OnExperienceChange(this, EventArgs.Empty);

KeepExp = Experience;

}

//return the level

public int GetLevelNumber()

{

return Level;

}

//Gets the current experince

public float GetExperienceNormalized()

{

return (float)Experience / ExperienceNextLevel;

}

//return experince text to display//

public string ReturnExpText()

{

return KeepExp + "/" + ExperienceNextLevel;

}

}

## Fanir\_EnemySpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class FanirEnemySpawner : MonoBehaviour

{

private int rand;

public Sprite[] Spawner\_Pic;

//arrray of names//

private string[] Name = { "Wallabyss", "Toalow", "Penguna", "Mantitar", "Kineling", "Magmelope", "Wintora", "Dracaza", "Quackal", "Staropotamus", "Territe", "Super Fanir" };

public Text EnemyName;

public Text EnemyLevel;

// Start is called before the first frame update

void Start()

{

Change();

LevelSystem levelSystem = new LevelSystem();

EnemyLevel.text = "Level : " + levelSystem.GetLevelNumber();

}

void Change()

{

LevelSystem levelSystem = new LevelSystem();

//Gets a random number with in 0 to the Srpite array Length

rand = Random.Range(0, 9);

//Displays the sprite that was selected by random

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[rand];

//Displays the name that is accesed with the sprite

EnemyName.text = " Name : " + Name[rand];

//boss sprite//

if (levelSystem.Level >= 2)

{

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[10];

EnemyName.text = " Name : " + Name[11];

}

}

}

## FanirSpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class FanirSpawner : MonoBehaviour

{

public Sprite[] Spawner\_Pic;

//array of names

private string[] Namelist = { "Fanir", "Super Fanir", "lagoon", "Super Lagoon"};

public Text Name;

// Start is called before the first frame update

void Start()

{

FanirSpawn();

}

void FanirSpawn()

{

//checks if player beat game with fanir//

if (ButtonNewHero.FanirBossBeat)

{

//is yes, then spawn super fanir

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[1];

Name.text = " Name : " + Namelist[1];

}

else

{

//if not then spawn normal fanir

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[0];

Name.text = " Name : " + Namelist[0];

}

}

}

## Lagoon\_EnemySpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class LagoonEnemySpawner : MonoBehaviour

{

private int rand;

public Sprite[] Spawner\_Pic;

private string[] Name = { "Crosaur", "Elanyte", "Hippopoke", "Pandoke", "Chimesel", "Whirlkey", "Boneleon", "Goldingale", "Quackal", "Drummingbird", "Frogre", "Super Lagoon"};

public Text EnemyName;

public Text EnemyLevel;

private void Awake()

{

EnemyName = transform.Find("EnemyName").GetComponent<Text>();

EnemyLevel = transform.Find("EnemyLevel").GetComponent<Text>();

}

// Start is called before the first frame update

void Start()

{

Change();

LevelSystem levelSystem = new LevelSystem();

EnemyLevel.text = "Level : " + levelSystem.GetLevelNumber();

}

void Change()

{

LevelSystem levelSystem = new LevelSystem();

//Gets a random number with in 0 to the Srpite array Length

rand = Random.Range(0,9);

//Displays the sprite that was selected by random

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[rand];

//Displays the name that is accesed with the sprite

EnemyName.text = " Name : " + Name[rand];

if(levelSystem.Level >= 5)

{

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[10];

EnemyName.text = " Name : " + Name[11];

EnemyLevel.text = "Level : " + (levelSystem.Level+1);

}

}

}

## LagoonSpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class LagoonSpawner : MonoBehaviour

{

public Sprite[] Spawner\_Pic;

//array of names

private string[] Namelist = { "Lagoon", "Super Lagoon"};

public Text Name;

// Start is called before the first frame update

void Start()

{

LagoonSpawn();

}

void LagoonSpawn()

{

//checks if player beat game with lag0on//

if (ButtonNewHero.FanirBossBeat)

{

//is yes, then spawn super lagoon

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[1];

Name.text = " Name : " + Namelist[1];

}

else

{

//if not then spawn normal lagoon

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[0];

Name.text = " Name : " + Namelist[0];

}

}

}

## Panboo\_EnemySpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class PanbooEnemySpawner: MonoBehaviour

{

private int rand;

public Sprite[] Spawner\_Pic;

