

Game and interactive simulation systems

Grass Buddy 2D Game

Student: Catalina-Andreea Ampleev

Professor: Bujor PavaloIU

### 1. Game description

The “Grass Buddy” game based on horizontal 2D shooter games with infinite scrolling background, but in this case the character does not shoot. The game as it is right now does not have levels, just one continuous round until there is no more lives.

The scope of the game is to avoid the obstacles by using up and down movements.

### 2. Game Parameters

The game consists of the following parameters:

- The main character: one nice green character
- The fire obstacles: 3 obstacles prefabs positioned on 3 points of the game space
- A spawner for the fire obstacles randomly spawns the 3 obstacle prefabs
- The lives of the main character: there are 3 lives

### 3. Gameplay:

The first thing that is displayed on the screen when you open the game is the menu scene. It shows the name of the game, the high score and has a short Main Menu consisting of two buttons: a button to play the game and one to quit the game. When you press play, you are redirected in the middle of the actual Game Scene, where you need to move up and down to avoid the coming flames. Once you passed by a flame your score increases by one. Your score and your lives are displayed on the screen, also there is a button for returning to the Main Menu. You have three chances to get a higher score. If you will not be able to pass by the flames, you will lose a life. When you lose your last life, the game restart itself.

### 4. Personal contribution

The game design is made by me using Photoshop. I created the main character, the backgrounds (main menu and game scene), the flame, flower. I end up using a png flame from internet because it looked better, and the flower is to be implemented.

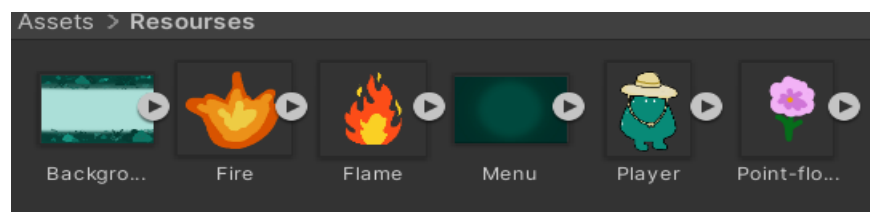


Fig.1. Created elements of the game

For the code and style of the game I used different tutorials for each part of the code and format and downloaded audio sources for background music and death sounds effect.

### 5. Game Design:

There are two scenes in the game: Menu Scene and Game Scene.

