Rune Raid Report Part Three

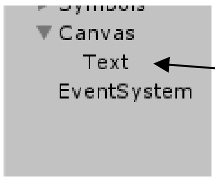
Contents

Development Part 3

Timer

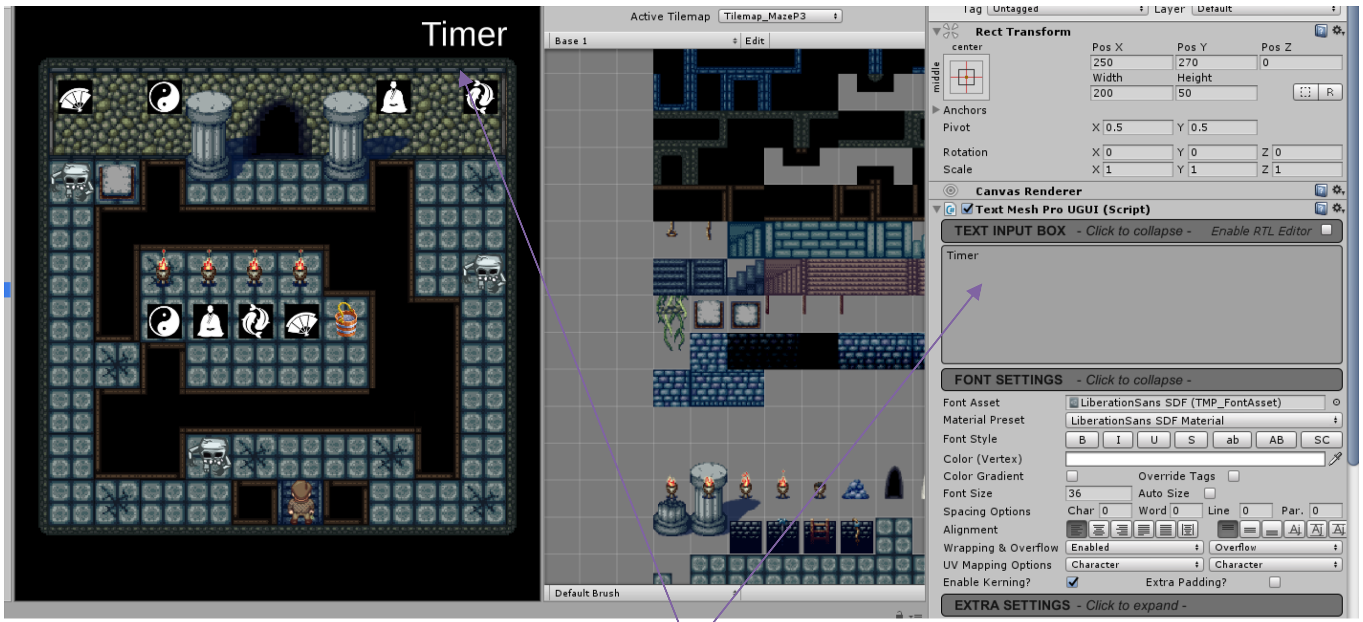
Now that I've developed all three levels, besides the Timer, Runes and story/text boxes, I next went and started to develop the Timer as the Rune requires the Timer and the Text boxes should be formatted after the Time creation.

I did some research on this best way to create a timer system and I found a YouTube video which made making a timer incredibly easy and simple to implement. The way to implement a timer is to utilise the build in Text UI to change its value each second. [1]



```
▸ Symbols
▼ Canvas
    Text
  EventSystem
```

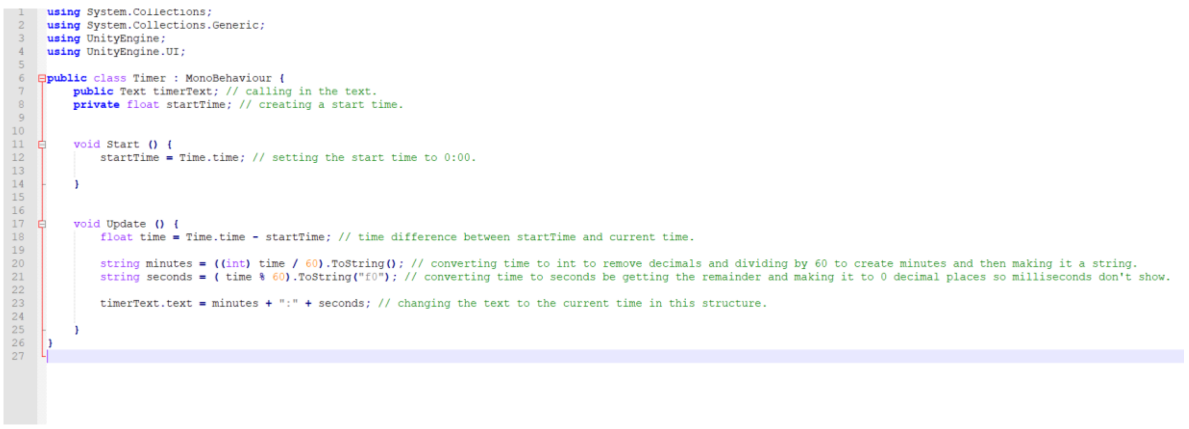
First, is to create a Text UI GameObject.



```
Tag  Untagged            Layer  Default
Rect Transform
  center    Pos X   Pos Y   Pos Z
  middle    250     270     0
            Width   Height
            200     50
  Anchors
  Pivot     X 0.5   Y 0.5
  Rotation  X 0     Y 0     Z 0
  Scale     X 1     Y 1     Z 1
Canvas Renderer
Text Mesh Pro UGUI (Script)
  TEXT INPUT BOX  - Click to collapse -   Enable RTL Editor
  Timer
  FONT SETTINGS  - Click to collapse -
  Font Asset           LiberationSans SDF (TMP_FontAsset)
  Material Preset      LiberationSans SDF Material
  Font Style           B  I  U  S  ab  AB  SC
  Color (Vertex)
  Color Gradient       ☐        Override Tags ☐
  Font Size            36       Auto Size ☐
  Spacing Options      Char 0   Word 0   Line 0   Par. 0
  Alignment
  Wrapping & Overflow  Enabled     Overflow
  UV Mapping Options   Character   Character
  Enable Kerning?      ☑        Extra Padding? ☐
  EXTRA SETTINGS  - Click to expand -
```

Then I adjusted its position and put "Timer" in the text box, this has no other reason but just for an easier understanding.

[1] https://www.youtube.com/watch?v=x-C95TuQtf0 – Date accessed 10/04/18 – Timer YouTube video



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class Timer : MonoBehaviour {
7      public Text timerText; // calling in the text.
8      private float startTime; // creating a start time.
9
10
11     void Start () {
12         startTime = Time.time; // setting the start time to 0:00.
13
14     }
15
16
17     void Update () {
18         float time = Time.time - startTime; // time difference between startTime and current time.
19
20         string minutes = ((int) time / 60).ToString(); // converting time to int to remove decimals and dividing by 60 to create minutes and then making it a string.
21         string seconds = ( time % 60).ToString("f0"); // converting time to seconds be getting the remainder and making it to 0 decimal places so milliseconds don't show.
22
23         timerText.text = minutes + ":" + seconds; // changing the text to the current time in this structure.
24
25     }
26 }
27
```

This is the script that the YouTube video told me to create which will make a simple Timer system. It works by setting the float variable "startTime" to equal "Time.time" which is a built in function that is the current time since the script executes, therefore in the start function, the "startTime" will equal 0.

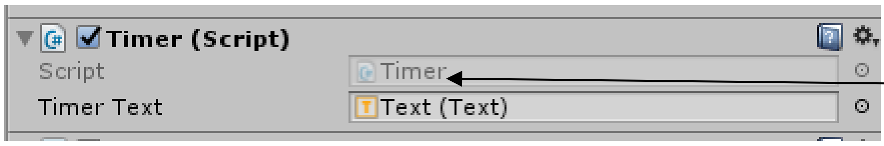
Next, in the Update function, the float "time" is set to the (Time.time – startTime) this is the difference between the two variables.

Line 20 creates a local string variable called "minutes" which is set to equal the integer of "time" then divided by 60 and then converted to a string. This is to remove all decimals from "time" and then divide it by 60 to convert it to minute form. For example, time 130 will be divided by 60 which is 2.1666… and then set to an integer which is just 2.

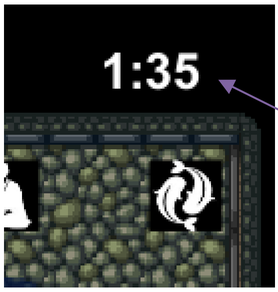
Line 21 creates a local string variable called "seconds" which is set to equal (time mod 60) and then converted to a string. The (time mod 60) is working out the remainder of the "time" out of 60. For example, time 130 will be remainder 10 which is the seconds.]#

Then line 23 sets the timerText.text to equal (minutes + ":" + seconds) this is the format that time is represented in. the "timerText.text" is the parameter for the Timer text GameObject.

So overall, this script changes the Timer text every second.

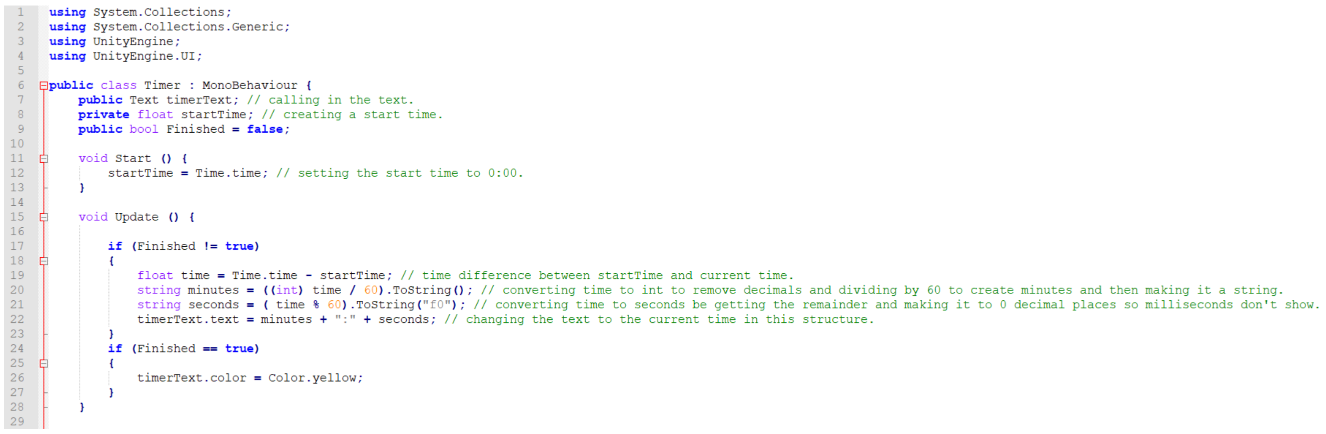


Setting up the TimerText parameter to the Text component.



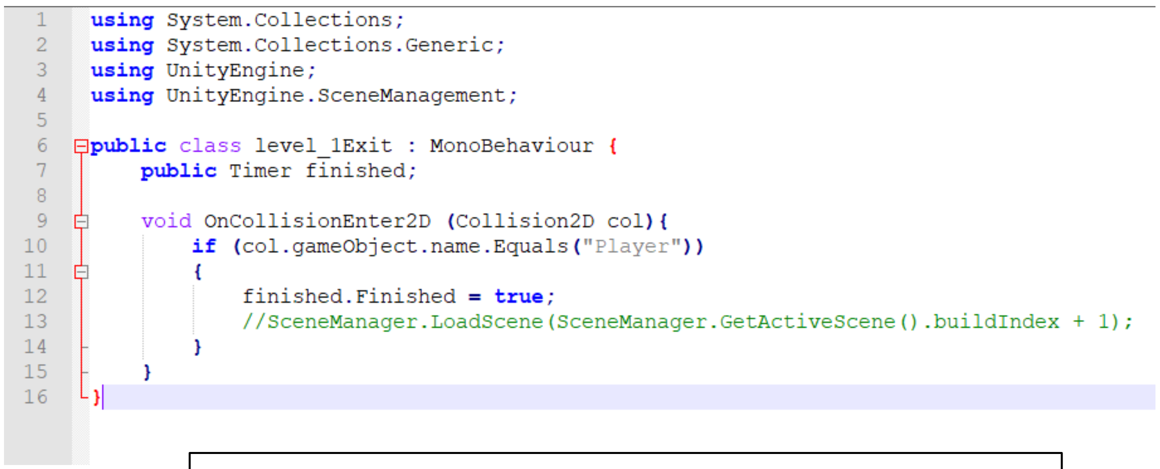
When I tested the timer, it was working fine as shown.

Next, I needed to make the timer stop when the player reaches the exit



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class Timer : MonoBehaviour {
7      public Text timerText; // calling in the text.
8      private float startTime; // creating a start time.
9      public bool Finished = false;
10
11     void Start () {
12         startTime = Time.time; // setting the start time to 0:00.
13     }
14
15     void Update () {
16
17         if (Finished != true)
18         {
19             float time = Time.time - startTime; // time difference between startTime and current time.
20             string minutes = ((int) time / 60).ToString(); // converting time to int to remove decimals and dividing by 60 to create minutes and then making it a string.
21             string seconds = ( time % 60).ToString("f0"); // converting time to seconds be getting the remainder and making it to 0 decimal places so milliseconds don't show.
22             timerText.text = minutes + ":" + seconds; // changing the text to the current time in this structure.
23         }
24         if (Finished == true)
25         {
26             timerText.color = Color.yellow;
27         }
28     }
29
```

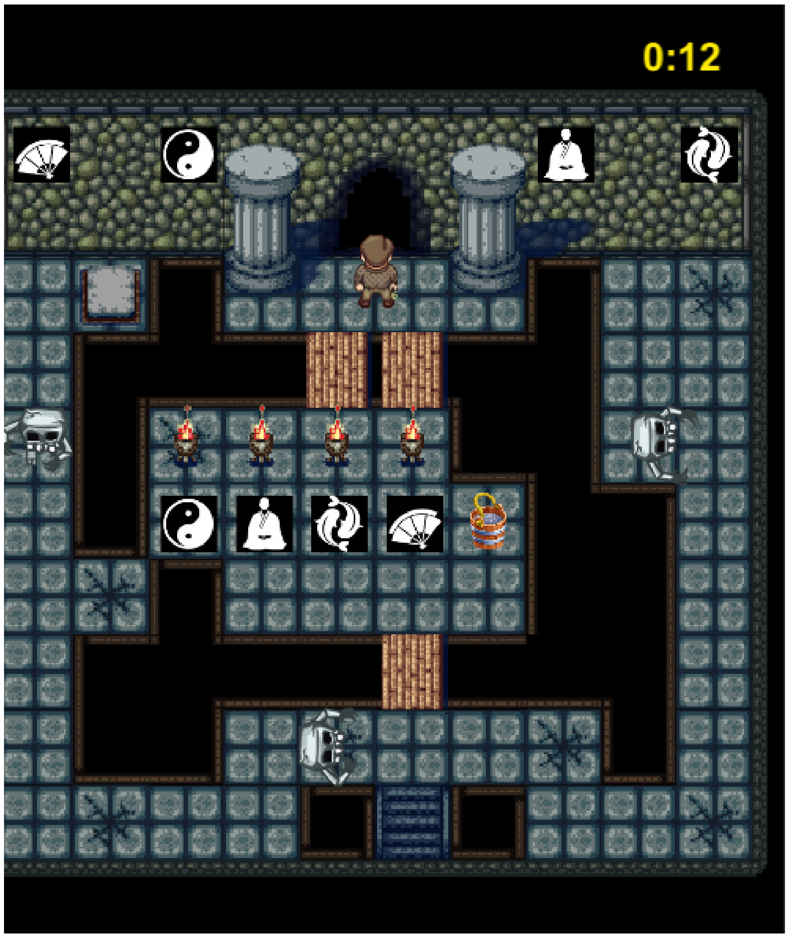
To do this is very simple, all that's needed is to create a Boolean variable and call it false to start off. Then the Timer changing part of the code will only run if the "Finished" Boolean doesn't equal "true" and then when "Finished" equals "true" it changes the colour of the text to yellow and prevents the lines 19 to 22 from being executed.



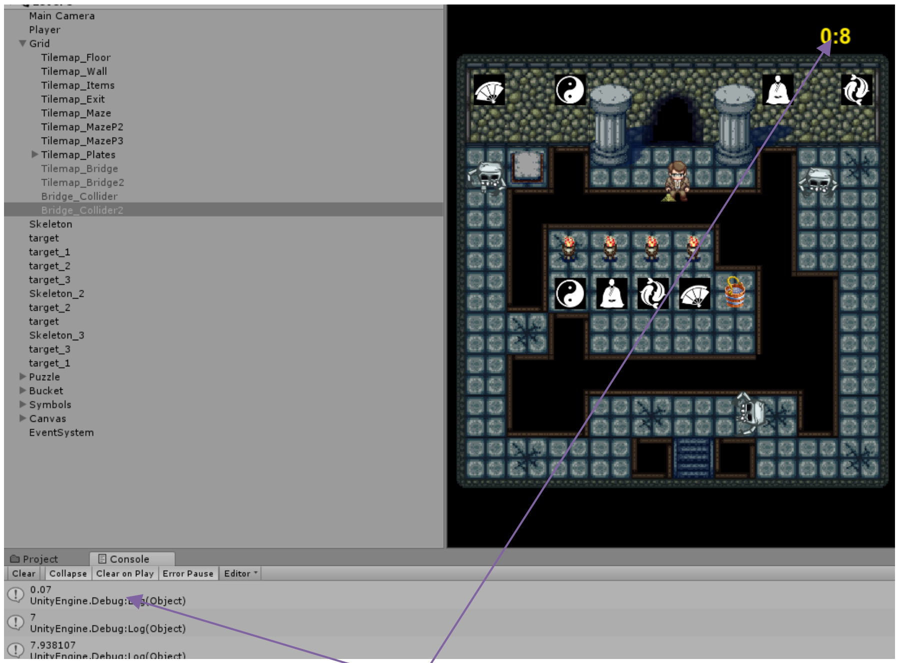
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class level_1Exit : MonoBehaviour {
    public Timer finished;

    void OnCollisionEnter2D (Collision2D col){
        if (col.gameObject.name.Equals("Player"))
        {
            finished.Finished = true;
            //SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
        }
    }
}
```

Then in the Exit script, when the Player collides with the exit it sets the "Finished" bool to equal "true" which is linked to the Timer script.

TESTING



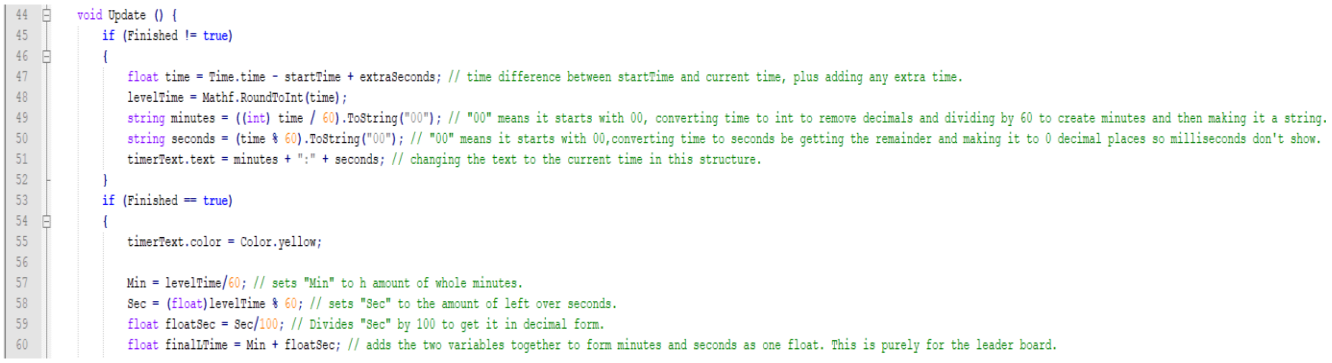
Showing that when the Player collides with the exit, it changes the Timer text colour to yellow and stops the Timer.