//array of names

private string[] Name = { "Toatone", "Croros", "Crazekey", "Ramparak", "Lobsteroid", "Pandyle", "Waroda", "Chickombo", "Dradrill", "Hunteleon", "Mermantis", "Super Panboo" };

public Text EnemyName;

public Text EnemyLevel;

private void Awake()

{

EnemyName = transform.Find("EnemyName").GetComponent<Text>();

EnemyLevel = transform.Find("EnemyLevel").GetComponent<Text>();

}

// Start is called before the first frame update

void Start()

{

Change();

LevelSystem levelSystem = new LevelSystem();

EnemyLevel.text = "Level : " + levelSystem.GetLevelNumber();

}

void Change()

{

LevelSystem levelSystem = new LevelSystem();

//Gets a random number with in 0 to the Srpite array Length

rand = Random.Range(0, 9);

//Displays the sprite that was selected by random

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[rand];

//Displays the name that is accesed with the sprite

EnemyName.text = " Name : " + Name[rand];

if (levelSystem.Level >= 5)

{

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[10];

EnemyName.text = " Name : " + Name[11];

EnemyLevel.text = "Level : " + (levelSystem.Level + 1);

}

}

}

## PanbooSpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class PanbooSpawner : MonoBehaviour

{

public Sprite[] Spawner\_Pic;

//array of names

private string[] Namelist = { "Panboo", "Super Panboo" };

public Text Name;

// Start is called before the first frame update

void Start()

{

PanbooSpawn();

}

void PanbooSpawn()

{

//checks if player beat game with panboo//

if (ButtonNewHero.FanirBossBeat)

{

//is yes, then spawn super panboo

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[1];

Name.text = " Name : " + Namelist[1];

}

else

{

//if not then spawn normal panboo

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[0];

Name.text = " Name : " + Namelist[0];

}

}

}

## SecretEnemySpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class SecretEnemySpawner : MonoBehaviour

{

private int rand;

public Sprite[] Spawner\_Pic;

//array of names//

private string[] Name = { "Wallabyss", "Toalow", "Penguna", "Mantitar", "Kineling", "Magmelope", "Wintora", "Dracaza", "Quackal", "Staropotamus", "Territe", "Super Fanir",

"Toatone", "Croros", "Crazekey", "Ramparak", "Lobsteroid", "Pandyle", "Waroda", "Chickombo", "Dradrill", "Hunteleon", "Mermantis", "Super Panboo",

"Crosaur", "Elanyte", "Hippopoke", "Pandoke", "Chimesel", "Whirlkey", "Boneleon", "Goldingale", "Quackal", "Drummingbird", "Frogre", "Scorpilite",

"Crofree", "Vultune", "Riotter" ,"Super Lagoon"};

public Text EnemyName;

public Text EnemyLevel;

private void Awake()

{

EnemyName = transform.Find("EnemyName").GetComponent<Text>();

EnemyLevel = transform.Find("EnemyLevel").GetComponent<Text>();

}

// Start is called before the first frame update

void Start()

{

Change();

LevelSystem levelSystem = new LevelSystem();

EnemyLevel.text = "Level : " + levelSystem.GetLevelNumber();

}

void Change()

{

LevelSystem levelSystem = new LevelSystem();

//Gets a random number with in 0 to the Srpite array Length

rand = Random.Range(0, 39);

//Displays the sprite that was selected by random

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[rand];

//Displays the name that is accesed with the sprite

EnemyName.text = " Name : " + Name[rand];

}

}

## SecretSpawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class SecretSpawner : MonoBehaviour

{

public Sprite[] Spawner\_Pic;

private string[] Namelist = { "Ray" };

public Text Name;

// Start is called before the first frame update

void Start()

{

RaySpawn();

}

void RaySpawn()

{

//spawn new hero//

GetComponent<SpriteRenderer>().sprite = Spawner\_Pic[0];

Name.text = " Name : " + Namelist[0];

}

}

## EnemyLevelStats

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyLevelStats : MonoBehaviour

{

private int EnemyHealthIncreaseLevel = EnemyKeepHealthStats;

public static int EnemyKeepHealthStats = 0;

private int EnemyAttackIncreaseLevel = EnemyKeepAttackStats;

public static int EnemyKeepAttackStats = 0;

void Start()

{

EnemyStatsLevel();

}

//Fanir enemy stat boost

public void EnemyStatsLevel()

{

//super fanir stats boost

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level >= 14s)

{

//gets the health and attack and appliys it to itown//

HeroLevelStats heroLevelStats = new HeroLevelStats();