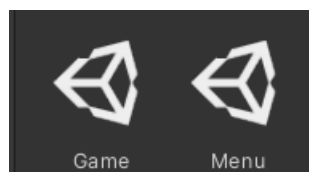


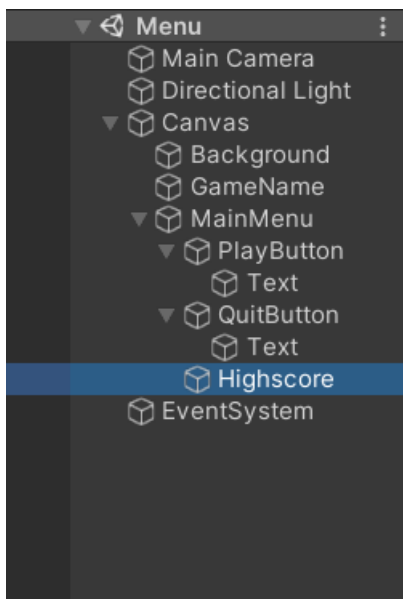
Fig.2 Game scenes

## 1. Menu scene



*Fig. 3 Menu Scene*

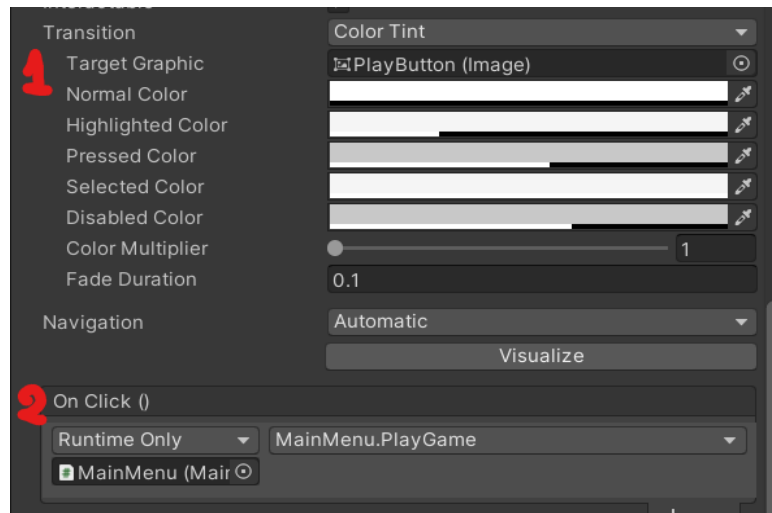
In order to make the menu I used UI (User Interface) to create a canvas where I places the text and buttons. As it shows in the figure below (Fig.4), the Background and the GameName are texts and a game object that forms the MainMenu with the PlayButton, QuitButton and Highscore.



*Fig. 1 Menu hierarchy*



*Fig. 5 Menu display*



*Fig.6 Button design and the action of the Play button*

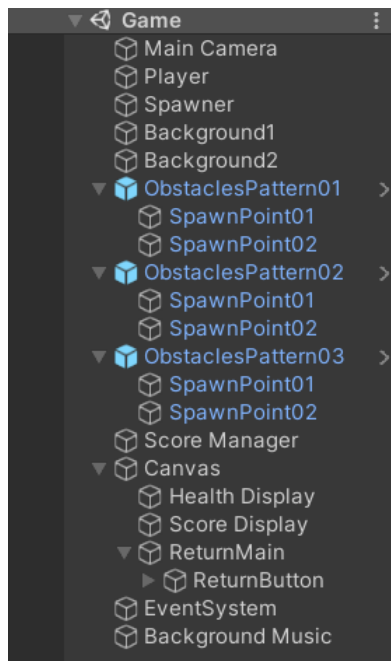
To make a visible action when you move or click the button, I used the characteristics of the button so when you set your cursor above or click the button, the button will turn black(1). “On click” option is used to add action to the button: for the PlayButton the action is to open the Game Scene(2) and for QuitButton is the quit the game.



*Fig.7 Play Button display when selected*

## 2. Game Scene

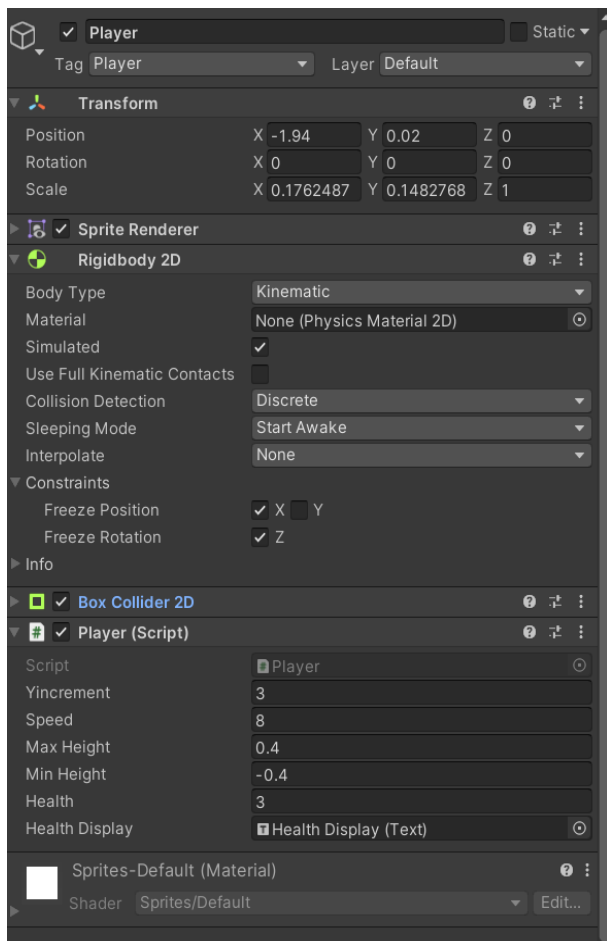
The game scene represents the game space, and it has all the elements mentioned above and more to create the scene such as: main camera, light, player, spawner, backgrounds, obstacles, score manager, canvas with score, life and a return button, and background music.