```
0.07
UnityEngine.Debug:Log(Object)
7
UnityEngine.Debug:Log(Object)
7.938107
UnityEngine.Debug:Log(Object)
```

When I added some code to output the time, minutes and seconds values as well in the console it wasn't working correctly. This is a problem because I need to use the variables to save them in the Leaderboard and so they must be correct and equal to the Timer. Aswell the Timer has gone to "0:8" which is 80 seconds and it hasn't converted to "1:2".

It figured it must be a rounding error.



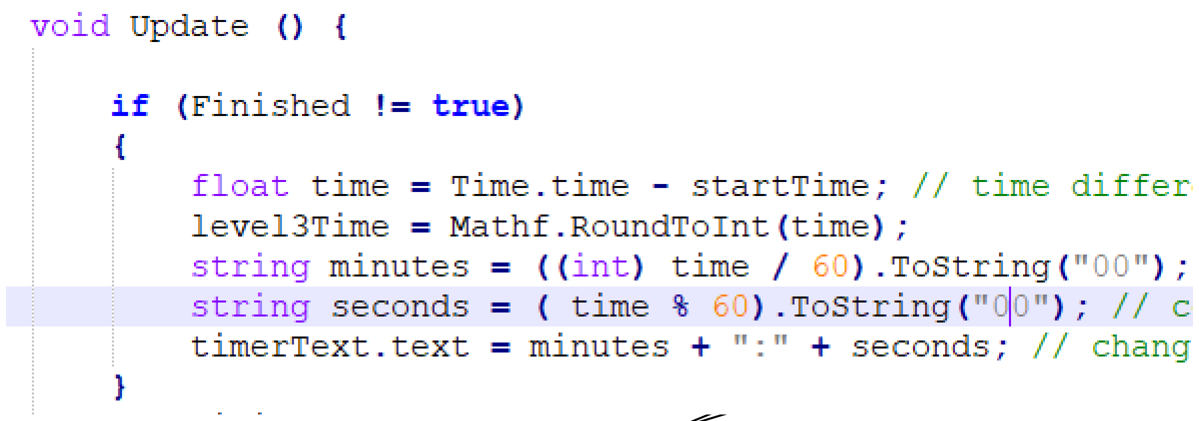
```
44  void Update () {
45      if (Finished != true)
46      {
47          float time = Time.time - startTime + extraSeconds; // time difference between startTime and current time, plus adding any extra time.
48          levelTime = Mathf.RoundToInt(time);
49          string minutes = ((int) time / 60).ToString("00"); // "00" means it starts with 00, converting time to int to remove decimals and dividing by 60 to create minutes and then making it a string.
50          string seconds = (time % 60).ToString("00"); // "00" means it starts with 00,converting time to seconds be getting the remainder and making it to 0 decimal places so milliseconds don't show.
51          timerText.text = minutes + ":" + seconds; // changing the text to the current time in this structure.
52      }
53      if (Finished == true)
54      {
55          timerText.color = Color.yellow;
56
57          Min = levelTime/60; // sets "Min" to h amount of whole minutes.
58          Sec = (float)levelTime % 60; // sets "Sec" to the amount of left over seconds.
59          float floatSec = Sec/100; // Divides "Sec" by 100 to get it in decimal form.
60          float finalLTime = Min + floatSec; // adds the two variables together to form minutes and seconds as one float. This is purely for the leader board.
```

To fix the problem I found out that the rounding that I was doing was rounding 1.6 to 1 because it was just to its integer, not nearest integer. Therefore, I needed the line 48, "Mathf.RoundToInt()" which rounds the float to its nearest integer so 1.6 will now round to 2 and I set this to the new integer variable "levelTime". This means that "levelTime" is the float "time" rounded to an integer so 130.56666 will become 131. [2]

Next, I created a few new variables "sec", "Min" "floatSec" and "finalLTime". "Min" is just "levelTime" divided by 60 which is the amount of minutes, as it is an integer which just takes the integer out, for example, 1.4 becomes 1 as I only want whole minutes, the 0.4 will be represented by the seconds instead.

The "Sec" is a float that will equal "levelTime" mod 60 which is the remainder left over when divided by 60. For Example the 130 will become 10. Then the "floatSec" is the "Sec" but divided by 100 to make it a decimal. For Example, 50 seconds will become 0.5 as that's how seconds is represented.

Then "finalLTime" is the "Min" integer added to the "floatSec", for example, Min of 2 and floarSec of 0.4 will become 2.4 which is 2 minutes and 40 seconds.

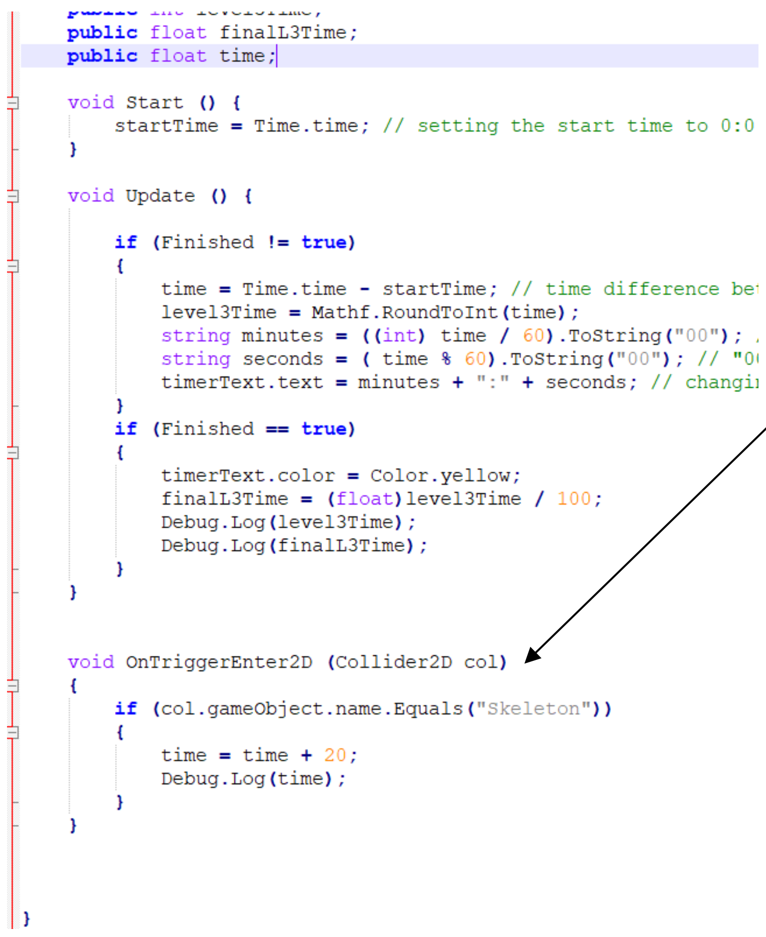


```
void Update () {

    if (Finished != true)
    {
        float time = Time.time - startTime; // time differ
        level3Time = Mathf.RoundToInt(time);
        string minutes = ((int) time / 60).ToString("00");
        string seconds = ( time % 60).ToString("00"); // c
        timerText.text = minutes + ":" + seconds; // chang
    }
```

I also formatted the strings a bit better by adding "00" to the brackets for converting to string. This is means that it will have a format of 00:00 which is the proper format for time.

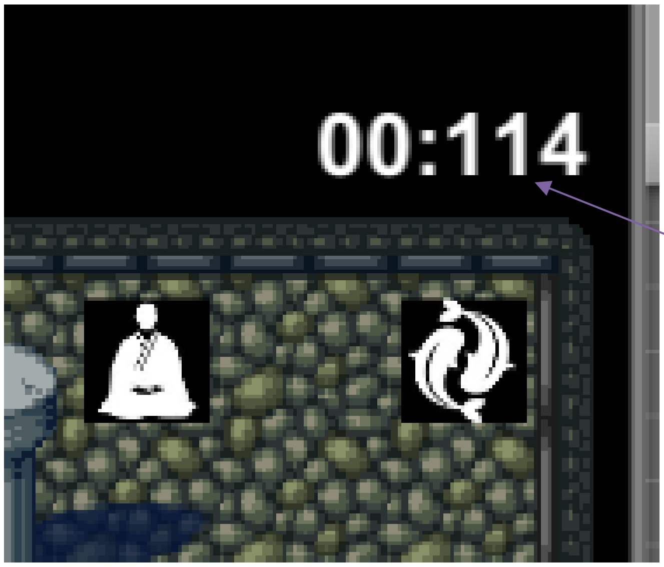
[2] https://doc
Rounding

Adding Time

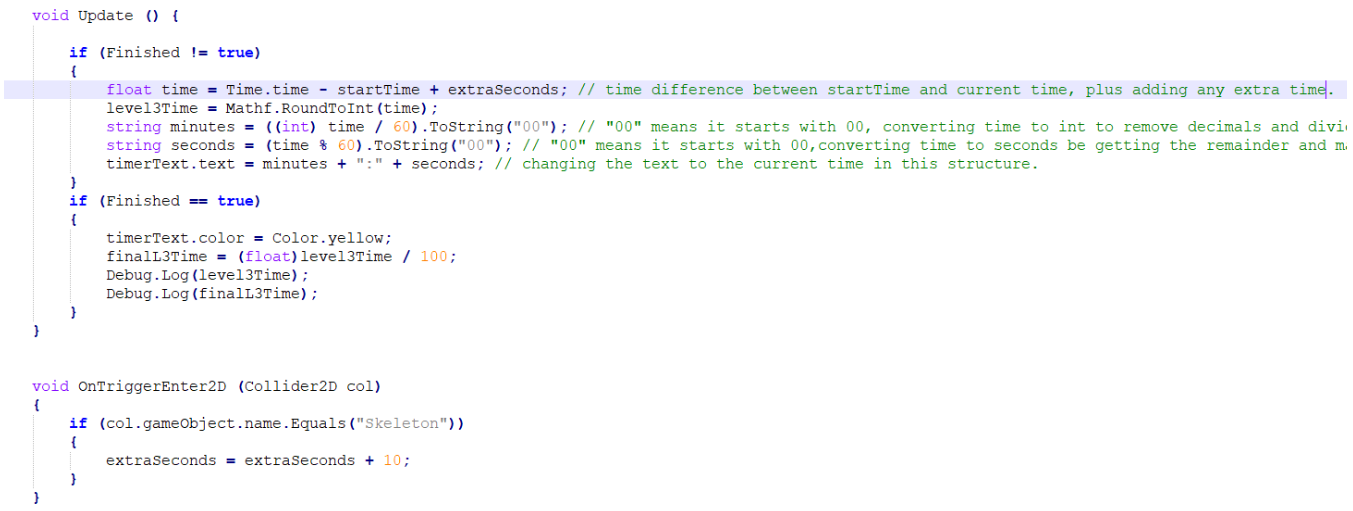


```
    public int level3Time;
    public float finalL3Time;
    public float time;

    void Start () {
        startTime = Time.time; // setting the start time to 0:0
    }

    void Update () {

        if (Finished != true)
        {
            time = Time.time - startTime; // time difference be
            level3Time = Mathf.RoundToInt(time);
            string minutes = ((int) time / 60).ToString("00"); ,
            string seconds = ( time % 60).ToString("00"); // "0
            timerText.text = minutes + ":" + seconds; // changi
        }
        if (Finished == true)
        {
            timerText.color = Color.yellow;
            finalL3Time = (float)level3Time / 100;
            Debug.Log(level3Time);
            Debug.Log(finalL3Time);
        }
    }


    void OnTriggerEnter2D (Collider2D col)
    {
        if (col.gameObject.name.Equals("Skeleton"))
        {
            time = time + 20;
            Debug.Log(time);
        }
    }



}
```

I Then wanted to add time to the timer when the Player collided with a Skeleton GameObject. To do this I added the collider function which adds 20 to time.

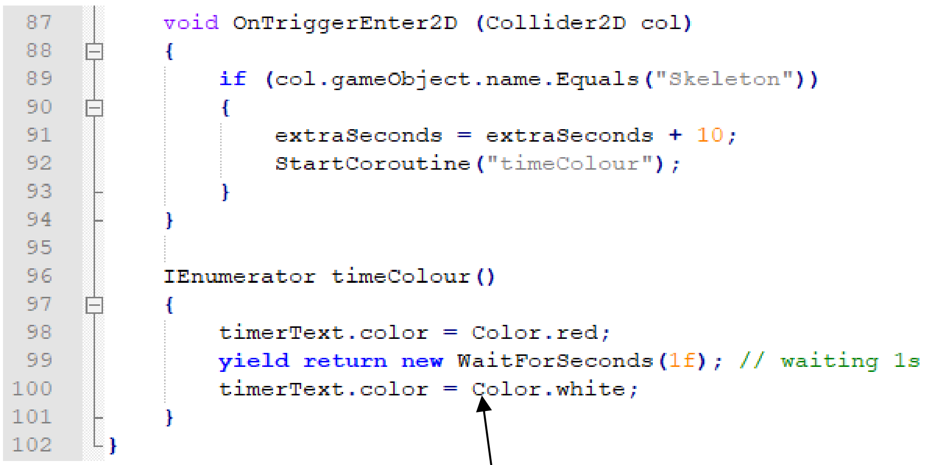


I tested this but it wasn't working properly, it did add 20 to the timer but it didn't convert over to minutes, it just added it to seconds and so it became over 60 seconds.



```
void Update () {

    if (Finished != true)
    {
        float time = Time.time - startTime + extraSeconds; // time difference between startTime and current time, plus adding any extra time.
        level3Time = Mathf.RoundToInt(time);
        string minutes = ((int) time / 60).ToString("00"); // "00" means it starts with 00, converting time to int to remove decimals and divi
        string seconds = (time % 60).ToString("00"); // "00" means it starts with 00,converting time to seconds be getting the remainder and m
        timerText.text = minutes + ":" + seconds; // changing the text to the current time in this structure.
    }
    if (Finished == true)
    {
        timerText.color = Color.yellow;
        finalL3Time = (float)level3Time / 100;
        Debug.Log(level3Time);
        Debug.Log(finalL3Time);
    }
}

void OnTriggerEnter2D (Collider2D col)
{
    if (col.gameObject.name.Equals("Skeleton"))
    {
        extraSeconds = extraSeconds + 10;
    }
}
```

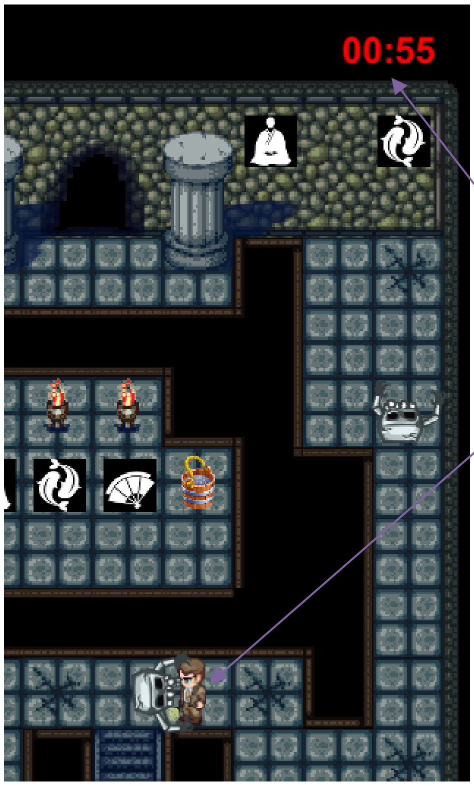
I found out the reason was because it was adding the 20 seconds incorrectly, what I should have done was create a new integer which is added to and combined with the "time" float, this way it will be directly added within the same line that "time" is set and also before any other variables are calculated and set.



```
    void OnTriggerEnter2D (Collider2D col)
    {
        if (col.gameObject.name.Equals("Skeleton"))
        {
            extraSeconds = extraSeconds + 10;
            StartCoroutine("timeColour");
        }
    }

    IEnumerator timeColour()
    {
        timerText.color = Color.red;
        yield return new WaitForSeconds(1f); // waiting 1s
        timerText.color = Color.white;
    }
}
```

I also wanted it so that when the Player had extra time added to the Timer, it would turn red for one second and then back to white. This is to clearly notify to the Player that time has been added to the Timer.

To do this a "IEnumerator" function "timeColour" is called which turns the text red, waits 1 second and then turns it back to white.



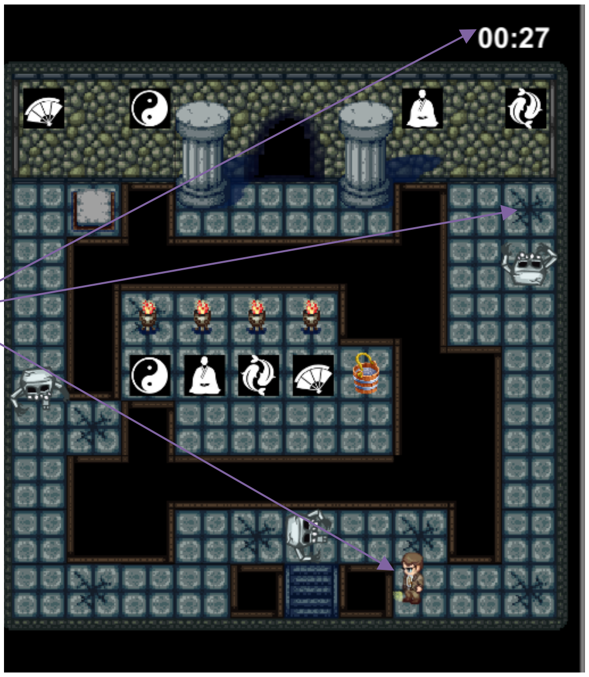
Showing that when the Player collides with a Skeleton, it does add 10 seconds to the Timer and turns it red for a second.

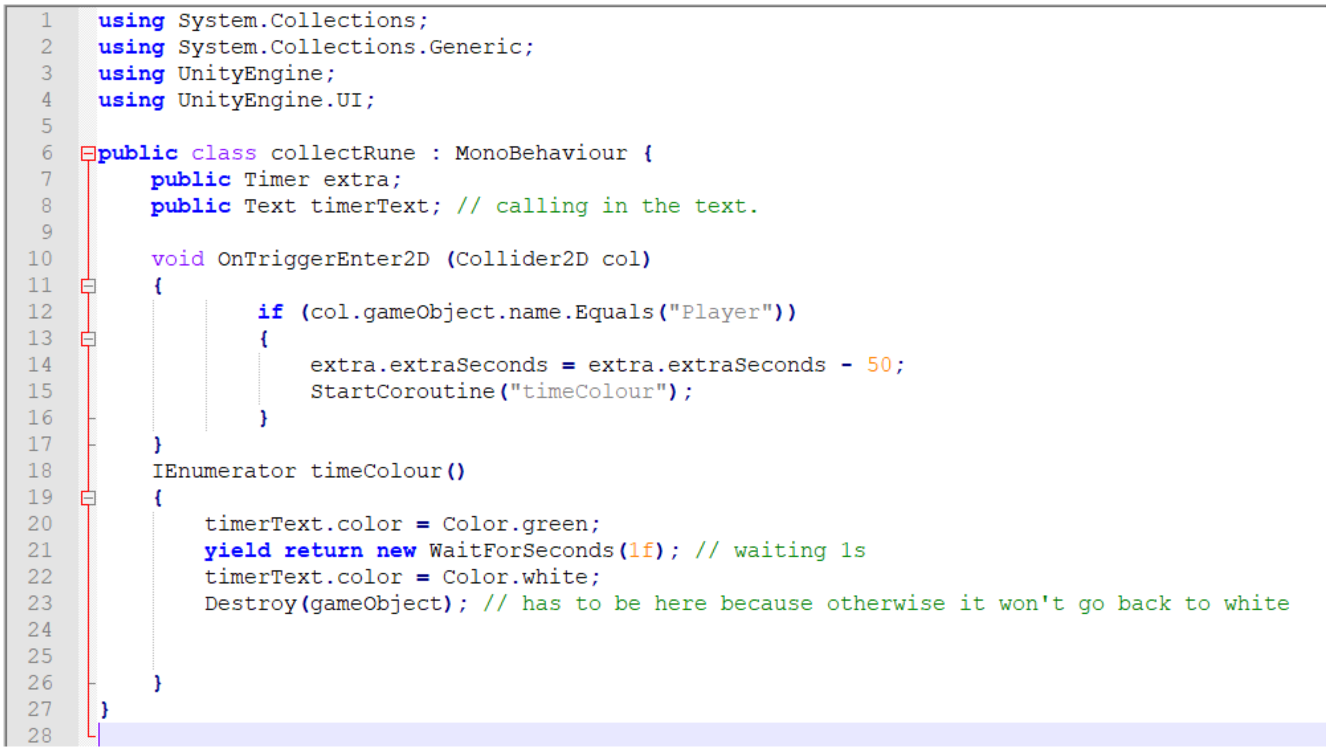
Rune

After developing the Timer is working, I could develop the Rune because when the Rune is picked up by the Player, I want it to deduct time from the Timer.

First I wanted to check how long it took the Player to walk over to the Rune, collect it and then walk back to the start. This is so that I can calculate how much time a Rune should deduct from the Timer because if it is less than the timer it took to walk over there, then it's not worth collecting.

As shown, it takes 27 seconds to walk over the Rune, collect it and then walk back to the start. Therefore, to make the walk worth it I decided to make a Rune deduct 50 seconds, as gain of 23 seconds is very much worth it. (The Rune was below the fish symbol).



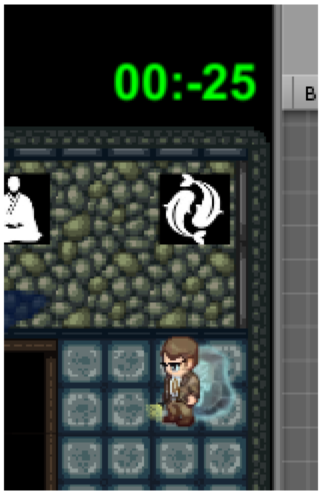


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class collectRune : MonoBehaviour {
    public Timer extra;
    public Text timerText; // calling in the text.

    void OnTriggerEnter2D (Collider2D col)
    {
            if (col.gameObject.name.Equals("Player"))
            {
                extra.extraSeconds = extra.extraSeconds - 50;
                StartCoroutine("timeColour");
            }
    }
    IEnumerator timeColour()
    {
        timerText.color = Color.green;
        yield return new WaitForSeconds(1f); // waiting 1s
        timerText.color = Color.white;
        Destroy(gameObject); // has to be here because otherwise it won't go back to white


    }
}

```

This script is put on the Rune GameObjects, it checks for a Player colliding with its 2D box collider and if it does then it takes away 50 from "extraSeconds" which is called from the Timer script.

It also changes the colour to green when the Player picks up the Rune as it calls the "timeColour" functions as well and then it destroys the GameObject.

TESTING

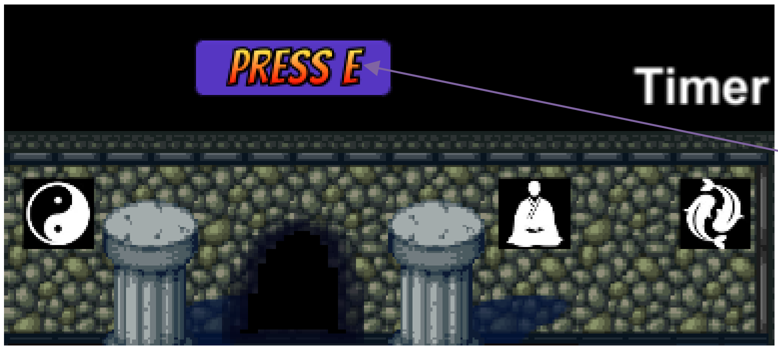


Showing that when the Player picks up the Rune, the Timer has 50 seconds taken away, in this case it was at 25 seconds but when the Player picks up the Rune, it takes away 50 and therefore the Timer is now -25 seconds. The Timer text also turns green for one second.