EnemyHealthIncreaseLevel = heroLevelStats.HealthIncreaseLevel \* 2;

EnemyKeepHealthStats = EnemyHealthIncreaseLevel;

EnemyAttackIncreaseLevel = heroLevelStats.AttackIncreaseLevel \* 2;

EnemyKeepAttackStats = EnemyAttackIncreaseLevel;

}

else if (levelSystem.Level % 2 == 0)

{

//otherwise normal stats boost//

EnemyHealthIncreaseLevel += Random.Range(5, 15);

EnemyKeepHealthStats = EnemyHealthIncreaseLevel;

EnemyAttackIncreaseLevel += Random.Range(2, 5);

EnemyKeepAttackStats = EnemyAttackIncreaseLevel;

}

else

{

//otherwise no stats boost

EnemyHealthIncreaseLevel = EnemyKeepHealthStats;

EnemyAttackIncreaseLevel = EnemyKeepAttackStats;

}

}

}

## HeroLevelStats

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class HeroLevelStats : MonoBehaviour

{

public int HealthIncreaseLevel = KeepHealthStats;

public static int KeepHealthStats = 0;

public int AttackIncreaseLevel = KeepAttackStats;

public static int KeepAttackStats = 0;

public int DefenceIncreaseLevel = KeepDefenceStats;

public static int KeepDefenceStats = 0;

// Start is called before the first frame update

void Start()

{

FanirStatsLevel();

LagoonStatsLevel();

PanbooStatsLevel();

}

//Fanir stats boost

public void FanirStatsLevel()

{

//Every 3 level get more attack boost//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level % 3 == 0)

{

AttackIncreaseLevel += 7;

KeepAttackStats = AttackIncreaseLevel;

}

//every 2 level gets defence and health boost//

if(levelSystem.Level % 2 == 0)

{

HealthIncreaseLevel += Random.Range(3, 8);

KeepHealthStats = HealthIncreaseLevel;

DefenceIncreaseLevel += Random.Range(3, 5);

KeepDefenceStats = DefenceIncreaseLevel;

}

else

{

//otherwise no stats boost//

AttackIncreaseLevel = KeepAttackStats;

HealthIncreaseLevel = KeepHealthStats;

DefenceIncreaseLevel = KeepDefenceStats;

}

}

//lagoon stats boost

public void LagoonStatsLevel()

{

//Every 3 level get more attack boost//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level % 3 == 0)

{

AttackIncreaseLevel += Random.Range(3, 5);

KeepAttackStats = AttackIncreaseLevel;

}

//every 2 level gets defence and health boost//

if (levelSystem.Level % 2 == 0)

{

HealthIncreaseLevel += 10;

KeepHealthStats = HealthIncreaseLevel;

DefenceIncreaseLevel += Random.Range(3, 5);

KeepDefenceStats = DefenceIncreaseLevel;

}

else

{

//otherwise no stats boost//

AttackIncreaseLevel = KeepAttackStats;

HealthIncreaseLevel = KeepHealthStats;

DefenceIncreaseLevel = KeepDefenceStats;

}

}

//panboo stats boost

public void PanbooStatsLevel()

{

//Every 3 level get more attack boost//

LevelSystem levelSystem = new LevelSystem();

if (levelSystem.Level % 3 == 0)

{

AttackIncreaseLevel += Random.Range(3, 5);

KeepAttackStats = AttackIncreaseLevel;

}

//every 2 level gets defence and health boost//

if (levelSystem.Level % 2 == 0)

{

HealthIncreaseLevel += Random.Range(3, 5);

KeepHealthStats = HealthIncreaseLevel;

DefenceIncreaseLevel += 10;

KeepDefenceStats = DefenceIncreaseLevel;

}

else

{

//otherwise no stats boost//

AttackIncreaseLevel = KeepAttackStats;

HealthIncreaseLevel = KeepHealthStats;

DefenceIncreaseLevel = KeepDefenceStats;

}

}

}

## ButtonDelay

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class ButtonDelay : MonoBehaviour

{

private Button Attack1;

private Button Attack2;

private Button Attack3;

private Button Attack4;

private Button Attack5;

private Button Attack6;

private Button Heal;

void Awake()

{

//Gets the reference//

Attack1 = GetComponent<Button>();

Attack2 = GetComponent<Button>();

Attack3 = GetComponent<Button>();

Attack4 = GetComponent<Button>();

Attack5 = GetComponent<Button>();