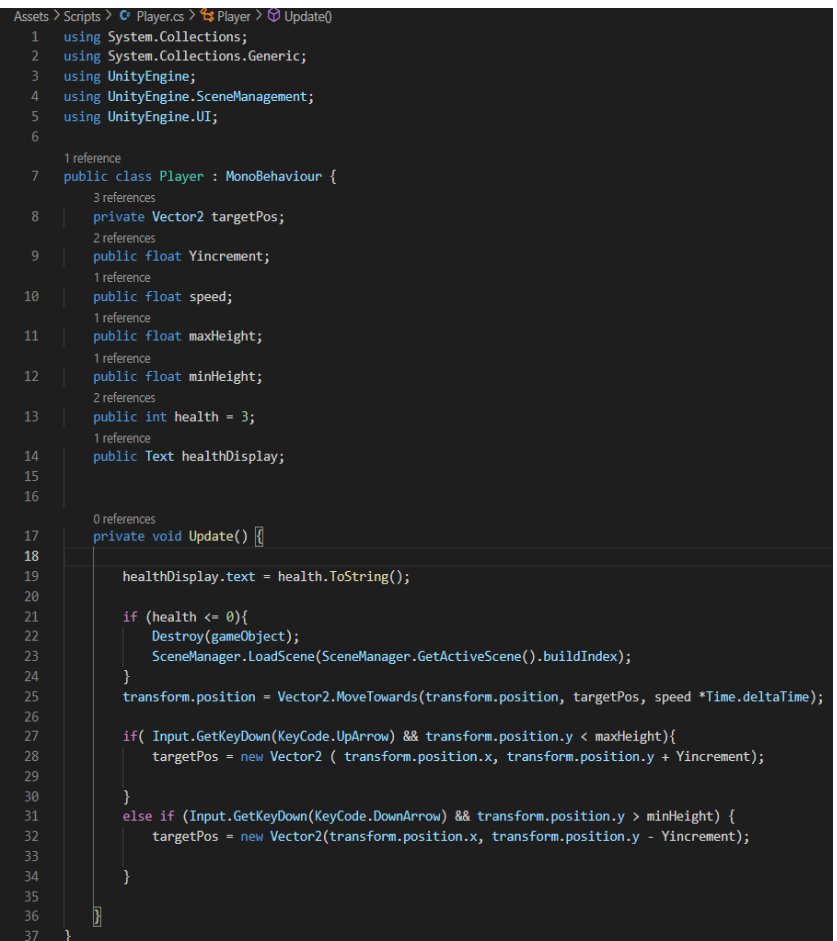
**Fig.8. Game Scene Hierarchy**

The Main Camera is placed on the center of the scene, where the character is placed and visible. The way we adjust the main camera, is the way the games will be seen. For this object I did not modify many things, just the position and set a depth by -1.

The next element is Player. The player needs to have a rigidbody. I chose the kinematic body type because it is important that the player to not be affected by any force and his behavior is controlled on script. I also added a box collider component, so he can collide with other objects on the scene. Furthermore, I constrained the player on the X and Z axis so the character moves just on the Y axis (up and down) and using code I set the speed, health and upper and lower limits; therefore the character will not fly off the scene.



*Fig.9 Characteristics of the player*



*Fig.10 Player's Script*

The Spawner is an important element in the game because I use it as a constant source of obstacles. It is placed outside the background or camera zone and spawns the obstacle patterns randomly. The obstacles patterns are 3 different ways that the obstacles come into the scene.