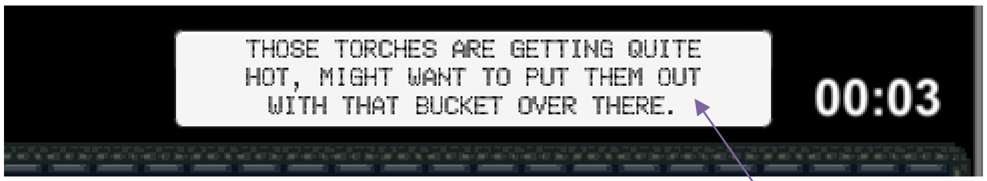
Text Boxes

Next, I wanted to develop the Hint system as well as text boxes that notified the player to press a button to interact with something. For example, to pick up the Water Bucket, a text box "press E" pops up when in range.

DESIGN

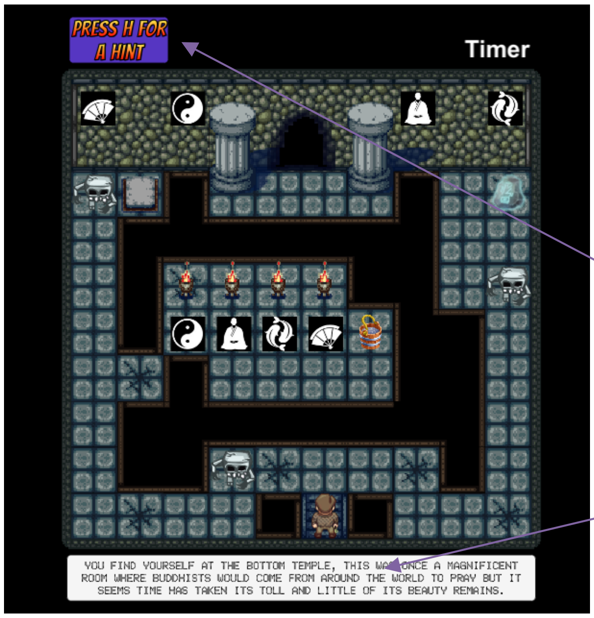


Press E image consisting of a rounded rectangle box and a text GameObject.


THOSE TORCHES ARE GETTING QUITE HOT, MIGHT WANT TO PUT THEM OUT WITH THAT BUCKET OVER THERE.

00:03

Hint image consisting of a rounded rectangle box and a text GameObject.

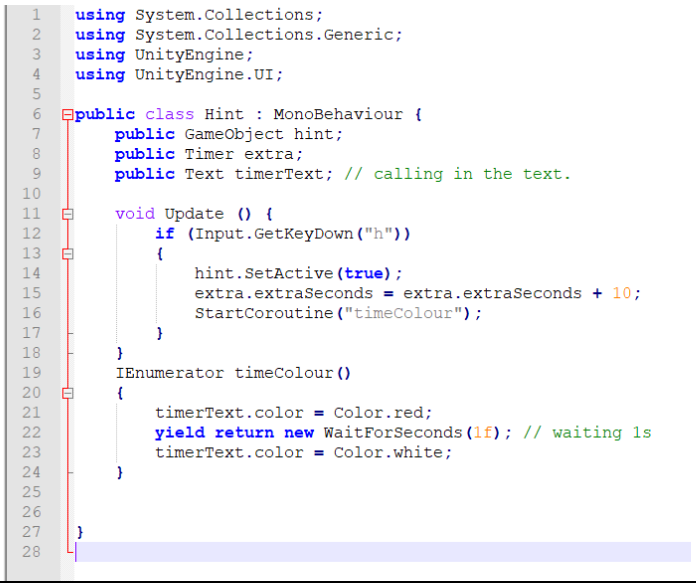


YOU FIND YOURSELF AT THE BOTTOM TEMPLE, THIS WAS ONCE A MAGNIFICENT ROOM WHERE BUDDHISTS WOULD COME FROM AROUND THE WORLD TO PRAY BUT IT SEEMS TIME HAS TAKEN ITS TOLL AND LITTLE OF ITS BEAUTY REMAINS.

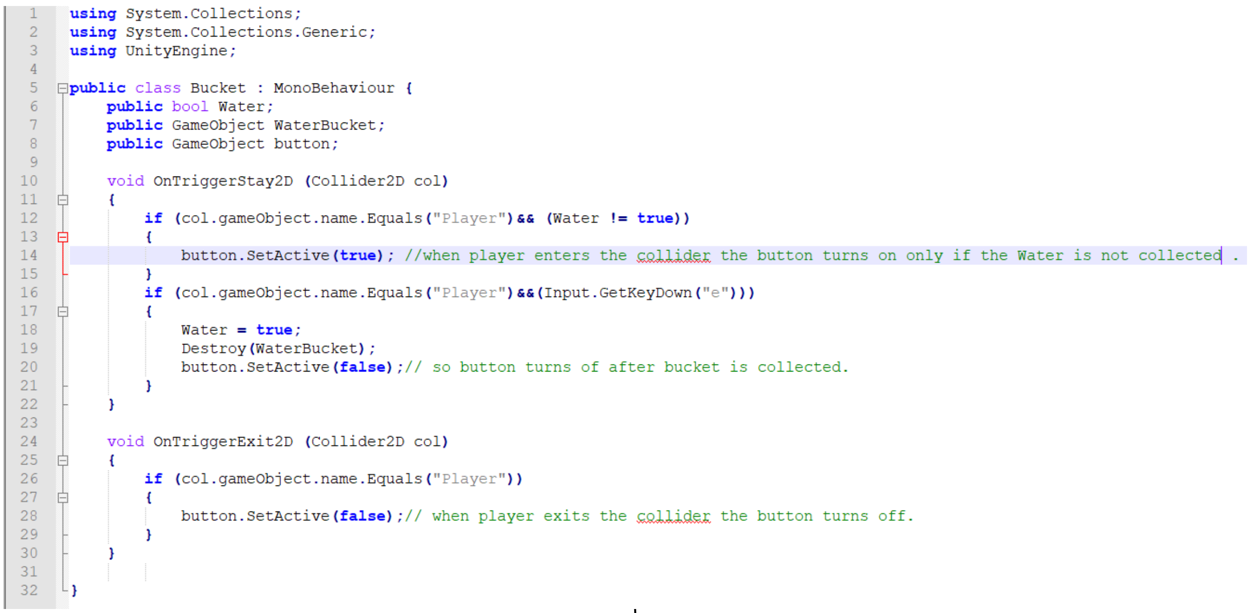
I've also added a Text box that tells the Player what button to press if they want a hint.

Storyline image consisting of a rounded rectangle box and a text GameObject.

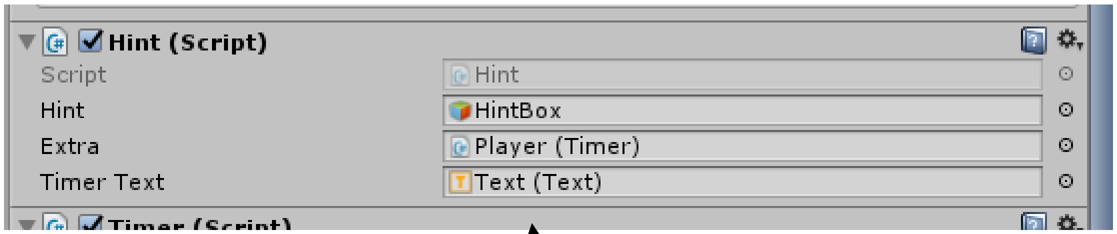
Scripts


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Hint : MonoBehaviour {
    public GameObject hint;
    public Timer extra;
    public Text timerText; // calling in the text.

    void Update () {
        if (Input.GetKeyDown("h"))
        {
            hint.SetActive(true);
            extra.extraSeconds = extra.extraSeconds + 10;
            StartCoroutine("timeColour");
        }
    }
    IEnumerator timeColour()
    {
        timerText.color = Color.red;
        yield return new WaitForSeconds(1f); // waiting 1s
        timerText.color = Color.white;
    }


}

```

When the Player pressed the button "h" then the Hint text box activates and also adds 10 seconds to the Timer because the Hint must come at a cost to the Player, otherwise they would just use it straight away as there would be no consequence. It also turns the Timer text to red for a second and then back to white.

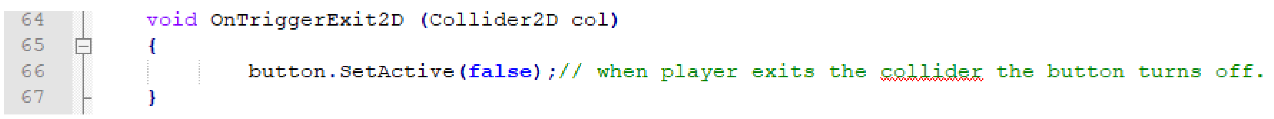

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Bucket : MonoBehaviour {
    public bool Water;
    public GameObject WaterBucket;
    public GameObject button;

    void OnTriggerStay2D (Collider2D col)
    {
        if (col.gameObject.name.Equals("Player")&& (Water != true))
        {
            button.SetActive(true); //when player enters the collider the button turns on only if the Water is not collected .
        }
        if (col.gameObject.name.Equals("Player")&&(Input.GetKeyDown("e")))
        {
            Water = true;
            Destroy(WaterBucket);
            button.SetActive(false);// so button turns of after bucket is collected.
        }
    }

    void OnTriggerExit2D (Collider2D col)
    {
        if (col.gameObject.name.Equals("Player"))
        {
            button.SetActive(false);// when player exits the collider the button turns off.
        }
    }


}
```

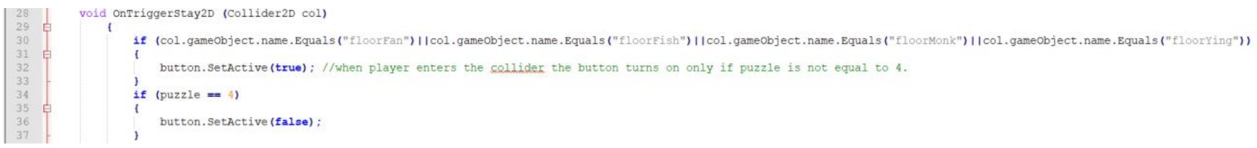
This script is on the Water Bucket GameObject, when the Player collides with the 2D box collider the "Press E" text will activate and if the Player leaves the area then the "Press E" text will deactivate. Also if the Player picks up the Water Bucket then the "Press E" text will deactivate.


Hint (Script)

Script	Hint
Hint	HintBox
Extra	Player (Timer)
Timer Text	Text (Text)

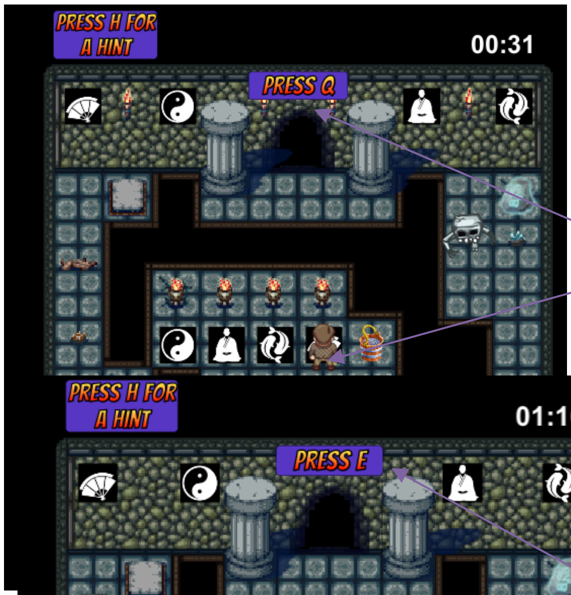
Setting up the parameters for the Hint script


```
    void OnTriggerExit2D (Collider2D col)
    {
            button.SetActive(false);// when player exits the collider the button turns off.
    }
```

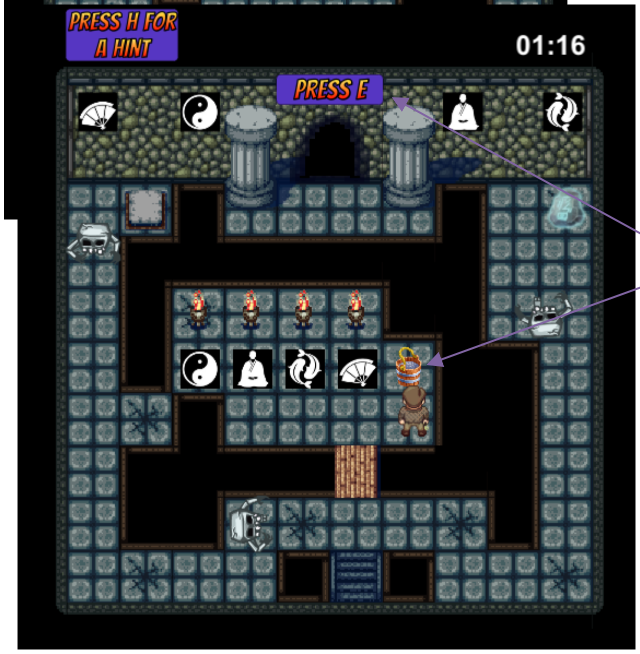

```
    void OnTriggerStay2D (Collider2D col)
        {
            if (col.gameObject.name.Equals("floorFan")||col.gameObject.name.Equals("floorFish")||col.gameObject.name.Equals("floorMonk")||col.gameObject.name.Equals("floorYing"))
            {
                button.SetActive(true); //when player enters the collider the button turns on only if puzzle is not equal to 4.
            }
            if (puzzle == 4)
            {
                button.SetActive(false);
            }
```

Testing

This is the script is the "Symbol" script and is attached to the Player. If the Player collides with either of the symbol colliders, but when the Puzzle equals 4, it deactivates the "Press Q" text box. It also deactivates the "Press Q" when the Player exits its 2D box collider as well.



Showing that when the Player walks on one of the 2D box collider symbols, it activates the "Press Q" text box.

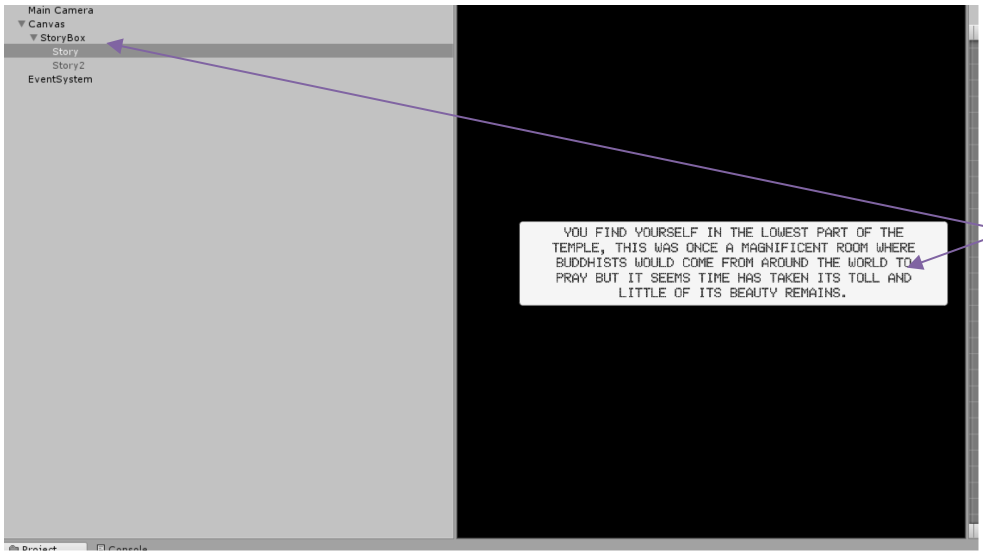


Showing that when the Player walks on the Water Bucket 2D box collider it activates the "Press E" text box.

STORYLINE CHANGES

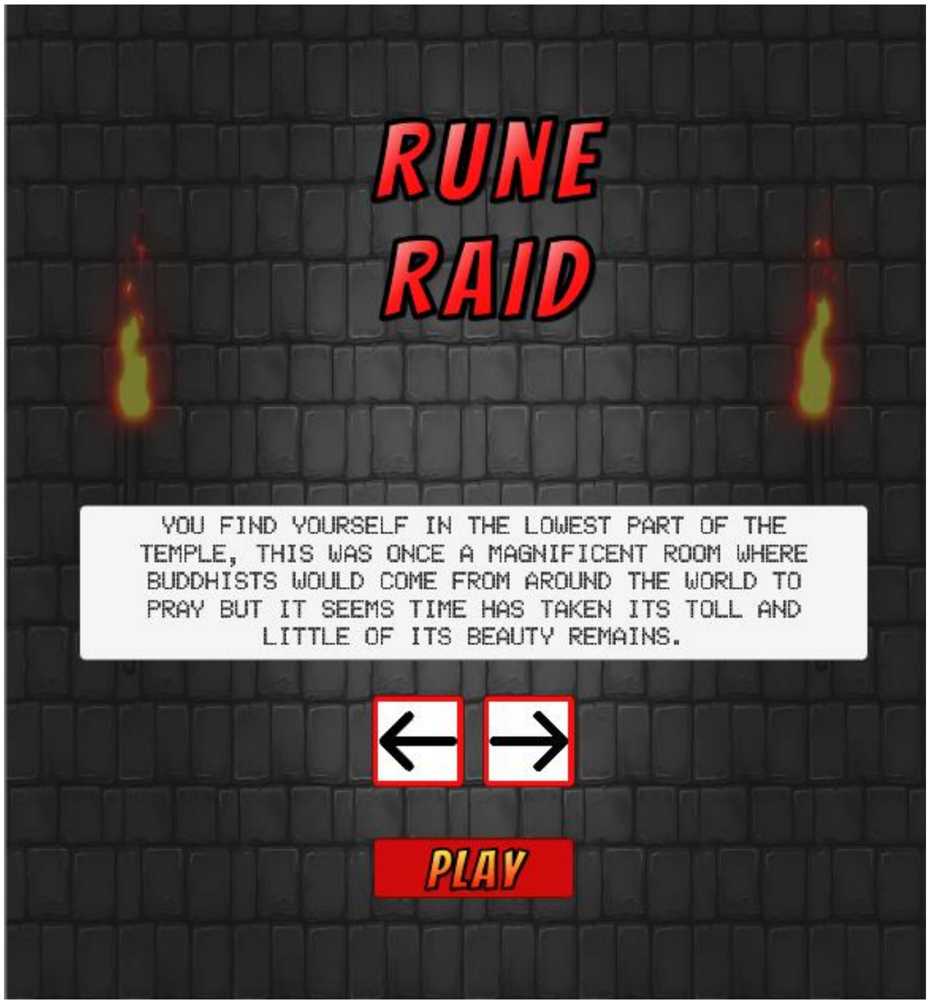
After some thought I came to the decision to make the storyline in a totally separate scene for some reasons. One is that the area available for the text box is too small and the second is that when the level loads, the player most likely isn't going to read the story because either they want to just complete the level or that the Timer is still going when they are reading it.

Therefore, because of this I decided to make 3 new scenes for the storyline, instead of in the level.



YOU FIND YOURSELF IN THE LOWEST PART OF THE TEMPLE, THIS WAS ONCE A MAGNIFICENT ROOM WHERE BUDDHISTS WOULD COME FROM AROUND THE WORLD TO PRAY BUT IT SEEMS TIME HAS TAKEN ITS TOLL AND LITTLE OF ITS BEAUTY REMAINS.

Creating a Text box containing the storyline. There is two parts to the story, "story" and "story2"



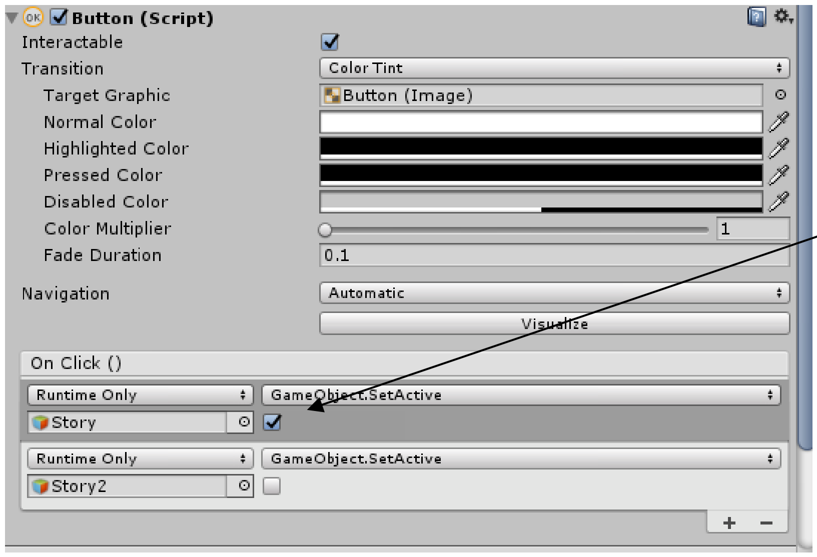
YOU FIND YOURSELF IN THE LOWEST PART OF THE TEMPLE, THIS WAS ONCE A MAGNIFICENT ROOM WHERE BUDDHISTS WOULD COME FROM AROUND THE WORLD TO PRAY BUT IT SEEMS TIME HAS TAKEN ITS TOLL AND LITTLE OF ITS BEAUTY REMAINS.

[3]

I brought in the "Rune Raid" sign from the Game Menu scene, added a background and 3 buttons.

The right arrow button activates the "Story2" text and deactivates the "story" text. The left arrow button does the opposite.