Attack6 = GetComponent<Button>();

Heal = GetComponent<Button>();

}

public IEnumerator ButtonAttackDelay()

{

//Disables all the buttons//

Attack1 = GameObject.Find("Attack1Button").GetComponent<Button>();

Attack1.interactable = false;

Attack2 = GameObject.Find("Attack2Button").GetComponent<Button>();

Attack2.interactable = false;

Attack3 = GameObject.Find("Attack3Button").GetComponent<Button>();

Attack3.interactable = false;

Attack4 = GameObject.Find("Attack4Button").GetComponent<Button>();

Attack4.interactable = false;

Attack5 = GameObject.Find("Attack5Button").GetComponent<Button>();

Attack5.interactable = false;

Attack6 = GameObject.Find("Attack6Button").GetComponent<Button>();

Attack6.interactable = false;

Heal = GameObject.Find("HealButton").GetComponent<Button>();

Heal.interactable = false;

//Wait 2 second//

yield return new WaitForSeconds(2f);

//Enable all button//

Attack1 = GameObject.Find("Attack1Button").GetComponent<Button>();

Attack1.interactable = true;

Attack2 = GameObject.Find("Attack2Button").GetComponent<Button>();

Attack2.interactable = true;

Attack3 = GameObject.Find("Attack3Button").GetComponent<Button>();

Attack3.interactable = true;

Attack4 = GameObject.Find("Attack4Button").GetComponent<Button>();

Attack4.interactable = true;

Attack5 = GameObject.Find("Attack5Button").GetComponent<Button>();

Attack5.interactable = true;

Attack6 = GameObject.Find("Attack6Button").GetComponent<Button>();

Attack6.interactable = true;

Heal = GameObject.Find("HealButton").GetComponent<Button>();

Heal.interactable = true;

}

}

## ButtonNewHero

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class ButtonNewHero : MonoBehaviour

{

public Button NewHero;

public static bool FanirBossBeat;

public static bool LagoonBossBeat;

public static bool PanbooBossBeat;

private void Start()

{

//if all the bosses are beat, secret her button appears otherwise it hidden

if (FanirBossBeat && LagoonBossBeat && PanbooBossBeat)

{

NewHero.gameObject.SetActive(true);

}

else

{

NewHero.gameObject.SetActive(false);

}

}

}

## MainMen

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

public Text newText;

public void PlayGame ()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

////////////////////////Starters///////////////////////////////////////

public void FanirStarter()

{

SceneManager.LoadScene("FanirScene");

}

public void LagoonStarer()

{

SceneManager.LoadScene("LagoonScene");

}

public void PanbooStarter()

{

SceneManager.LoadScene("PanbooScene");

}

public void SecretHero()

{

SceneManager.LoadScene("SecretHeroScenes");

}

////////////////////////Fanir///////////////////////////////////////

public void Greenwing\_GameWon()

{

SceneManager.LoadScene("FanirGameWon");

}

public void Greenwing\_GameOver()

{

SceneManager.LoadScene("FanirGameOver");

}

////////////////////////Lagoon///////////////////////////////////////

public void Lagoon\_GameOver()

{

SceneManager.LoadScene("LagoonGameOver");

}

public void Lagoon\_GameWon()

{

SceneManager.LoadScene("LagoonGameWon");

}

////////////////////////Panboo///////////////////////////////////////

public void Panboo\_GameOver()

{

SceneManager.LoadScene("PanbooGameOver");

}

public void Panboo\_GameWon()

{

SceneManager.LoadScene("PanbooGameWon");

}

////////////////////////Secret Hero///////////////////////////////////////

public void SecretHero\_GameOver()

{

SceneManager.LoadScene("SecretHeroLose");

}

public void SecretHero\_GameWon()

{

SceneManager.LoadScene("SecretHeroWin");

}

//Reset Everything once main menu button clicked.

public void Main\_Menu()

{

SceneManager.LoadScene("SampleScene");

LevelSystem.KeepLevel = 1;

LevelSystem.KeepExp = 0;

HeroLevelStats.KeepAttackStats = 0;

HeroLevelStats.KeepDefenceStats = 0;

HeroLevelStats.KeepHealthStats = 0;

EnemyLevelStats.EnemyKeepAttackStats = 0;

EnemyLevelStats.EnemyKeepHealthStats = 0;

}

public void QuitGame ()

{

Debug.Log("Quiting");

Application.Quit();

}

}