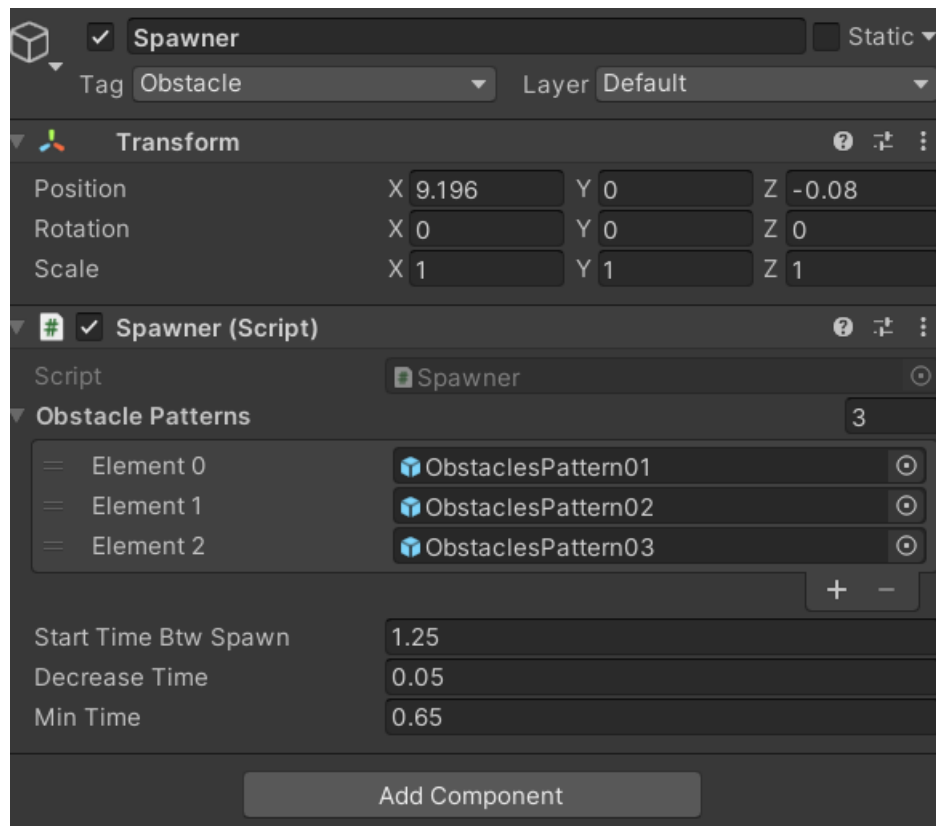
```

Assets > Scripts > Spawner.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Spawner : MonoBehaviour
6  {
7      public GameObject [] obstaclePatterns;
8      private float timeBtwSpawn;
9      public float startTimeBtwSpawn;
10     public float decreaseTime;
11     public float minTime = 0.65f;
12
13     private void Update() {
14         if (timeBtwSpawn <= 0){
15
16             int rand = Random.Range(0, obstaclePatterns.Length);
17             Instantiate(obstaclePatterns[rand], transform.position, Quaternion.identity);
18             timeBtwSpawn = startTimeBtwSpawn;
19             if ( startTimeBtwSpawn > minTime){
20                 startTimeBtwSpawn -= decreaseTime;
21             }
22         }
23         else{
24             timeBtwSpawn -= Time.deltaTime;
25         }
26     }
27 }
28
29