The Play button loads the next scene which will be the next level

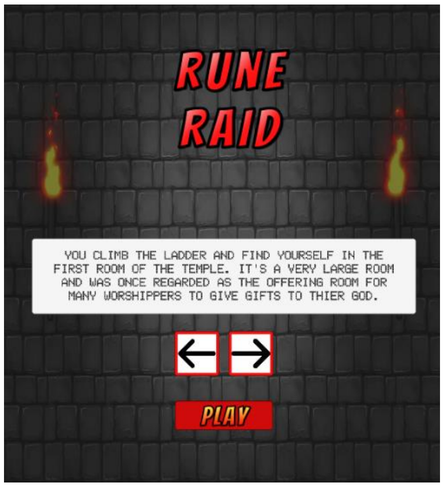


```
Button (Script)
Interactable         ✓
Transition           Color Tint
  Target Graphic     Button (Image)
  Normal Color
  Highlighted Color
  Pressed Color
  Disabled Color
  Color Multiplier   1
  Fade Duration      0.1
Navigation           Automatic
                     Visualize
On Click ()
Runtime Only   GameObject.SetActive
Story          ✓
Runtime Only   GameObject.SetActive
Story2         ☐
```

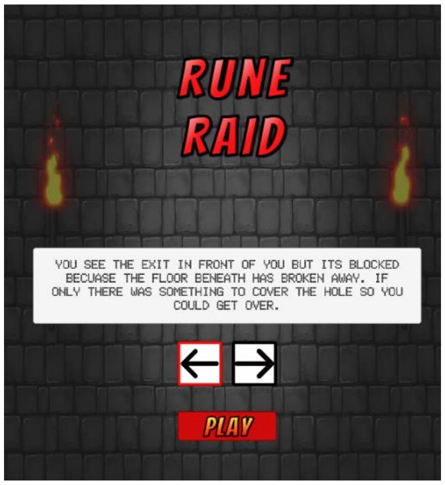
Setting up the arrow buttons

[3] https://www.videoblocks.com/video/dark-and-spooky-dungeon-with-torches-hanging-on-a-brick-wall-nrpgpdc9eiknnrjtf - Date accessed 12/04/18 - The background image

I've done this for each of the levels, all having their own storyline scene.

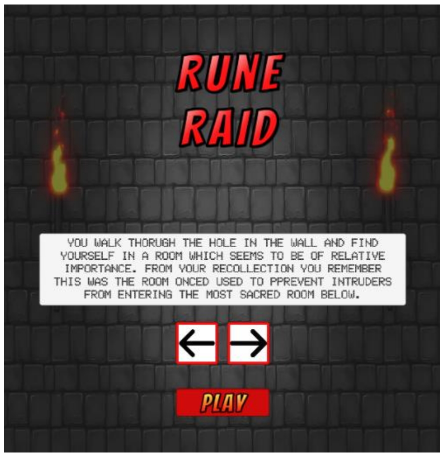


YOU CLIMB THE LADDER AND FIND YOURSELF IN THE FIRST ROOM OF THE TEMPLE. IT'S A VERY LARGE ROOM AND WAS ONCE REGARDED AS THE OFFERING ROOM FOR MANY WORSHIPPERS TO GIVE GIFTS TO THIER GOD.

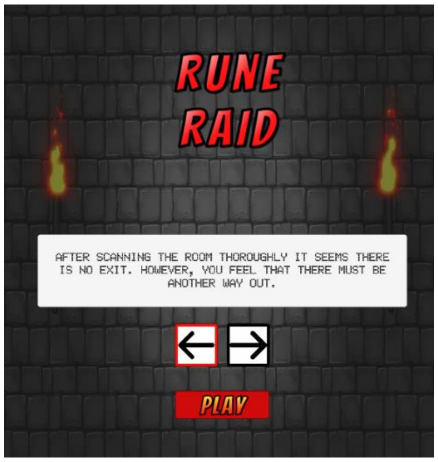


YOU SEE THE EXIT IN FRONT OF YOU BUT ITS BLOCKED BECUASE THE FLOOR BENEATH HAS BROKEN AWAY. IF ONLY THERE WAS SOMETHING TO COVER THE HOLE SO YOU COULD GET OVER.

Story level 1, part 1 and 2

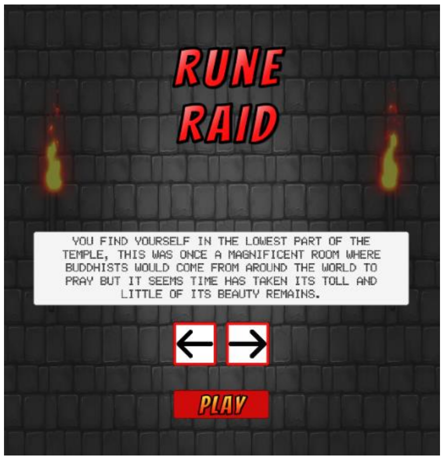


YOU WALK THORUGH THE HOLE IN THE WALL AND FIND YOURSELF IN A ROOM WHICH SEEMS TO BE OF RELATIVE IMPORTANCE. FROM YOUR RECOLLECTION YOU REMEMBER THIS WAS THE ROOM ONCED USED TO PPREVENT INTRUDERS FROM ENTERING THE MOST SACRED ROOM BELOW.

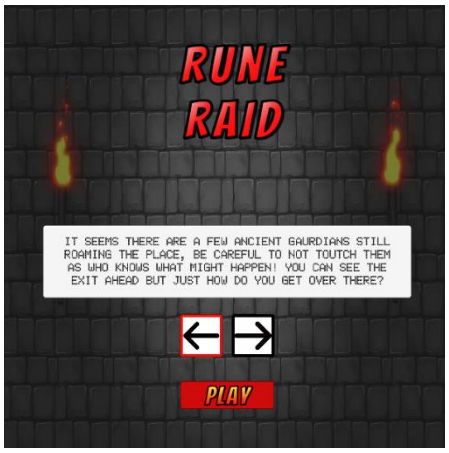


AFTER SCANNING THE ROOM THOROUGHLY IT SEEMS THERE IS NO EXIT. HOWEVER, YOU FEEL THAT THERE MUST BE ANOTHER WAY OUT.

Story level 2, part 1 and 2



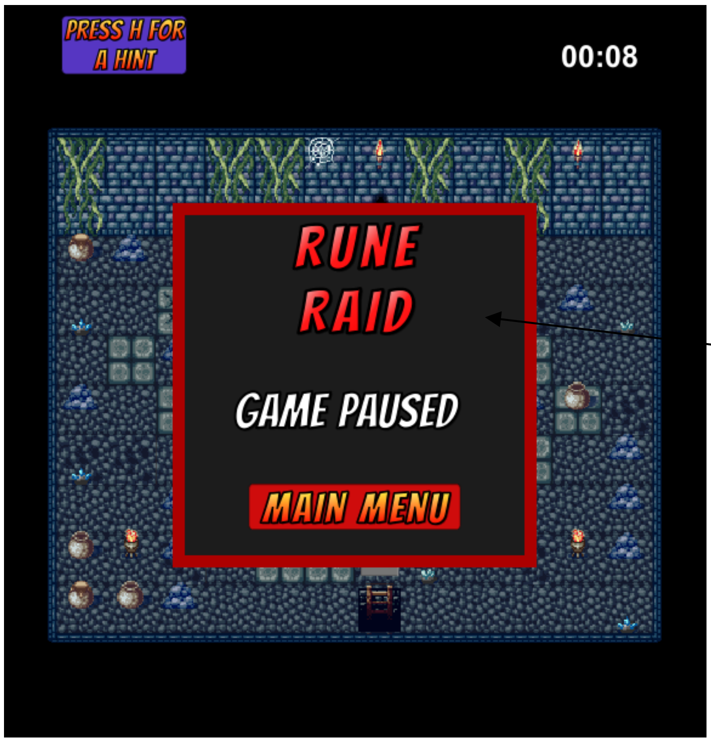
YOU FIND YOURSELF IN THE LOWEST PART OF THE TEMPLE, THIS WAS ONCE A MAGNIFICENT ROOM WHERE BUDDHISTS WOULD COME FROM AROUND THE WORLD TO PRAY BUT IT SEEMS TIME HAS TAKEN ITS TOLL AND LITTLE OF ITS BEAUTY REMAINS.



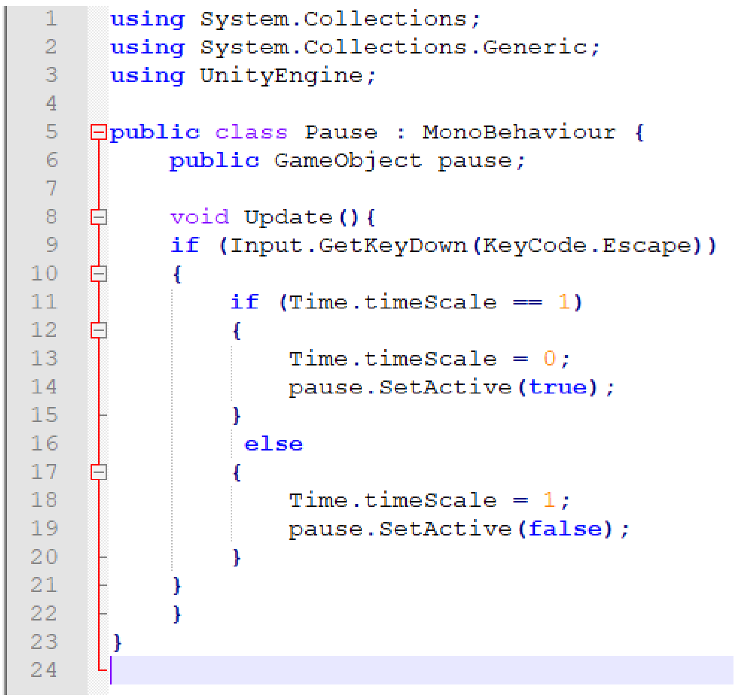
IT SEEMS THERE ARE A FEW ANCIENT GAURDIANS STILL ROAMING THE PLACE, BE CAREFUL TO NOT TOUTCH THEM AS WHO KNOWS WHAT MIGHT HAPPEN! YOU CAN SEE THE EXIT AHEAD BUT JUST HOW DO YOU GET OVER THERE?

Story level 3, part 1 and 2

Pasue Game



I've just created a GameObject that is a pause menu, consisting of a few other GameObjects for the design and also a Main Menu Button to allow the user to go back to the Main Menu if they wish to.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Pause : MonoBehaviour {
    public GameObject pause;

    void Update(){
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        if (Time.timeScale == 1)
        {
            Time.timeScale = 0;
            pause.SetActive(true);
        }
         else
        {
            Time.timeScale = 1;
            pause.SetActive(false);
        }
    }
    }
}
```

This script is added to the Player GameObject and is to allow the user to pause the game at any time.

Line 9 checks if the Esc button is pressed, if it is then if the "Time.timeScale" equals 1 then it is set to 0 and the pause GameObject is set to active, this is the pause menu.

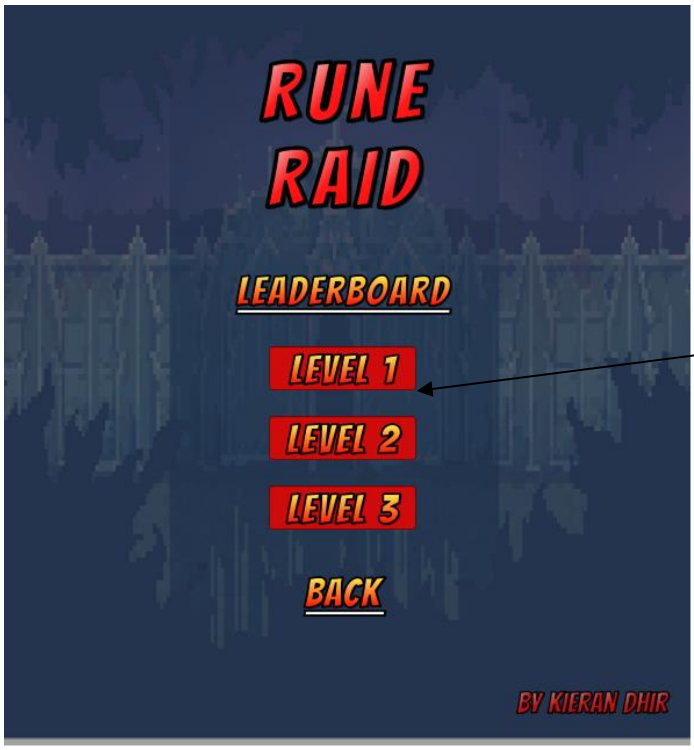
Then if the esc button is pressed again, it runs the else statement and sets the "Time.timeScale" to equal 1 and deactivates the pause GameObject.

Time.timeScale = 1 means that frames are running.

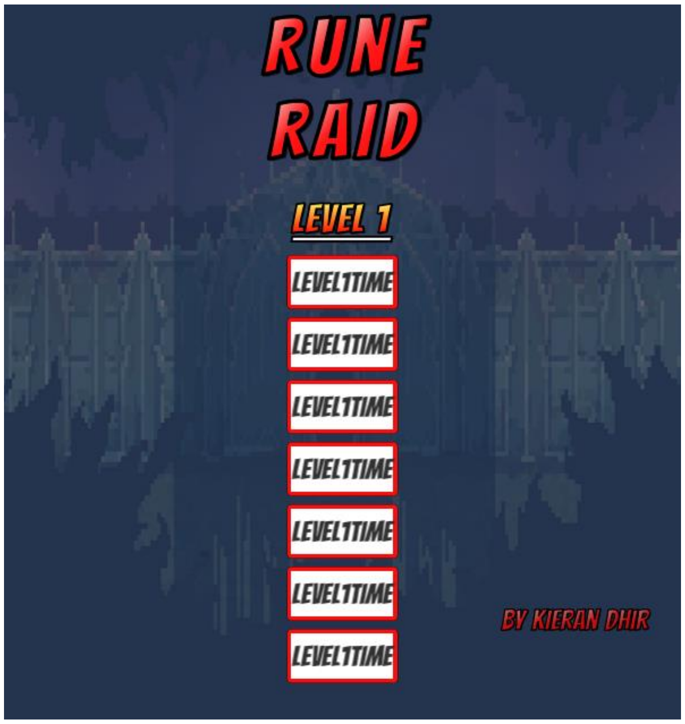
Time.timeScale = 0 means that frames are not running.

Leaderboard

Design



Creating the Leaderboard start screen. The Back button takes the user back to the Game Menu, the level 1 button turns off this Leaderboard layout and turns on the Level 1 layout. It is the same for level 2 and 3.



The level 1 Leaderboard, it is the same for level 2 and 3 but just will contain the data from its own level.

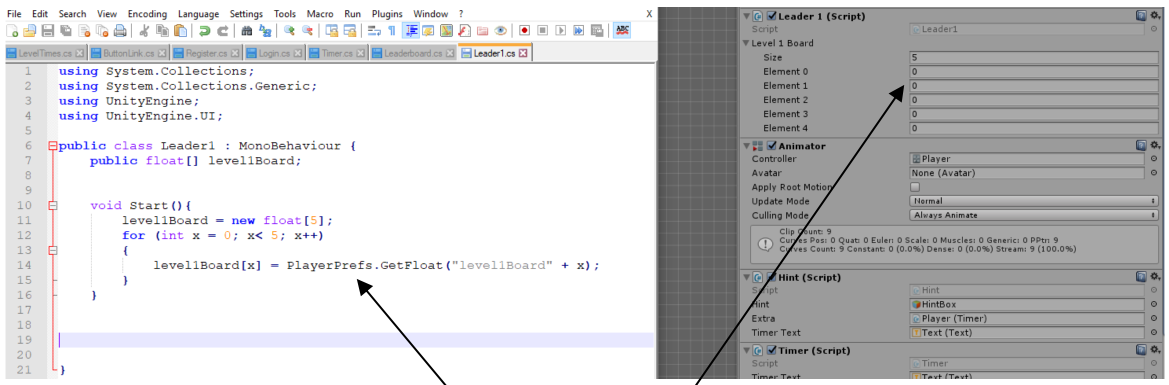
However, at first I wanted to create a leaderboard that will contain every accounts time but then I thought that it would be better if the leaderboard only stored the top 5 quickest times.

This is because I believe it will make it more competitive because the players have to earn a spot on the leaderboard. Many games also do this for leaderboards, instead of having every account it normally has only the top 100 but for Rune Raid, 5 is more relevant as it will have a smaller player base.

Here, I'm showing that more than 5 times is too cramped

SCRIPT

I found a YouTube video that very clearly explained how to create a leaderboard that saved times and their names. Therefore, I used this person's code and modified it to my needs.


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
LevelTimes.cs ButtonLink.cs Register.cs Login.cs Timer.cs Leaderboard.cs Leader1.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Leader1 : MonoBehaviour {
    public float[] level1Board;


    void Start(){
        level1Board = new float[5];
        for (int x = 0; x< 5; x++)
        {
            level1Board[x] = PlayerPrefs.GetFloat("level1Board" + x);
        }
    }


}
```

Leader 1 (Script)
Script: Leader1
Level 1 Board
Size: 5
Element 0: 0
Element 1: 0
Element 2: 0
Element 3: 0
Element 4: 0

Animator
Controller: Player
Avatar: None (Avatar)
Apply Root Motion: ☐
Update Mode: Normal
Culling Mode: Always Animate

```
Clip Count: 9
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 9
Curves Count: 9 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 9 (100.0%)
```

Hint (Script)
Script: Hint
Hint: HintBox
Extra: Player (Timer)
Timer Text: Text (Text)

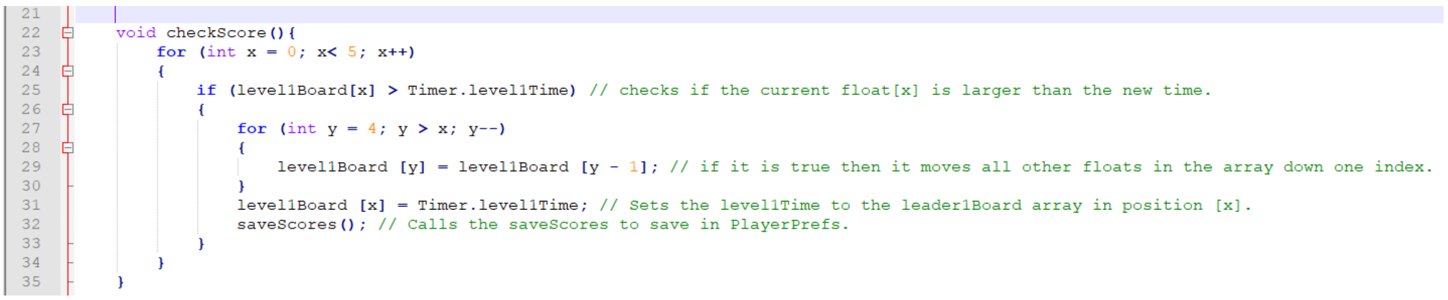
Timer (Script)
Script: Timer
Timer Text: Text (Text)

The way to store the time data is to use "PlayerPrefs" which is a built I function that saves data to a file within Unity which can be called, changed and queried.

As there is 5 places on the leaderboard, then 5 different PlayerPrefs must be made and contain the data from the top 5 quickest account times for the level. A float array must be also created to use and modify the values from PlayerPrefs.

In the Start function it sets the "level1Board" to the new float with an index of 5 because of the 5 times needed to be stored. Then it sets each "level1Board index to the values in the PlayerPrefs via the loop. For Example, the "level1Board" index 2 gets the float value from

After each index value for the "level1Board" array is set, another function needs to be made which executes when a player completes a level with a time. This function will check whether the time is quicker than any of the currently stored values in the array and if they are then it will add the new time in and move the rest below it, down by one place, eliminating the old 5th time. For example, if a timer quicker than the index's 4 and 5 then that means that the new time must go into the index 4 and the previous index 4 value must be moved to index 5 and index 5 previous value is removed completely.



```
21
22  void checkScore(){
23      for (int x = 0; x< 5; x++)
24      {
25          if (level1Board[x] > Timer.level1Time) // checks if the current float[x] is larger than the new time.
26          {
27              for (int y = 4; y > x; y--)
28              {
29                  level1Board [y] = level1Board [y - 1]; // if it is true then it moves all other floats in the array down one index.
30              }
31              level1Board [x] = Timer.level1Time; // Sets the level1Time to the leader1Board array in position [x].
32              saveScores(); // Calls the saveScores to save in PlayerPrefs.
33          }
34      }
35  }
```

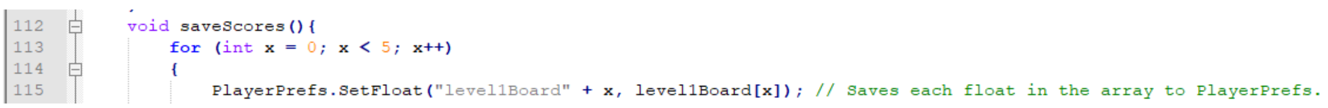
This is the code that does this. It checks each index value via the looping from 0 to 5 meaning index 0 to 4 are checked.

Line 25 checks if the new time value is smaller (quicker) than the current value stored in the array index. If it is then the next for loop is executed, which then loops from 4 to the index that it is first quicker than. Index 4 is actually the 5th place as it starts at index 0. This loop is a reverse loop as it will go from 4 to say 2, instead of the normal 0 to 5. It sets each index to the index value above. For example, index 4 becomes the value of index 3 as 4 - 1 = 3.

In simple terms, if the "level1Board" index 2 value is larger than the new time then the line 27 loop runs 3 times. This is because the current index value 4 must become the index 5 value, the current index 3 value must then become the index 4 value, the current index 2 value must become the index 3 value and then the index 2 value must be set to the value of the new time. It must be done in reverse to correctly shift the scores down. If it was normal forward then the index 2 value would become the index 3 value but then when the old index 3 value needs to become the new index 4 it can't because the value has been overwritten.

A full example, check score is executed and the line 23 runs through 2 times and finds that the Index 1 is larger than the new time. This means that the line 27 is run which then loops from index 4 to the index X which is the current index that line 25 is on, in this case index 1.

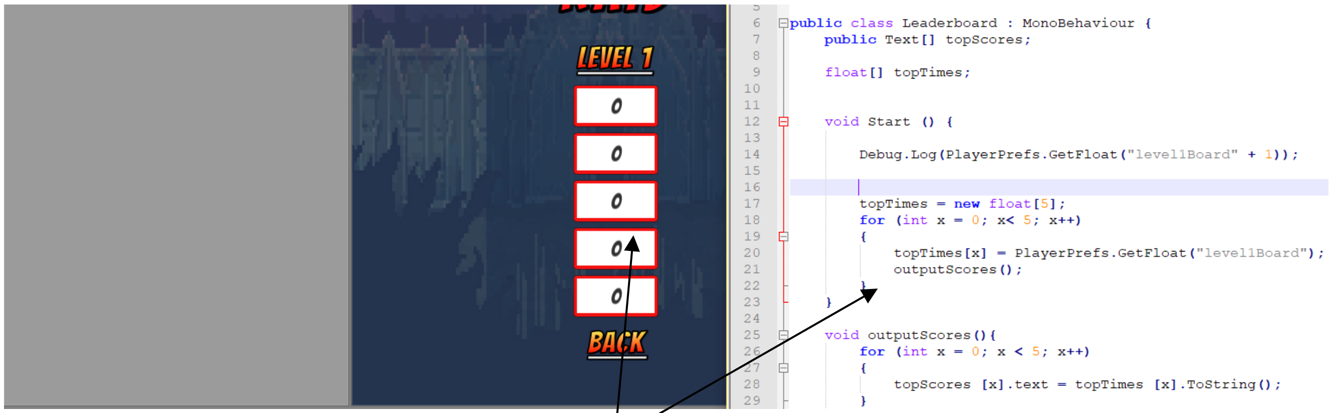
That means that it loops from 4 to 1 which is 4 loops. Then all the indexes from 4 to 1 is changed by moving them down one index, so index 5 is set to the value of index 4 and then index 4 is set to the value of index 3 and so on. Then when the loops finishes, the index X from line 25 is set to the new time value, then "saveScores" is called which is used to save the "level1Board" values back to the PlayerPrefs.



```
void saveScores(){
    for (int x = 0; x < 5; x++)
    {
        PlayerPrefs.SetFloat("level1Board" + x, level1Board[x]); // Saves each float in the array to PlayerPrefs.
```

This is the "saveScore" function which just loops through each index, saving each index from "level1Board" to the PlayerPrefs "level1Board + x", x is the index value so "level1Board3" for example. This is done with the "SetFloat" instead of "GetFloat".

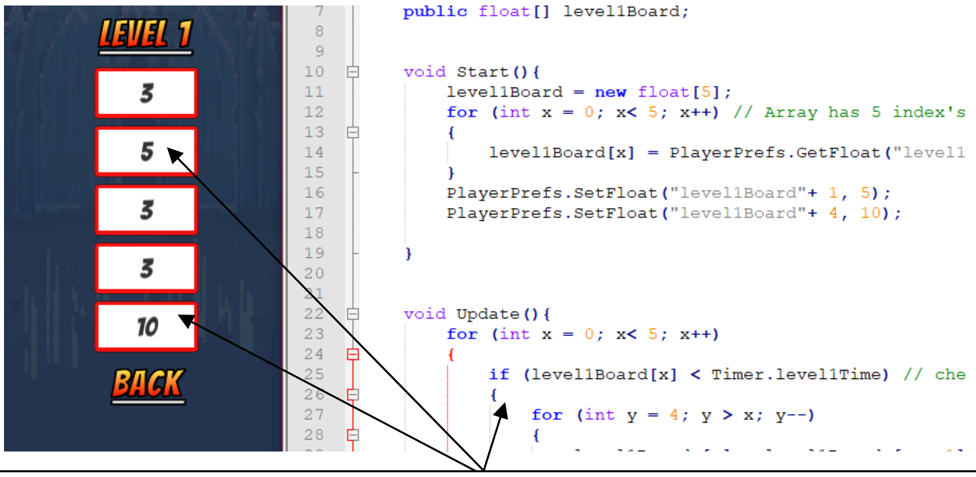
To display the PlayerPrefs in the leadreboard text boxes, another script is needed. This will load in all the 5 PlayerPrefs and then set them to the Text.



```
public class Leaderboard : MonoBehaviour {
    public Text[] topScores;

    float[] topTimes;


    void Start () {

        Debug.Log(PlayerPrefs.GetFloat("level1Board" + 1));


        topTimes = new float[5];
        for (int x = 0; x< 5; x++)
        {
            topTimes[x] = PlayerPrefs.GetFloat("level1Board");
            outputScores();
        }
    }

    void outputScores(){
        for (int x = 0; x < 5; x++)
        {
            topScores [x].text = topTimes [x].ToString();
        }
```

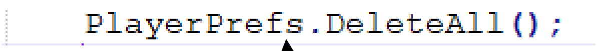
The code is very similar to the other script as the Start function calls in the PlayerPrefs the same way and sets each PlayerPrefs to an array called "topTimes". Then is calls the "outputScores" function which sets each array "topTime" index value to a string and then to equal the text "topScores". This will mean that the Text will now display the values originally from the PlayerPrefs.

As shown above, I haven't added in any times to the PlayerPrefs and so they all are set to 0 which is equivalent to null.



```
 7      public float[] level1Board;
 8
 9
10      void Start(){
11          level1Board = new float[5];
12          for (int x = 0; x< 5; x++) // Array has 5 index's
13          {
14              level1Board[x] = PlayerPrefs.GetFloat("level1
15          }
16          PlayerPrefs.SetFloat("level1Board"+ 1, 5);
17          PlayerPrefs.SetFloat("level1Board"+ 4, 10);
18
19      }
20
21
22      void Update(){
23          for (int x = 0; x< 5; x++)
24          {
25              if (level1Board[x] < Timer.level1Time) // che
26              {
27                  for (int y = 4; y > x; y--)
28                  {
```

I then added some values to the PlayerPrefs as shown on lines 16 and 17. I've set PlayerPrefs1 to 5 and PlayerPrefs4 to 10 and as shown index 1 displays 5 and index 4 displays 10. The other values I've added in also.

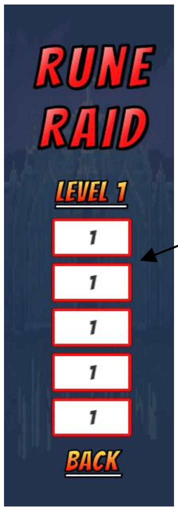


```
PlayerPrefs.DeleteAll();
```

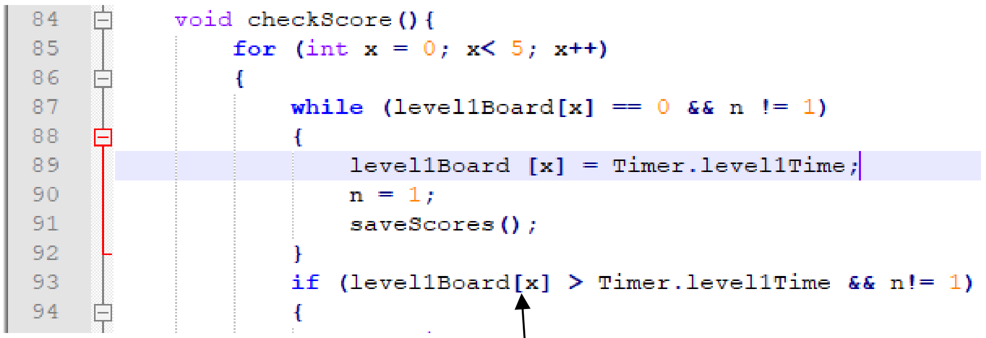
This is the line needed to delete all data stored in all PlayerPrefs which I needed when I wanted to test.

Testing

When I tested all the scripts by putting in a new time, it came up with a problem, this was that it set all the values to the new time because all the values to start if were 0 and also if index 0 was larger than the time it would change it properly but then loop again for each of the other indexes as well.



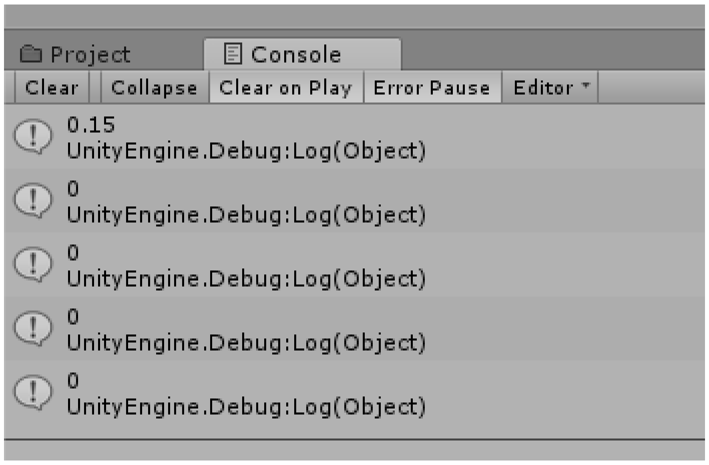
Shown here that when the value 1 was added and ran through the scripts, it set all the indexes to the same value instead of only index 0.



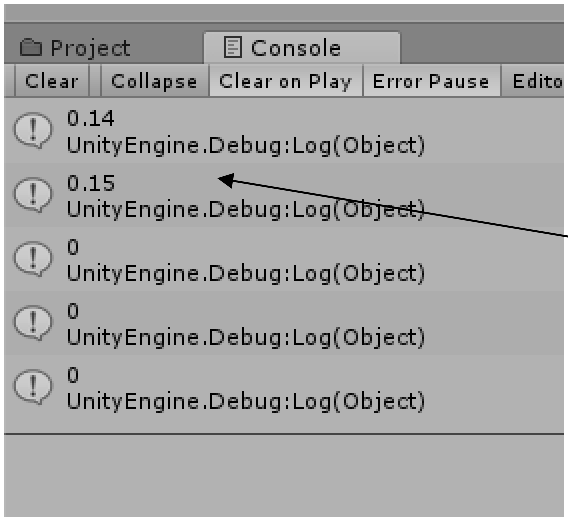
```
84      void checkScore(){
85          for (int x = 0; x< 5; x++)
86          {
87              while (level1Board[x] == 0 && n != 1)
88              {
89                  level1Board [x] = Timer.level1Time;
90                  n = 1;
91                  saveScores();
92              }
93              if (level1Board[x] > Timer.level1Time && n!= 1)
94              {
```

To fix this I added a while loop on line 87 to 91 in the "checkScore" function. This is so that if there is an array index value equal to 0 it just straight up sets the new score to that index because the score is going to need to be added as there are still empty indexes in the top 5. Then it sets the integer n to equal 1 so that the loop isn't run again, this is to prevent the same time being set more than once.

I also made the code output every index value for this testing part and as shown, when a new time is added to an empty list it only adds it to the index 0.



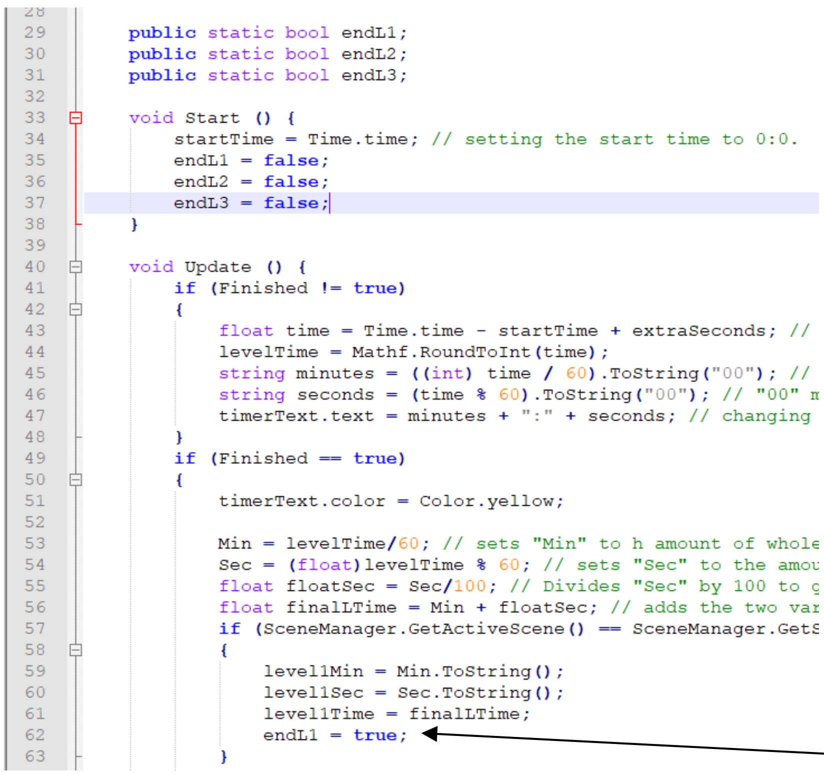
```
0.15
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
```



```
0.14
UnityEngine.Debug:Log(Object)
0.15
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
0
UnityEngine.Debug:Log(Object)
```

Also showing that adding another value also only adds it to one index

Calling in New Time

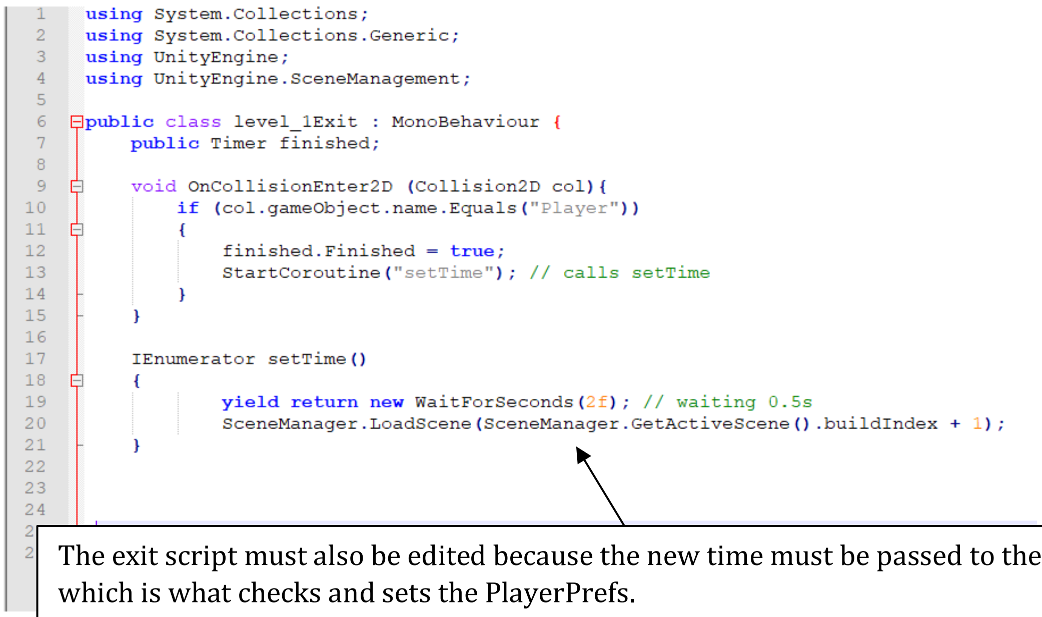


```
    public static bool endL1;
    public static bool endL2;
    public static bool endL3;

    void Start () {
        startTime = Time.time; // setting the start time to 0:0.
        endL1 = false;
        endL2 = false;
        endL3 = false;
    }

    void Update () {
        if (Finished != true)
        {
            float time = Time.time - startTime + extraSeconds; //
            levelTime = Mathf.RoundToInt(time);
            string minutes = ((int) time / 60).ToString("00"); //
            string seconds = (time % 60).ToString("00"); // "00" m
            timerText.text = minutes + ":" + seconds; // changing
        }
        if (Finished == true)
        {
            timerText.color = Color.yellow;

            Min = levelTime/60; // sets "Min" to h amount of whole
            Sec = (float)levelTime % 60; // sets "Sec" to the amou
            float floatSec = Sec/100; // Divides "Sec" by 100 to g
            float finalLTime = Min + floatSec; // adds the two var
            if (SceneManager.GetActiveScene() == SceneManager.GetS
            {
                level1Min = Min.ToString();
                level1Sec = Sec.ToString();
                level1Time = finalLTime;
                endL1 = true;
            }
```

To allow the leaderboard to function, a new time must be passed over to check and then add it in if necessarily.

To do this I needed to adapt the Timer script so that when the player finalises their time, it sets a Boolean static variable to true so that the checking/adding scripts can be called to execute.

This "endL1" bool is set to true which tells the other script to execute with the level 1 new time and through the level 1 leaderboard.

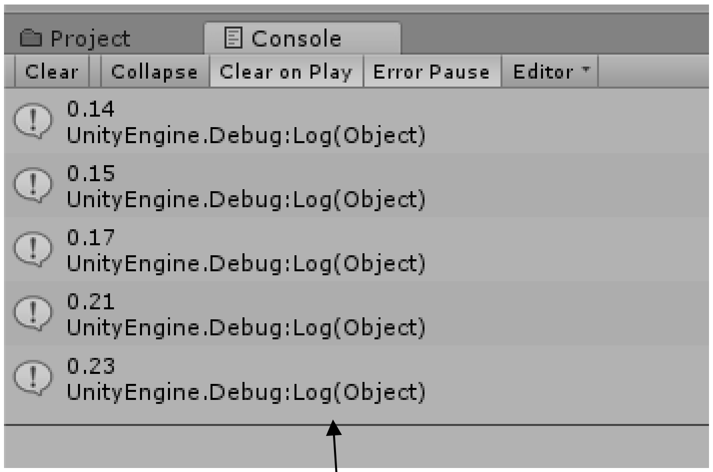


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class level_1Exit : MonoBehaviour {
    public Timer finished;

    void OnCollisionEnter2D (Collision2D col){
        if (col.gameObject.name.Equals("Player"))
        {
            finished.Finished = true;
            StartCoroutine("setTime"); // calls setTime
        }
    }

    IEnumerator setTime()
    {
        yield return new WaitForSeconds(2f); // waiting 0.5s
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
```

The exit script must also be edited because the new time must be passed to the "leader" script which is what checks and sets the PlayerPrefs.

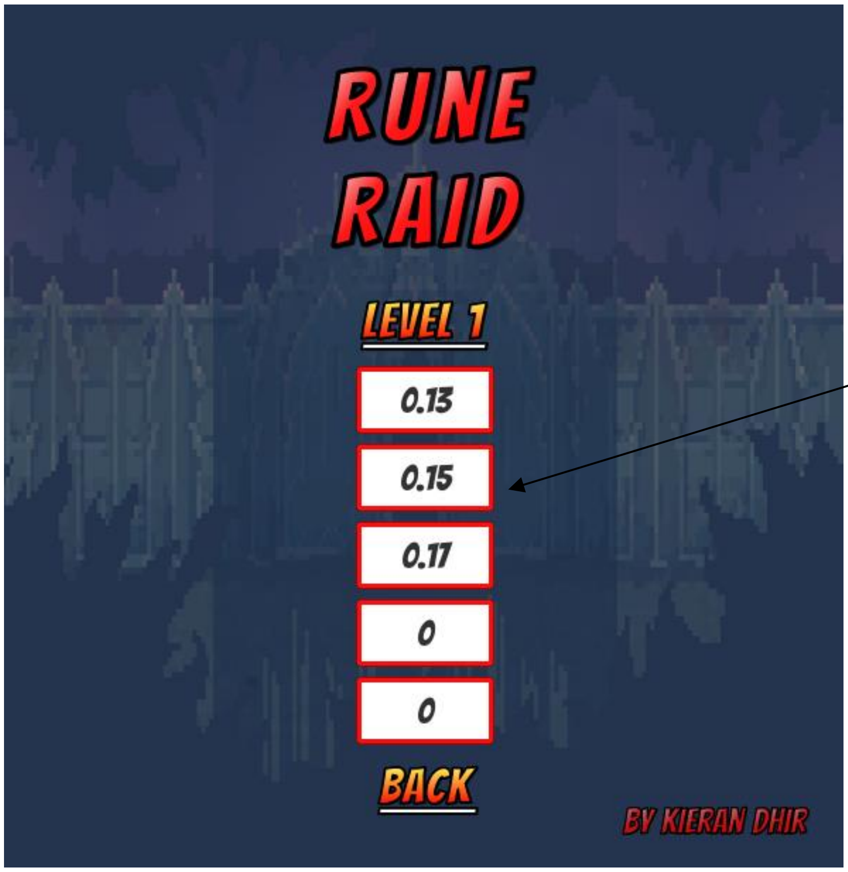
However, it can't do this if the net scene is immediately loaded because it will lose the chance to pass and run the new time through the scripts. Therefore, I've added in a delay that waits 2 seconds before then next scene is loaded, this allows enough time for the new time to be passed and run through the scripts.

I then just made the scripts run when the "endL1" is equal to true.