```

*Fig.11 Spawner's Script*

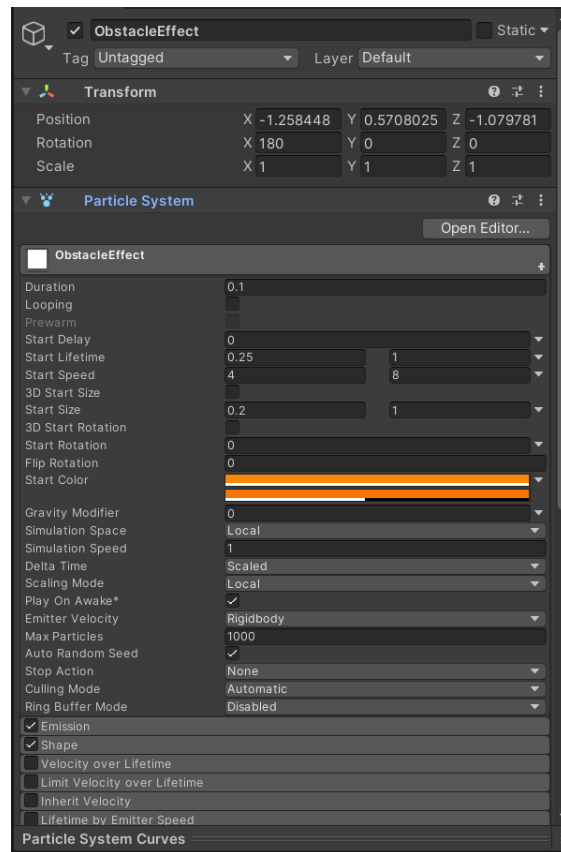
The spawner start time, decrease time and min time refer to the time interval of the spawning.



*Fig.12 Spawner's characteristics*

Another mention about the obstacles is the prefab I created for its effect. When the player collides with the obstacle (the flame) which means losing a life/death of the character, where will

play a particle effect that looks like burning the character and it also play a specific sound effect also like burning.



*Fig. 13 Obstacle Effect characteristics*

The Score Manager is a game object placed on the left side of the character (meaning behind it). The reason why I put it there it is because using the code I made this the place where the obstacles are destroyed and add as score. Score Manager' mechanism is destroying every flame the character managed to pass and count it as a point. At first the score is integer initiated as 0, because of course the score should start from zero point. Next step was to add an instance for the score display. The “ OnTriggerEnter2D(Collider2D other)” function calls the object with a collider( the obstacles) passing the game object that “is trigger”, then add one by one as a score number and save the high score displayed on the Main Menu. The “PlayerPrefs.SetInt(“highscore”, score)” save the score in the unity space and the last “if” condition the high score to be bigger than the score, so it can replace the last saved high score.

```

0 references
7 public class ScoreManager : MonoBehaviour
4 references
8 { private int score = 0;
1 reference
9 public Text scoreDisplay;
0 references
10 private void OnTriggerEnter2D(Collider2D other)
11 {
12
13     if ( other.gameObject.CompareTag("Obstacle"))
14     {
15         Destroy(other.gameObject);
16         //increase score
17         score++;
18         scoreDisplay.text = "Score: " + score;
19     }
20
21     if(score > PlayerPrefs.GetInt("highscore"))
22     {
23         PlayerPrefs.SetInt("highscore", score);
24     }
25 }
26
27