```
0.14
UnityEngine.Debug:Log(Object)
0.15
UnityEngine.Debug:Log(Object)
0.17
UnityEngine.Debug:Log(Object)
0.21
UnityEngine.Debug:Log(Object)
0.23
UnityEngine.Debug:Log(Object)
```

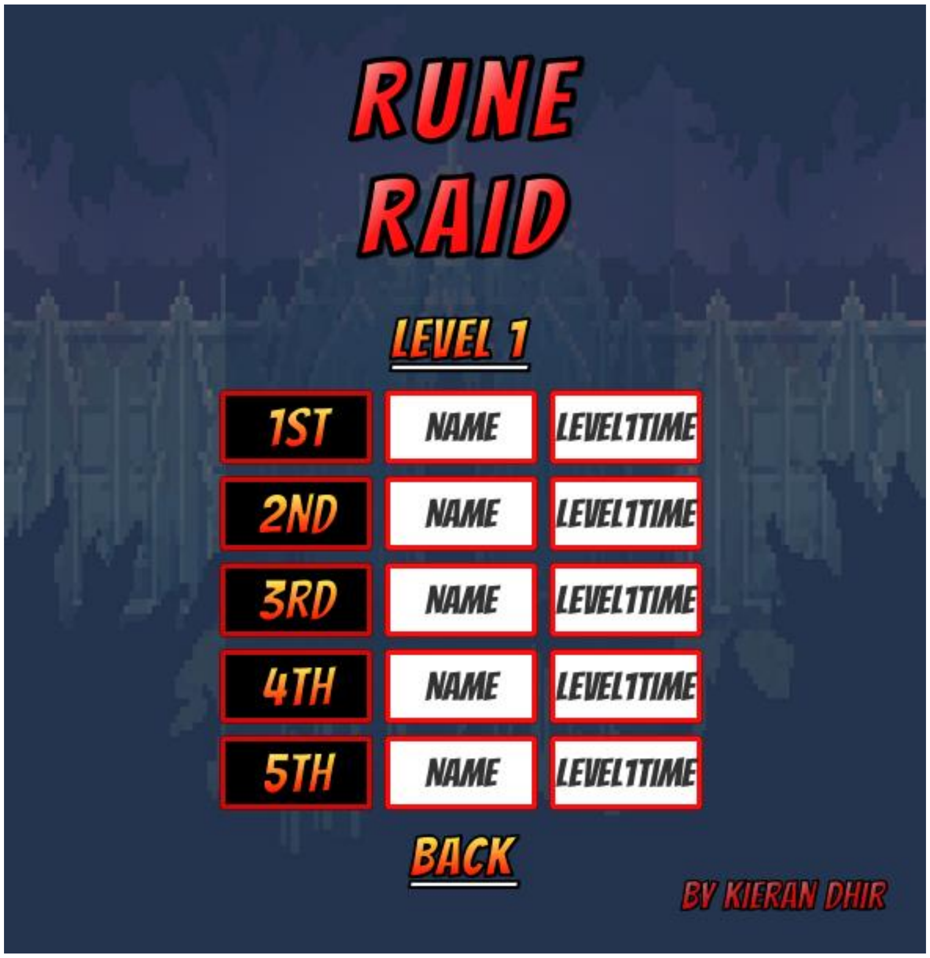
As shown here, I played through the level and finished 5 times and all the time scores were correctly added in with the correct order.



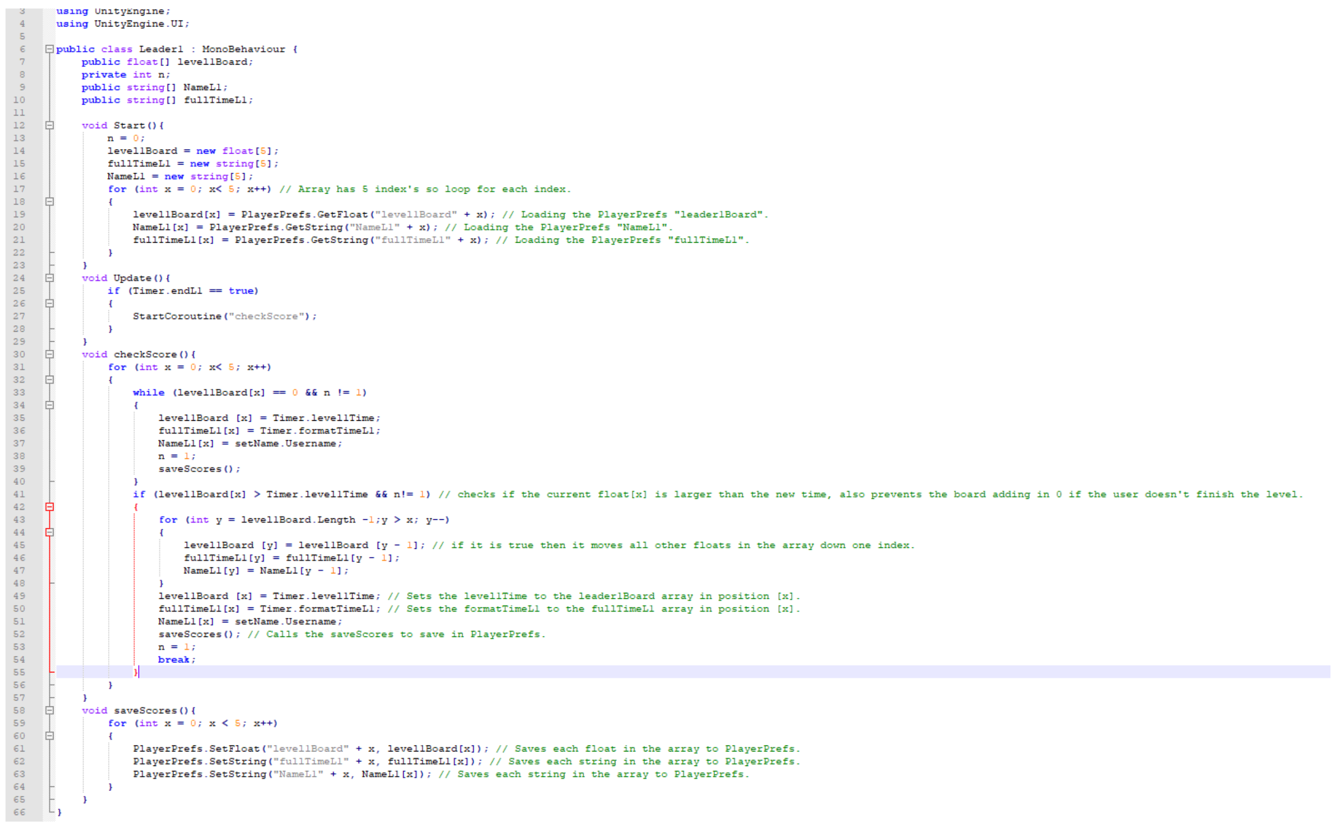
Showing the times being correctly displayed in order.

New PlayerPrefs

Next I needed to correct the formatting so that the Times are in the format of 00:00 and also have it with the accounts names next to the times to show which time belongs to which account.



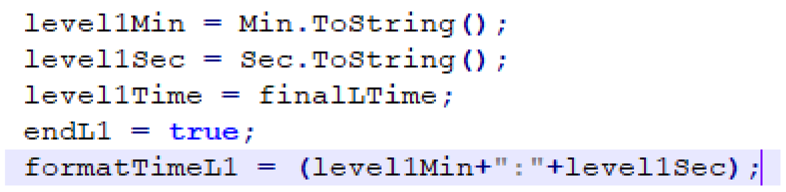
The design I created for the account names as well as static place position text.



```
using UnityEngine;
using UnityEngine.UI;

public class Leader1 : MonoBehaviour {
    public float[] level1Board;
    private int n;
    public string[] NameL1;
    public string[] fullTimeL1;

    void Start(){
        n = 0;
        level1Board = new float[5];
        fullTimeL1 = new string[5];
        NameL1 = new string[5];
        for (int x = 0; x< 5; x++) // Array has 5 index's so loop for each index.
        {
            level1Board[x] = PlayerPrefs.GetFloat("level1Board" + x); // Loading the PlayerPrefs "leader1Board".
            NameL1[x] = PlayerPrefs.GetString("NameL1" + x); // Loading the PlayerPrefs "NameL1".
            fullTimeL1[x] = PlayerPrefs.GetString("fullTimeL1" + x); // Loading the PlayerPrefs "fullTimeL1".
        }
    }
    void Update(){
        if (Timer.endL1 == true)
        {
            StartCoroutine("checkScore");
        }
    }
    void checkScore(){
        for (int x = 0; x< 5; x++)
        {
            while (level1Board[x] == 0 && n != 1)
            {
                level1Board [x] = Timer.level1Time;
                fullTimeL1[x] = Timer.formatTimeL1;
                NameL1[x] = setName.Username;
                n = 1;
                saveScores();
            }
            if (level1Board[x] > Timer.level1Time && n!= 1) // checks if the current float[x] is larger than the new time, also prevents the board adding in 0 if the user doesn't finish the level.
            {
                for (int y = level1Board.Length -1;y > x; y--)
                {
                    level1Board [y] = level1Board [y - 1]; // if it is true then it moves all other floats in the array down one index.
                    fullTimeL1[y] = fullTimeL1[y - 1];
                    NameL1[y] = NameL1[y - 1];
                }
                level1Board [x] = Timer.level1Time; // Sets the level1Time to the leader1Board array in position [x].
                fullTimeL1[x] = Timer.formatTimeL1; // Sets the formatTimeL1 to the fullTimeL1 array in position [x].
                NameL1[x] = setName.Username;
                saveScores(); // Calls the saveScores to save in PlayerPrefs.
                n = 1;
                break;
            }
        }
    }
    void saveScores(){
        for (int x = 0; x < 5; x++)
        {
            PlayerPrefs.SetFloat("level1Board" + x, level1Board[x]); // Saves each float in the array to PlayerPrefs.
            PlayerPrefs.SetString("fullTimeL1" + x, fullTimeL1[x]); // Saves each string in the array to PlayerPrefs.
            PlayerPrefs.SetString("NameL1" + x, NameL1[x]); // Saves each string in the array to PlayerPrefs.
        }
    }
}
```

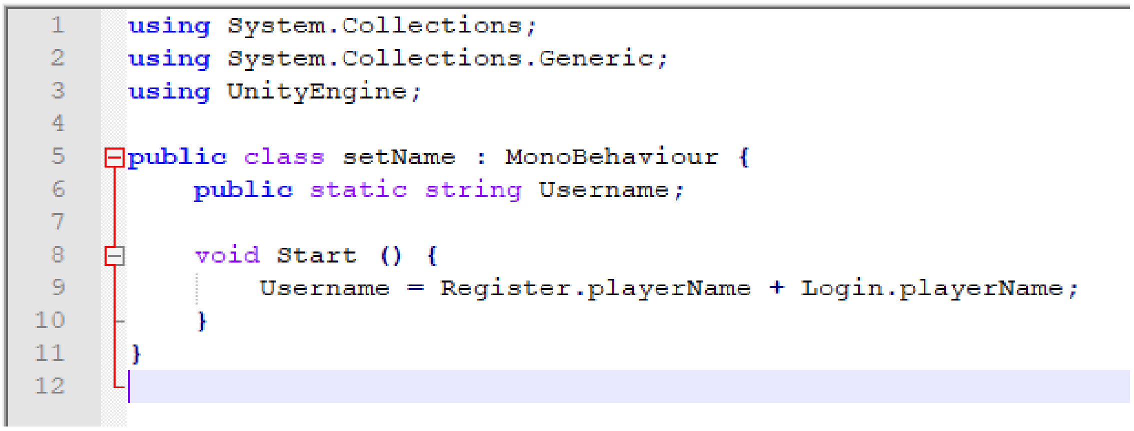
I just quickly added in two other PlayerPrefs which are the Names and the Formatted Time.

It calls the Name which is taken from the "setName" script and then takes the Formatted Time from the "Timer" script



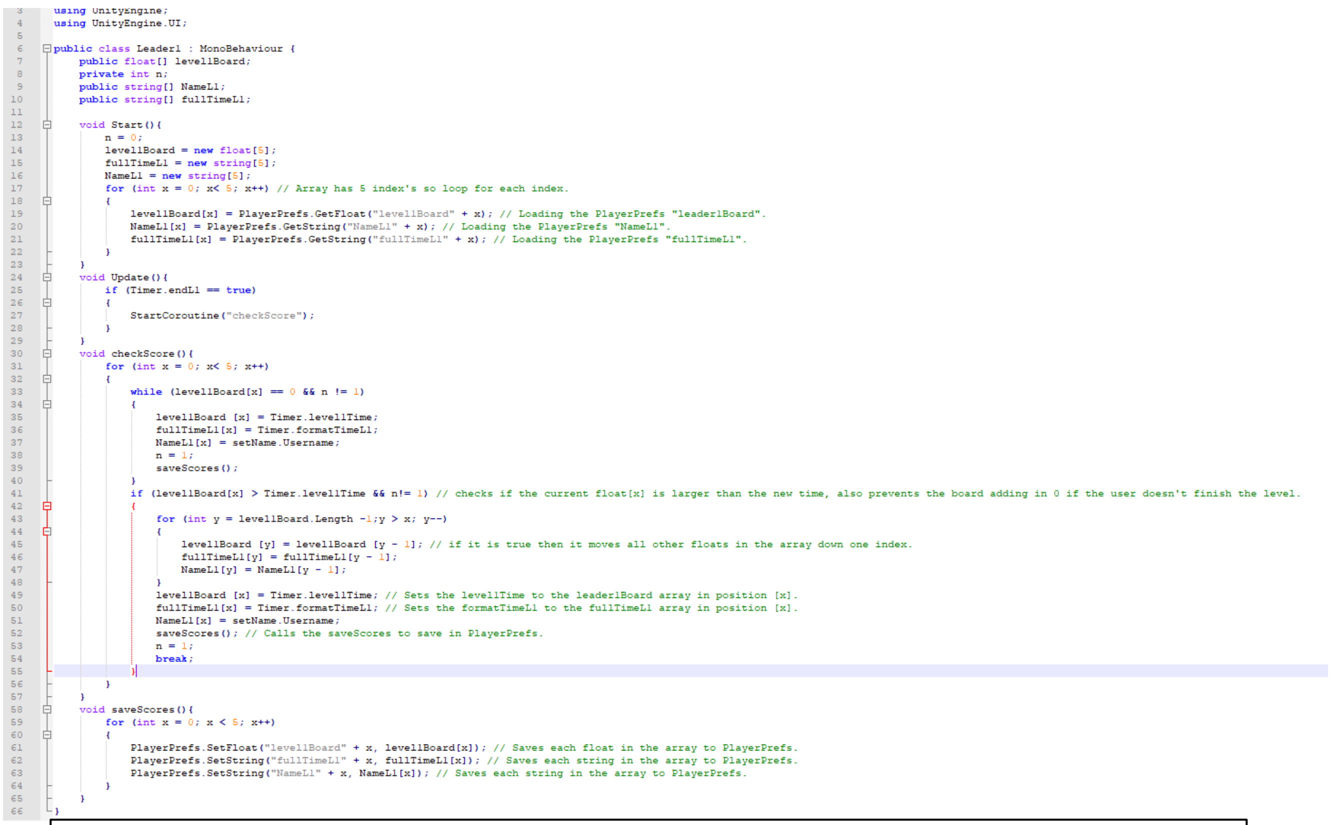
```
level1Min = Min.ToString();
level1Sec = Sec.ToString();
level1Time = finalLTime;
endL1 = true;
formatTimeL1 = (level1Min+":"+level1Sec);
```

This "formatTimeL1" is just the "Min" and "Sec" integers converted to a sting and concatenated with the ":" in the middle.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class setName : MonoBehaviour {
    public static string Username;

    void Start () {
        Username = Register.playerName + Login.playerName;
    }
}

```

This is just a small script that creates a static string called “Username” and is the Registers playerName and the Login playerName added together. This is because one will be null and the other will be the current accounts username because the user either logins in via the register script or the login script.



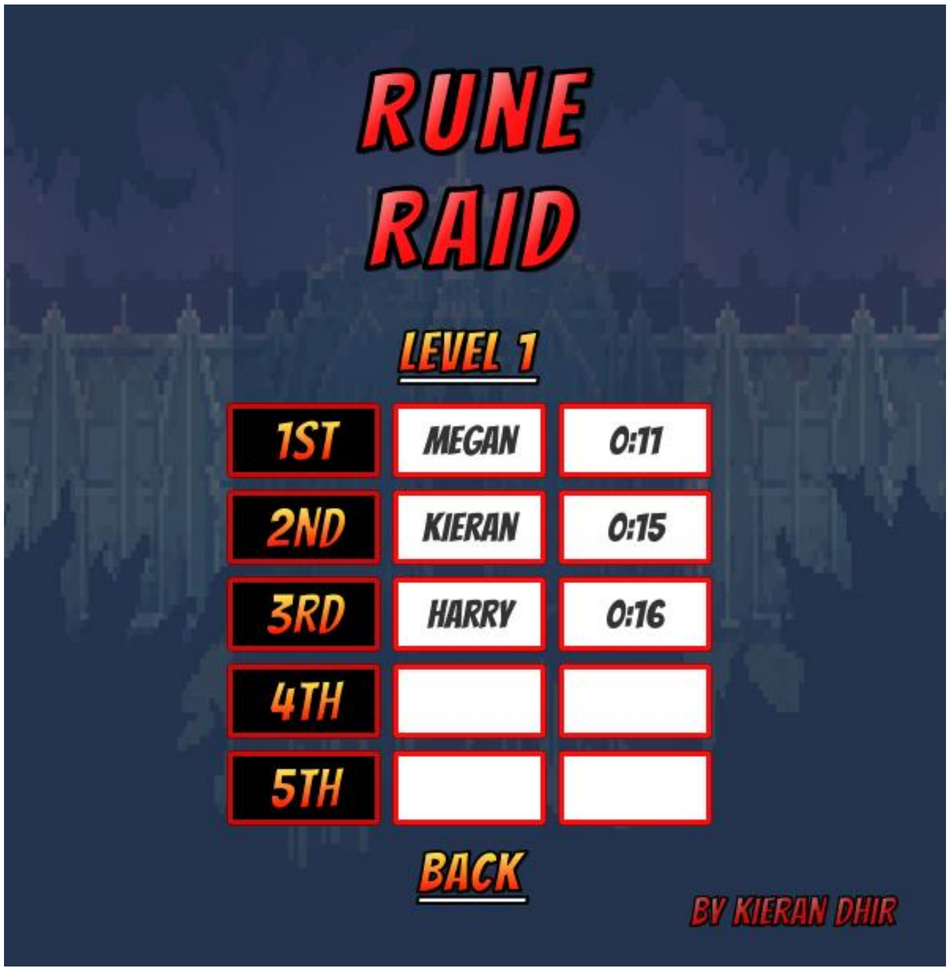
```
using UnityEngine;
using UnityEngine.UI;

public class Leader1 : MonoBehaviour {
    public float[] level1Board;
    private int n;
    public string[] NameL1;
    public string[] fullTimeL1;

    void Start(){
        n = 0;
        level1Board = new float[5];
        fullTimeL1 = new string[5];
        NameL1 = new string[5];
        for (int x = 0; x< 5; x++) // Array has 5 index's so loop for each index.
        {
            level1Board[x] = PlayerPrefs.GetFloat("level1Board" + x); // Loading the PlayerPrefs "leader1Board".
            NameL1[x] = PlayerPrefs.GetString("NameL1" + x); // Loading the PlayerPrefs "NameL1".
            fullTimeL1[x] = PlayerPrefs.GetString("fullTimeL1" + x); // Loading the PlayerPrefs "fullTimeL1".
        }
    }
    void Update(){
        if (Timer.endL1 == true)
        {
            StartCoroutine("checkScore");
        }
    }
    void checkScore(){
        for (int x = 0; x< 5; x++)
        {
            while (level1Board[x] == 0 && n != 1)
            {
                level1Board [x] = Timer.level1Time;
                fullTimeL1[x] = Timer.formatTimeL1;
                NameL1[x] = setName.Username;
                n = 1;
                saveScores();
            }
            if (level1Board[x] > Timer.level1Time && n!= 1) // checks if the current float[x] is larger than the new time, also prevents the board adding in 0 if the user doesn't finish the level.
            {
                for (int y = level1Board.Length -1;y > x; y--)
                {
                    level1Board [y] = level1Board [y - 1]; // if it is true then it moves all other floats in the array down one index.
                    fullTimeL1[y] = fullTimeL1[y - 1];
                    NameL1[y] = NameL1[y - 1];
                }
                level1Board [x] = Timer.level1Time; // Sets the level1Time to the leader1Board array in position [x].
                fullTimeL1[x] = Timer.formatTimeL1; // Sets the formatTimeL1 to the fullTimeL1 array in position [x].
                NameL1[x] = setName.Username;
                saveScores(); // Calls the saveScores to save in PlayerPrefs.
                n = 1;
                break;
            }
        }
    }
    void saveScores(){
        for (int x = 0; x < 5; x++)
        {
            PlayerPrefs.SetFloat("level1Board" + x, level1Board[x]); // Saves each float in the array to PlayerPrefs.
            PlayerPrefs.SetString("fullTimeL1" + x, fullTimeL1[x]); // Saves each string in the array to PlayerPrefs.
            PlayerPrefs.SetString("NameL1" + x, NameL1[x]); // Saves each string in the array to PlayerPrefs.
        }
    }
}
```

This script is now called when the “endL1” is equal to true from the Timer script. Then the “checkScore” function is run which checks the “level1Board” times against the new time from the Timer script. If the new time is quicker than an index then it appends the new time formatted time instead of the new time float to the index and also the name all to the same index but different arrays.

The reason why I can't just format the float time is because to format it I must change it into a string and if it's a string I can't compare it to check if its smaller or larger than other times because they are string.

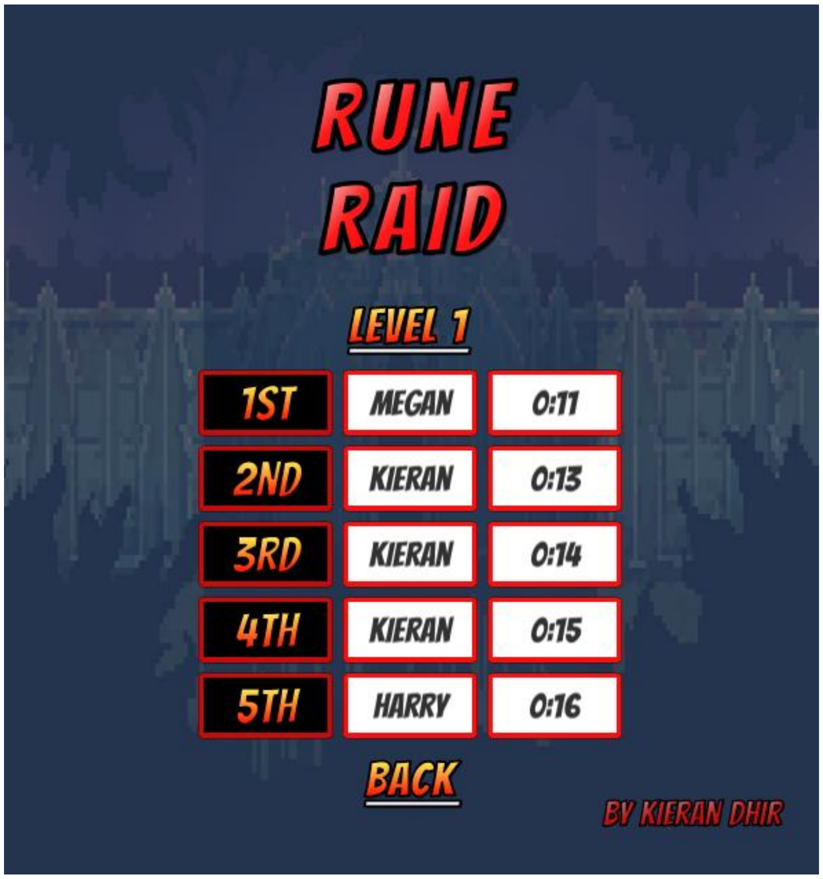
Therefore, that is the reason why I need another array for the formatted times because it compares with the float array and then sets the PlayerPrefs with the formatted array instead.

TESTING

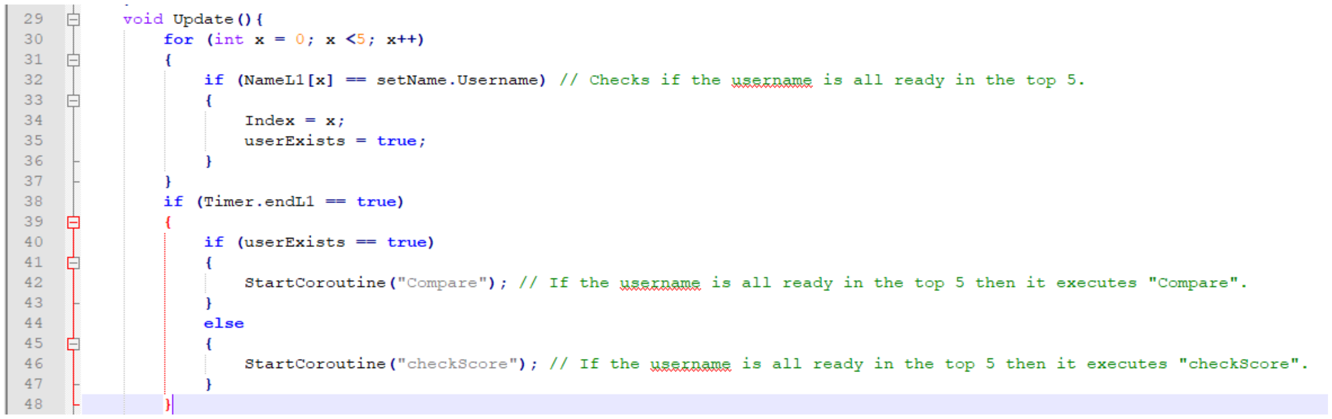


1ST	MEGAN	0:11
2ND	KIERAN	0:15
3RD	HARRY	0:16
4TH		
5TH		

As show here, the scripts are working and each account name is also added with its time which is formatted in the correct way. The order is also correct.



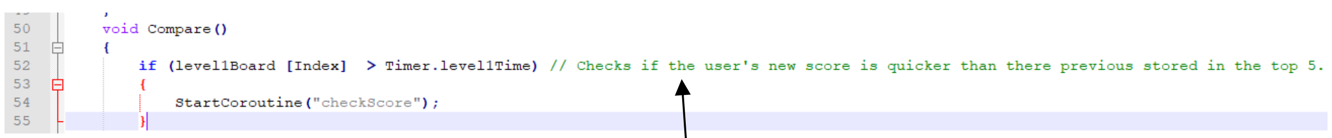
As shown above, the account Kieran has three times in the top 5 which I don't want because I only want one account to have one time in the top 5.



```
void Update(){
    for (int x = 0; x <5; x++)
    {
        if (NameL1[x] == setName.Username) // Checks if the username is all ready in the top 5.
        {
            Index = x;
            userExists = true;
        }
    }
    if (Timer.endL1 == true)
    {
        if (userExists == true)
        {
            StartCoroutine("Compare"); // If the username is all ready in the top 5 then it executes "Compare".
        }
        else
        {
            StartCoroutine("checkScore"); // If the username is all ready in the top 5 then it executes "checkScore".
        }
    }
```

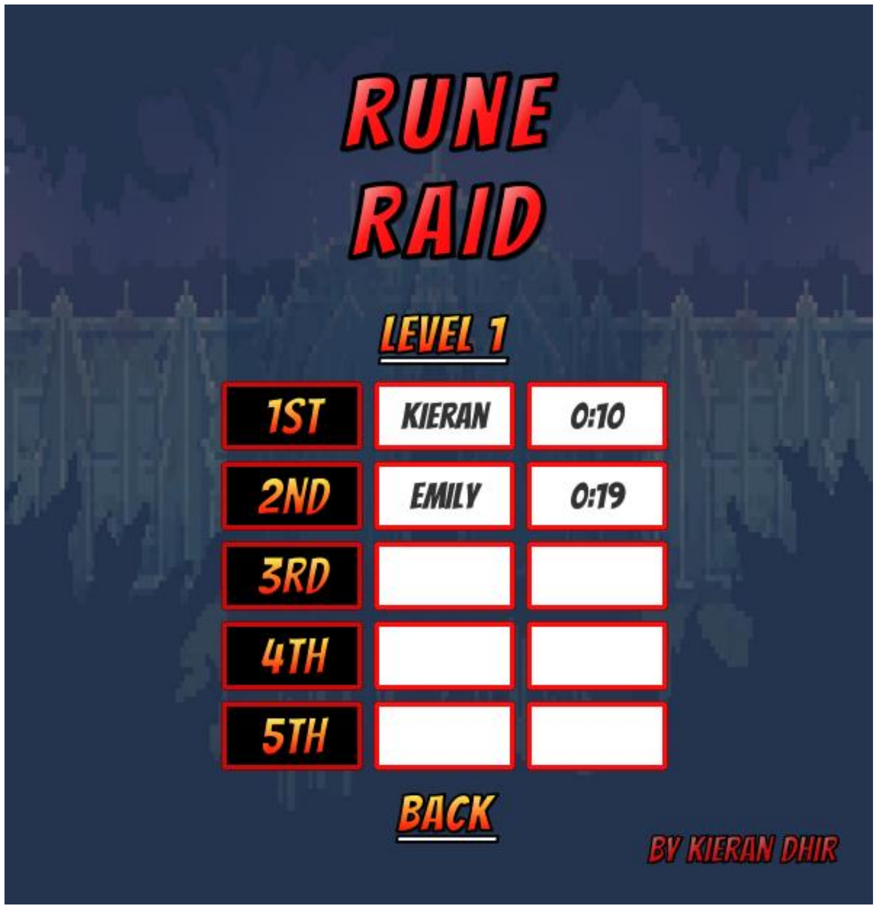
These IF statements are needed to prevent an account from taking its place in the top 5 multiple times.

First it checks if the Name array has the Username already in one of its indexes, if it does then it runs the Compare function. If the username isn't located in one of the 5 indexes then the "CompareScore" is called.



```
    void Compare()
    {
        if (level1Board [Index]  > Timer.level1Time) // Checks if the user's new score is quicker than there previous stored in the top 5.
        {
            StartCoroutine("checkScore");
        }
```

The compare function just checks whether the "level1Board" index with the accounts username in it, is larger than the new time being checked. If it is larger than it means that the accounts new timer is faster than the one stored and so they have beat their record and so it must be updated. If the stored time is quicker than the new time then there is no need to add the slower time in and so it doesn't do anything.

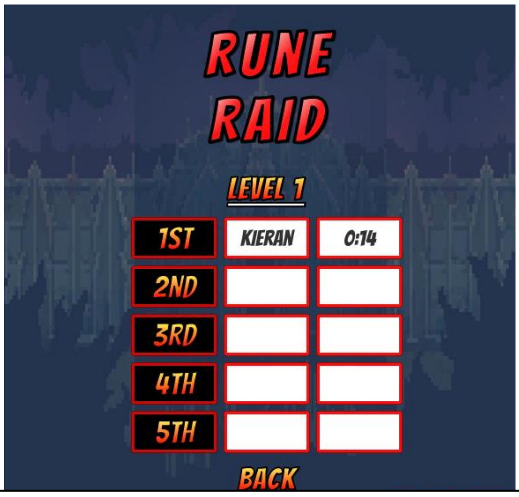


1ST	KIERAN	0:10
2ND	EMILY	0:19
3RD		
4TH		
5TH		

No an account can only hold one place in the top 5.

After all was working, I just copied the scripts and changed the variable names to adapt to level 2 and level 3 such as “level1Time” changed to “level2Time”.

I did this and all is working fine.



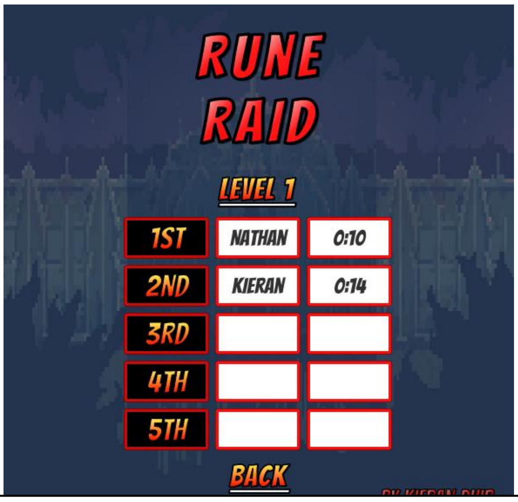
RUNE RAID

LEVEL 1

1ST	KIERAN	0:14
2ND		
3RD		
4TH		
5TH		

BACK

Showing an account time and name being added into the top 5.



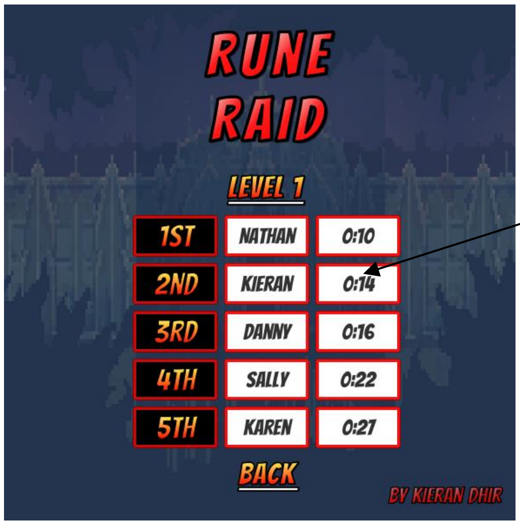
RUNE RAID

LEVEL 1

1ST	NATHAN	0:10
2ND	KIERAN	0:14
3RD		
4TH		
5TH		

BACK

Showing an account time and name being added into the top 5 above the previous 1srt place as it’s quicker.



RUNE RAID

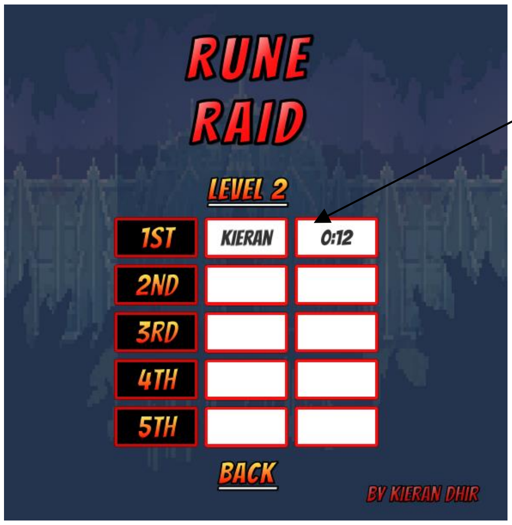
LEVEL 1

1ST	NATHAN	0:10
2ND	KIERAN	0:14
3RD	DANNY	0:16
4TH	SALLY	0:22
5TH	KAREN	0:27

BACK

BY KIERAN DHIR

Showing a full top 5 which is ordered correctly.

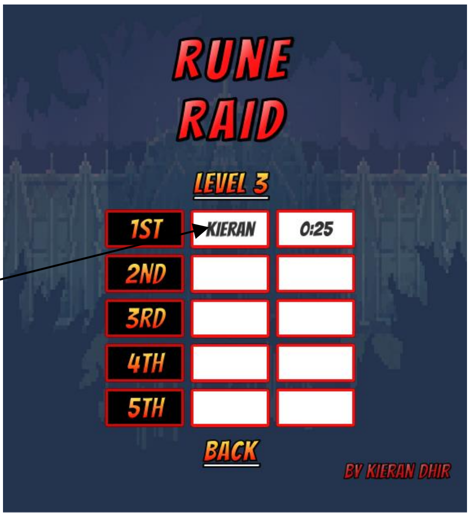

RUNE RAID

LEVEL 2

1ST	KIERAN	0:12
2ND		
3RD		
4TH		
5TH		

The level 2 top 5 working.

The level 3 top 5 working.


RUNE RAID

LEVEL 3

1ST	KIERAN	0:25
2ND		
3RD		
4TH		
5TH		

Meeting Success Criteria

I am confident that for the Leaderboard I've met all the success criteria relevant to this section of development that I have developed so far.

- Documentation of testing – I've clearly shown evidence of testing that I've done throughout the development of this scene and all the normal, boundary an extreme testing when applicable.

- Documentation of errors whilst in development – The leaderboard isn't an aspect that the user can interact with because it based on prior data. However, there are buttons that the user can navigate through the scene to look at Level 1 board, or level 2 board etc. However, the user can interact with the Leaderboard prior by getting a time within the top 5 range for a level.

- The player will be able to interact fully with every Rune Raid aspect – Every aspect I've set out to develop I've and ensured that the user can use it fully by testing them myself. However, I've not yet added a few aspects yet as I've discussed previously such as the timer and the Rune.

- All algorithms will be as efficient as I can possibly make them – The scripts I have created I believe are very efficient because they utilise loops to reduce lines of code and also only execute the loops between 1 to 5 times. I have made stopping conditions for the each loop so that they only loop when necessary to reduce unneeded processing,

- All decisions made will be shown and justified – Every significant decision I've made to this development I've fully recorded and justified such as, the reason why I chose to change the Leaderboard from recording every accounts fastest time, to only the top 5.

- All the code is split up into specific functions within modules – Every script has their own functions within them that are grouped effectively to by relevance and each different aspect has their own script such as the three Leaderboard and three Leader scripts.

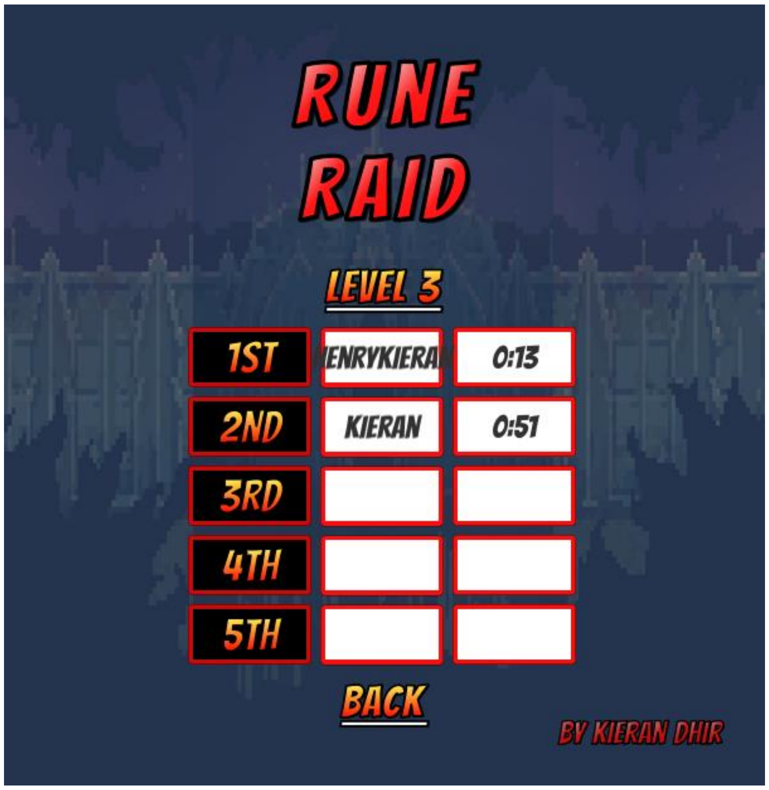
- Fully functional leader board that contains each player's best individual time score for each level – As I've mentioned above, I made the decision to alter this so that the Leaderboard only recorded the top 5 quickest time scores for each level. Therefore, I haven't full met this success criteria officially but I did explain why I made the decision not to. However, the top 5 times for each level are all fully working and when a new time is made by an account, the scripts do check it with the current top 5 scores to see if it needs to be added in and if it does then make the changes needed to implement the new time and alter the existing.

Beta Testing

Beta Testing

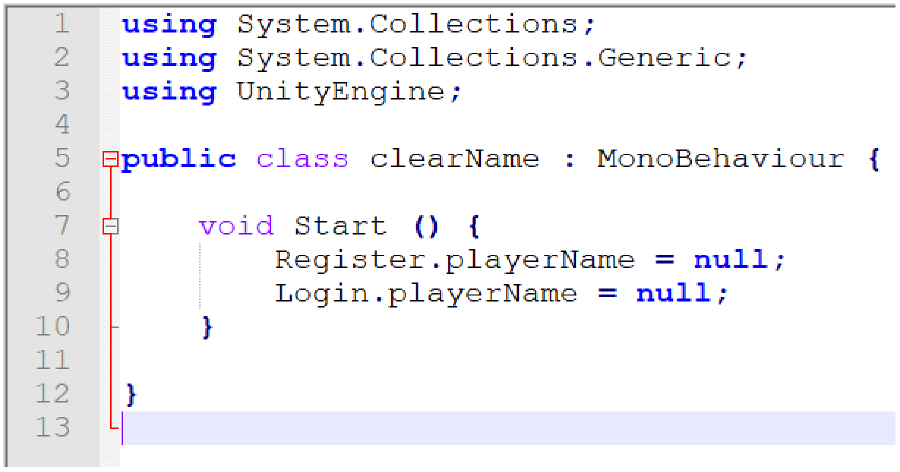
For the Beta testing I gave Rune Raid to some of my family and friends to see what they thought of the game and also to test for any bugs or undesired effects and then notify me on them.

Error



1ST	ENRYKIERA	0:13
2ND	KIERAN	0:51
3RD		
4TH		
5TH		

An error that was occurring was with the leaderboards where a username account stored on the board was a combination of two names. This occurred when the user played the Game by logging in either through the Login or Register and then after they finished the game, another user went back to the Game Menu and used the other login system (if the first user used login then the second used Register). Because the Username wasn't reset after the user went back to the Game Menu it meant that there could be one name for register and then one name for the login. And so when they are added together to form the Username it doesn't add a name and null but instead two names.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class clearName : MonoBehaviour {

    void Start () {
        Register.playerName = null;
        Login.playerName = null;
    }

}

```

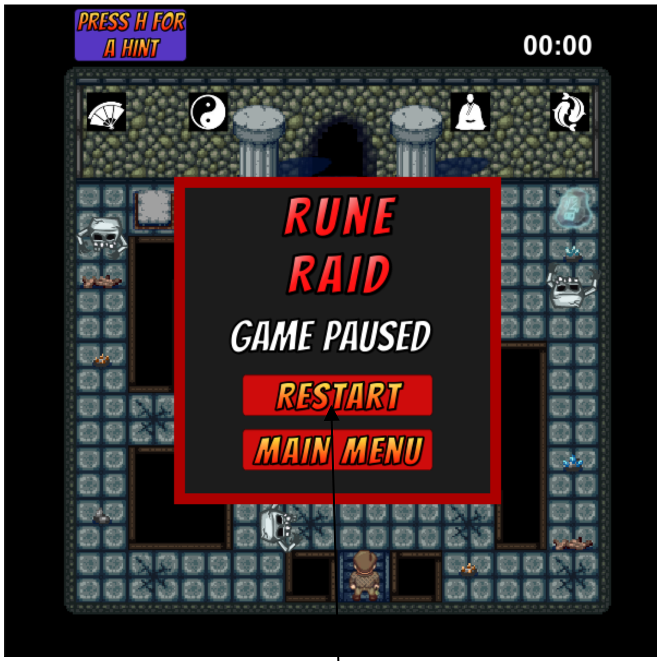
This is the small script needed to fix this problem. It clears both the Register playerName variable as well as the Login playerName variable when the scene with this script is loaded. I added it to the Game Menu scene and so when every time the Game Menu is loaded, both the name variables are cleared and so the “Username” is always one name added to null, preventing two names from being added together in the future.

WANTED ASPECTS

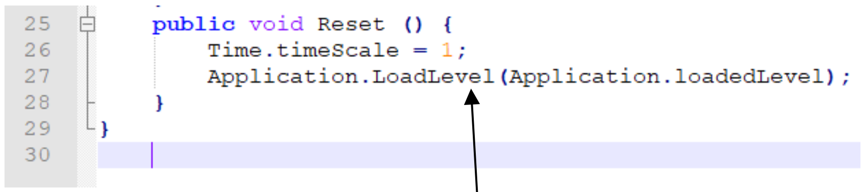
Restart Button

After my brother was playing Rune Raid he realised that he couldn't restart the level whilst playing if he wanted to because there was no option to restart the level from the beginning. He wanted this because for level 1 he pushed the pushing block against the wall and then couldn't continue with the level because you can't move it away from the wall and so he wanted to restart the level but couldn't.

I thought this as a very valid point and I proceeded to create a restart button which will be on the Pause Menu.



The restart button.



```
25     public void Reset () {
26         Time.timeScale = 1;
27         Application.LoadLevel(Application.loadedLevel);
28     }
29 }
30
```

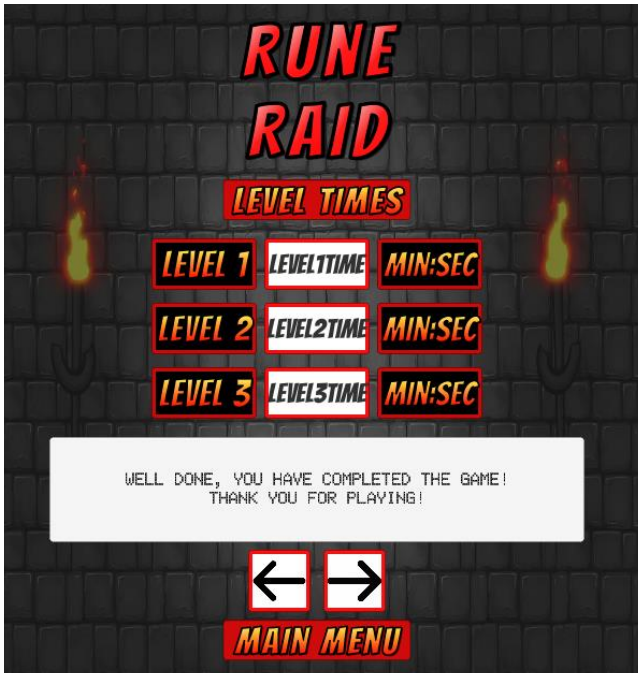
This is the script that is added to the Restart button "OnClick" component. It sets the Time.timeScale to equal 1 because in the pause menu it is equal to 0 and so must be set back to 1. Then line 27 reloads the current level but loading the loaded level.

Scores Ending Scene

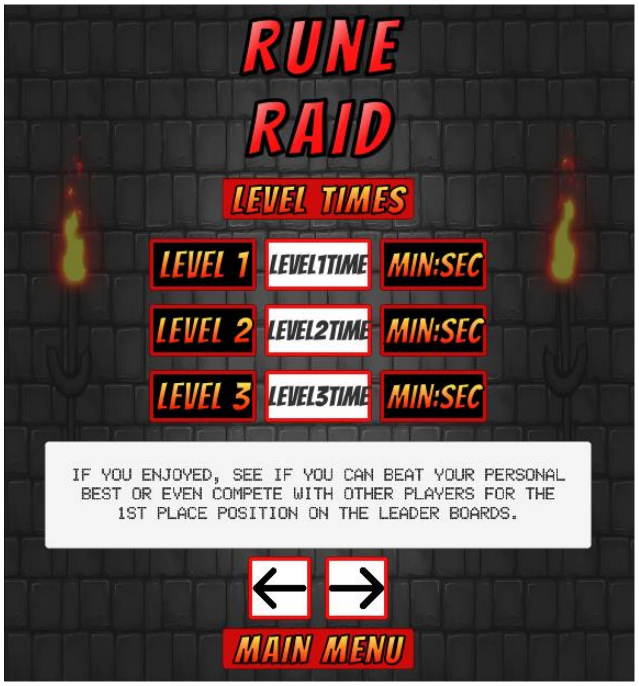
One of my friends that was beta testing Rune Raid said to me that it would be nice if there was another scene at the end that showed all the times that they got for that run-through and also some extra text saying that they have finished the game to prevent an abrupt ending.

I thought this was great idea because when the player completes a level they don't have much time to see what score they got and also if they didn't make it into the top 5 then there is no way of them finding out what times the active for the three levels. Also the extra text congratulating the player for completing the game is also a nice touch to add to Rune Raid to make it just that bit more user-friendly instead of the current abrupt ending of just going back to the Main Menu scene.

Design



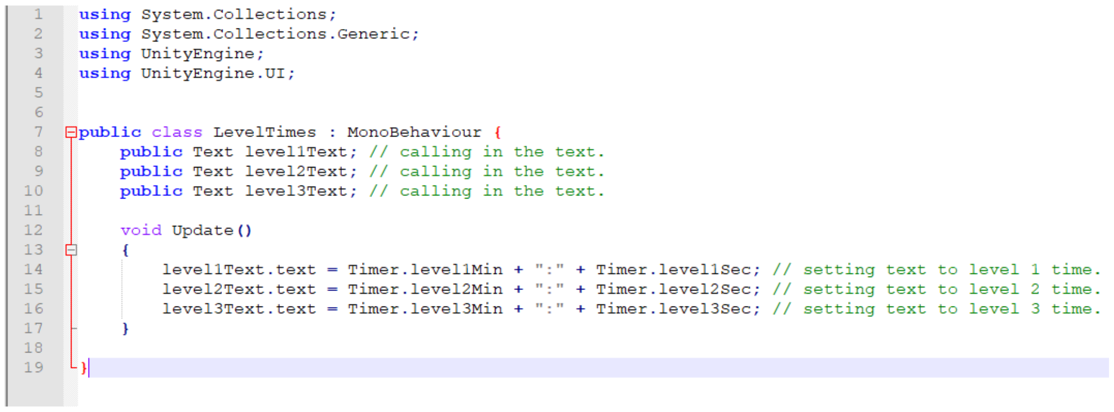
WELL DONE, YOU HAVE COMPLETED THE GAME!
THANK YOU FOR PLAYING!



IF YOU ENJOYED, SEE IF YOU CAN BEAT YOUR PERSONAL BEST OR EVEN COMPETE WITH OTHER PLAYERS FOR THE 1ST PLACE POSITION ON THE LEADER BOARDS.

This is the design I developed for the ending, I just copied over the GameObjects from the Leaderboard and the Storyline scenes for the buttons and text. The three middle GameObjects are the text boxes that will have the three times that the current account has just achieved in this run-through.

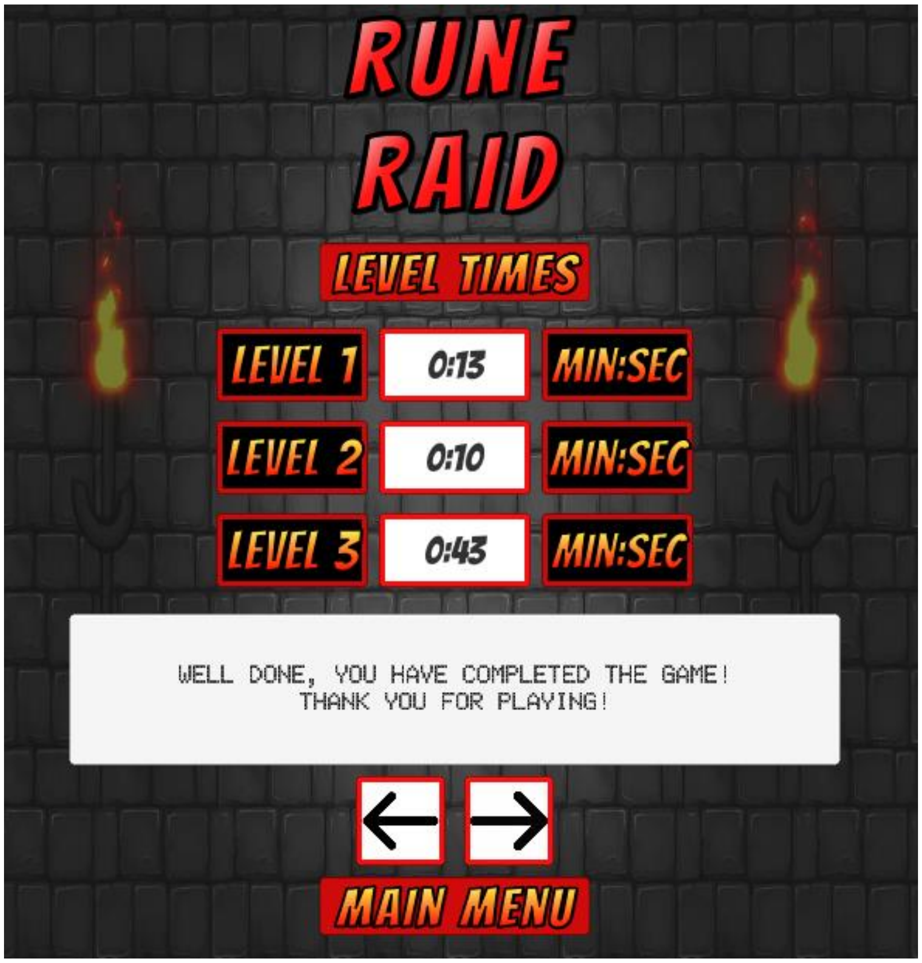
Script



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class LevelTimes : MonoBehaviour {
    public Text level1Text; // calling in the text.
    public Text level2Text; // calling in the text.
    public Text level3Text; // calling in the text.

    void Update()
    {
        level1Text.text = Timer.level1Min + ":" + Timer.level1Sec; // setting text to level 1 time.
        level2Text.text = Timer.level2Min + ":" + Timer.level2Sec; // setting text to level 2 time.
        level3Text.text = Timer.level3Min + ":" + Timer.level3Sec; // setting text to level 3 time.
    }

}
```

All I needed was this script which just takes each levels time from the Timer, it gets the "level1Min" and concatenates it with the "level1Sec" with a ":" in the middle. Then it sets that to the level1Text. It is the same for level 2 time and level 3 time.

Testing



LEVEL 1	0:13	MIN:SEC
LEVEL 2	0:10	MIN:SEC
LEVEL 3	0:43	MIN:SEC

WELL DONE, YOU HAVE COMPLETED THE GAME!
THANK YOU FOR PLAYING!

As shown, when I played through the entire game, completing each level, it loaded the "Ending" scene which displayed the times that I got for each level and the ones that were displayed were indeed the same times that I got for each level.

Overall feedback

After these changes were made, my Beta testers played Rune Raid again a few times to make sure there were no more errors and they found none. I asked them whether they liked the game and if they were pleased with the changes I've made due to their comments. They all said they were happy with the changes I've made and that the extra features were working great.

They also said that the levels were challenging, especially the level 3 puzzle which many took significantly more time to complete than the other two previous puzzles. They also said that they liked the hint system because it meant they there was an option for help if they got stuck which would prevent them from getting frustrated. As I final say, they all very much liked the retro-cartoon style design and said it was very appealing.

Evaluation

Evaluation

My Success Criteria

Success criteria	Fully/partially or unmet?	How to improve in further development
Fully functional leader board that contains each player's best individual time scores for each level.	The Leaderboard is complete but it isn't what I first set myself to make. At first as shown by the success criteria, I wanted to have the Leaderboard store every accounts fastest time no matter the position. However, I've only developed a Leaderboard with 5 places and so not every accounts fastest time is stored. I did however, did explain my reasoning for this change and the change that I made I've fully completed. (To see my reasoning go to Leaderboard Design). I've tested it multiple times as well as the Beta tester and to ensure that there are no errors we only found one, which I have fixed and so I'm confident that it is fully working to its new specifications.	In further development I could indeed store every accounts fastest time scores for each level. To do this I would have to create a scrolling mechanism where the player could scroll up and down the Leaderboard scene because as I've shown before, I can't fit any more than 5 place positons with the current design. I would also have to edit the code to be able to adjust to have an unlimited array of any size which is quite simple because instead of the array being a fixed size of 5 it will just be a changing integer which can be added to when a new account is added. Also I would have to make a script that would create the 4 text boxes position, name, time and Min:Sec each time a new account is created.
Rune Raid must have three unique levels for each player to play on.	I have successfully created 3 levels that the player can play on. I even when further and created 3 extra storyline scenes as well as an ending scene which all coincide with each other. Therefore, I believe I have fully met this success criteria.	N/A
A user friendly login interface.	I believe I have created a very friendly login interface which is very easy to use and the text and buttons are large so that any user should be able to see/read them. Many of my friends that I have shown the login interface to, said they liked the design and thought that it is easy to navigate and use.	N/A

	Also, when registering the user is notified when they are trying to create an invalid account and the reason why it is invalid. This is also the same for the login where it notifies the user if they input incorrect data for an account. Therefore, I believe I have fully met this success criteria.	
Score Timer.	The Timer I have fully implemented into each of the three levels and are clearly shown on the top right on each level. I have implemented Runes that successfully deduct time from the Timer to act as a reward and I have tested it to ensure that it is working, (go to Rune to see this). Also it turns it green for a second to ensure that the user knows that time has been deducted. I have also implemented monsters (skeletons) on level three. If the Player collides with a skeleton then it adds 10 seconds to the Timer successfully and also turns it red for a second to ensure that the user knows that time has been added. When the player reaches the exit, the Timer successfully stops and turns yellow to indicate that it has stopped. It then is set to multiple variables which then are used in many other scripts throughout the game. Therefore, I believe I have fully met this success criteria.	N/A
Documentation of error trapping.	I am confident that I have documented all error trapping where it was necessary. The majority of Rune Raid didn't actually require any error trapping because there was only	N/A

	a few controlled inputs that the user could input. However, for the aspects that did require error trapping I am confident that I implemented it successfully. A major example is the Login and Register scenes which require a lot of error trapping to prevent invalid data from being inputted by the player. Such as, inputting a username that doesn't exist in the Login section or making a username that is the same as their password for the Register section. It tells the user that what they are doing is invalid and then resets the required variables to allow the user to input another Username&/or Password. Therefore, I believe I have fully met this success criteria.	
Documentation of testing.	I am 100% certain that I have documented testing throughout my development and Beta testing stages. I have clearly stated where I have been testing Rune Raid, the results of the tests and then the repercussions of them. When I tested I tested normal data, boundary data and extreme data to ensure that all data inputted would work in the desired way. I have clearly done this throughout and so I am confident that I have met this success criteria.	N/A
Documentation of errors whilst in development.	Every error that I came across whilst in development or beta testing, I have clearly documented by showing the error, explaining the reason for it, explaining the way I'm going approach the problem and then showing the fix and results of	N/A

	the fix. Also, I have referenced any sources that I used to help me find a solution for an error. Therefore, I believe I have fully met this success criteria.	
The player will be able to fully interact with all of Rune Raid's aspects.	Every aspect that I have implemented into Rune Raid I have ensured that the user can interact with it in the desired way. I have done this by testing rigorously every aspect of Rune Raid and also my Beta testers have helped me do this as well. Examples are the Runes that the player can collect throughout each level that deduct time of their Timer. Another is the pause menu that the user can access at any time during each of the three levels where from there they can walk away from their keyboard to do another task, restart the level if they feel they need to and finally return to the Main Menu if they wish. There are many other aspects that I have implemented in Rune Raid other than these two that I have mentioned. I have documented and tested them all to the same degree to ensure that the user can fully interact with them. Therefore, I believe I have fully met this success criteria.	N/A
All algorithms will be as efficient as I can possibly make them.	Every algorithm that I have developed I have gone through multiple times, trying to find ways to make them more efficient and many times I have found more efficient ways, mainly through looping. Some of my complex scripts I did take from YouTube which I have referenced and so I know they are efficient. However, I did ensure that I adapted them to	With further development I could improve the efficiency of many of my algorithms due to having a lager time scale to research and develop more complex code. I could have also ran my scripts through interpreters to find where my algorithms were the most inefficient and improve on them.

	Rune Raid's environment and in many cases it wasn't a script that was perfect for Rune Raid and did required multiple adaptions to achieve my desired results. Although, I know that some of my scripts can be refined even more to be more efficient and so I can't say that they are of the best efficiency but I am confident that I have done my best to make them as efficient as I possibly can.	
All decisions made will be shown and justified.	I am 100% confident that I have documented every significant decision that I have had to make during this entire process and the reasoning behind it. For example, I have documented the reasoning why I have only made a top 5 Leaderboard instead of every accounts quickest time. Another, is the Pushing Block on level 1 where I had the problem with the 2D box colliders not working properly due to a fault with Unity. I explained the reason why it wasn't working and then my thought process on how I approached a comprised solution, which was the area effectors. Therefore, I believe I have fully met this success criteria.	N/A
Rune Raid's puzzles must be challenging for the player.	I personally believe that each of the level puzzles are challenging. The first level is relatively easy but still requires some thought on how to solve it and also there is a secret Rune hidden in a pot to add to the challenge of finding the Rune. The first level is meant to ease the user into the game and so it is not meant to be very hard.	With further development I could significantly increase the difficultly of the puzzles so that they were of the standard of many official puzzle games on the market. I could have done this by simply taking my time to design the level puzzles to ensure that they had a lot of complexity to them and multiple stages.

	The second level is much more difficult than the previously level 1 puzzle and so meets the design requirement of the level puzzles getting progressively harder. It requires high concentration to complete the puzzle because one second of the user not paying total attention could easily result in them getting the puzzle wrong and being forced to start again, missing out on the rune chest. The rune chest adds to the challenge because to be able to collect the rune the user must complete the three stages of the puzzle perfectly without any prior failures. The last puzzle on level three I believe is definitely a step up from the puzzle on level 2. This is because there are multiple aspects to the puzzle. One is that the user must avoid touching the skeletons that are walking around the level because if they touch them they will get a time penalty of 10 seconds. And so they must avoid the skeletons whilst trying to complete the puzzle at the same time. Then the puzzle itself definitely is a challenge because the majority of my beta testers didn't even think about reading the symbols backwards until very late into the level and so just shows that it is not as simple as it seems. With the time scale I have been given and resources, I believe I have fully met this success criteria to the best of my ability. However, I could have mad the puzzle even more challenging with further development.	However, the reason why this could only be done in further development was because I just didn't have enough time to make them any more challenging.

All the code is split up into specific functions within modules.	I have ensured that every aspect of Rune Raid has its own script and within the scripts the code is bundle up by relevance with other code to create the functions. This is to ensure that I can easily debug each function/script on its own to help with development and also make it easy for an outsider to read my code and understand it. I am confident that I have done this and as evidence, I have clearly documented over 20 different scripts and many more functions within them.	N/A

MAINTENANCE ISSUES AND LIMITATIONS

Maintenance Issues

Problem	Solution
Exponential accounts could lead to problems with the storing the data in files. This is because as the player base grows, the more storage is needed to store the account files which could lead to storage problems such as running out of storage. It could also lead to more unorganised storage of the files because currently they are all just stored in one file called UnityData.	Make sure that the storage space available is large enough to store a large amount account data and if it does reach maximum storage, invest in another storage device to add to the system to increase the storage space available. A better way to organise the account files is to assign each account file a unique number index so that they can be organised efficiently.
The files that are stored are all checked when the Register and Login scripts are run. This could lead to a major problem due to running time, as the amount of account files increase, the time it takes to check each file is Big O	To fix the issue of time complexity I could assign each account file a unique number index, order them in ascending or descending and then implement a binary search algorithm. This Binary Search algorithm can be used to search for the account files for the Register and Login scripts to

notation Log n^2 because each file isn't indexed via number but strings instead. This could lead to major very long processing times to Register a new account or Login a user, which could lead to the user becoming frustrated due to the long waits.	check whether an account exists for Register and Login and check password for Login. Using a Binary search instead will significant increase the time efficiency and therefore increased Rune Raid's performance. The search algorithm that I'm using is of time complexity of Log N^2 and a binary search is Log N which is more time efficient.
The more people that play Rune Raid the more likely that more errors occur due to the increase in different data inputs. This could lead to a major problem if there are a lot of hidden errors still within Rune Raid that managed to slip through my testing and the beta testing. This is because if there are a lot of errors that occur, it could take a very long time for just me to fix these problems and so could cause Rune Raid to be down for a long period of time, which may frustrate players.	I must be able to create a way for players to log errors that they come across. This is so that I can easily find out any errors that the player base is facing. This quick way to post errors will allow me to be able to rapidly respond to errors to try and create a patch as fast as possible. Also, to fix multiple errors at one time may take too much time for me and so I may need to enlist the help of other developers to share the work load and reduce the time it take me to release patches.

Limitations

Problem	Solution
Improving the performance of my algorithms by increasing the efficiency will require a lot of time and knowledge. For me as a sole developer it may just be too much work load for me to do in a feasible amount of time. This may result in slow releases of patches and updates which then may lead to the player base decreasing due to them becoming bored or fed up with errors that are not fixed.	To help share the work load I could enlist the help of other developers which could bring their own expertise to significantly decrease the time it takes to release patches and updates. This means that the player base may keep increasing due to errors being patched quickly and also new content being released at regular intervals.
Hardware limitations just the user's own local storage but if I wanted to convert the game from local to worldwide then it would require me to invest in hardware storage and specific processors to process every accounts data. However, as I'm just a student, I don't have the money to invest in this expanse of hardware and also I don't have the knowledge to implement these into a functional system.	I could try and create a fund me which would mean I would rely on donations from people to fund the hardware that I need to scale up Rune Raid to worldwide. I could also try and partner with a publishing company that would allow me to use their own hardware to store Rune Raid's data and they would handle the running of the systems and I would pay them a monthly fee.
As I only had a few beta testers, there still may be many hidden errors that have slipped	To fix this I could have enlisted the help of hundreds, even thousands of beta testers to

though the testing. Therefore, Rune Raid may actually have multiple errors in the first public release of the game.	ensure that the majority of errors within the game has been found and then patched. This is called User acceptance testing.
Due to time restraints I have not created an Operations manual which shows how the program works. This could lead to problems when other developers help me with the development and also trying to pitch to publishing companies.	If I had more time for developing I could have allocated time to ensure that I created an operations manual. I could also enlist the help of another person to help me with this if necessary.