```

*Fig. 14 Score Manager Script*

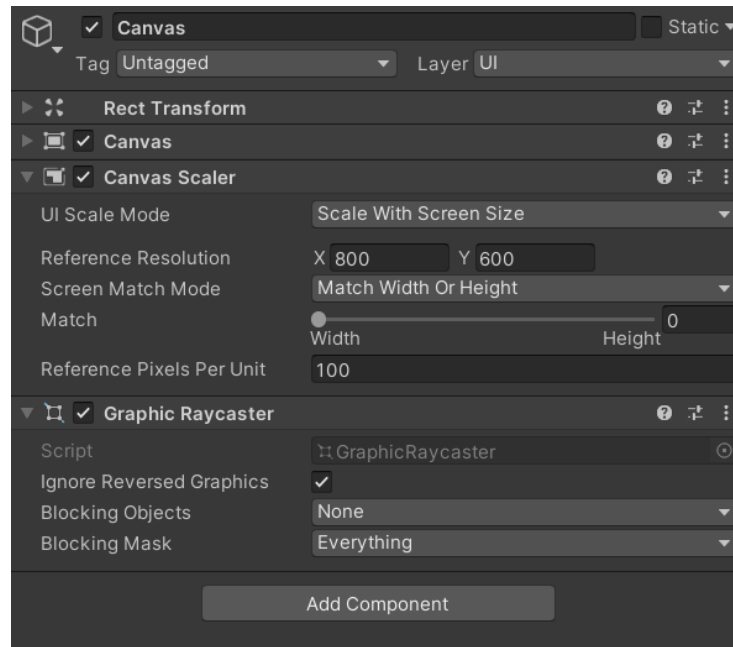
```

Assets > Scripts > MainMenus > MainMenu > Start()
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5 using UnityEngine.UI;
6
0 references
7 public class MainMenu : MonoBehaviour{
1 reference
8 public Text Hstext;
0 references
9 private void Start()
10 {
11     Hstext.text = "HIGHSCORE: " + PlayerPrefs.GetInt("highscore");
12 }
13
0 references
14 public void PlayGame ()
15 {
16     SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
17 }
18
0 references
19 public void QuitGame ()
20 {
21     Application.Quit();
22 }
23
24 }

```

*Fig. 15 Main Menu Script*

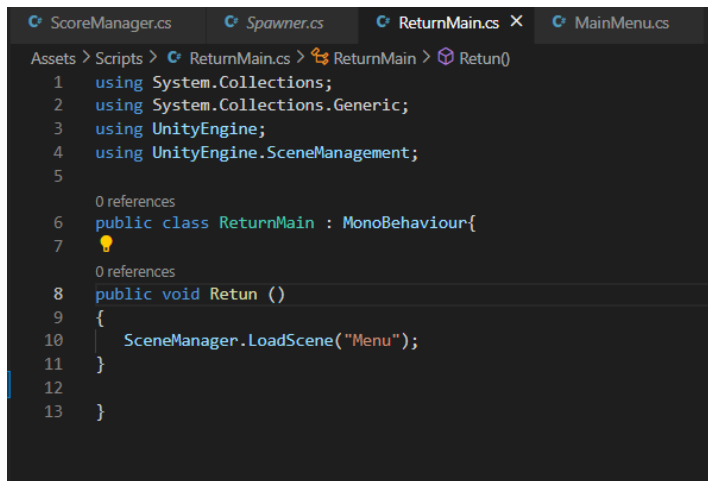
In the UI “Canvas” you can find the text display for life and score, and a button that returns you to the Main Menu. I set the canvas to “Scale With Screen Size” for making the game adaptable to any screen resolution without losing any elements.



*Fig. 16 Main Menu characteristics*

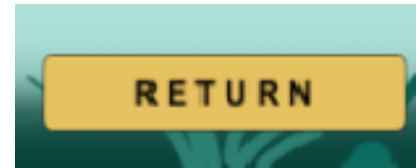


The return button is found on the down side of the screen. By adding a script to it, I created an action which can get you to the Main menu if you click it.

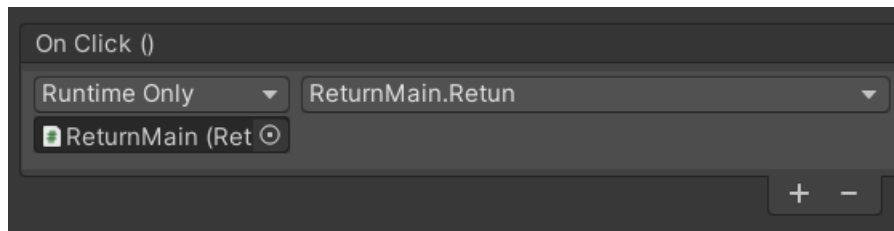


```
Assets > Scripts > ReturnMain.cs > ReturnMain > Return()
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class ReturnMain : MonoBehaviour{
7     0 references
8     public void Retun ()
9     {
10         SceneManager.LoadScene("Menu");
11     }
12 }
13 }
```

*Fig. 17 Return Button's Script*

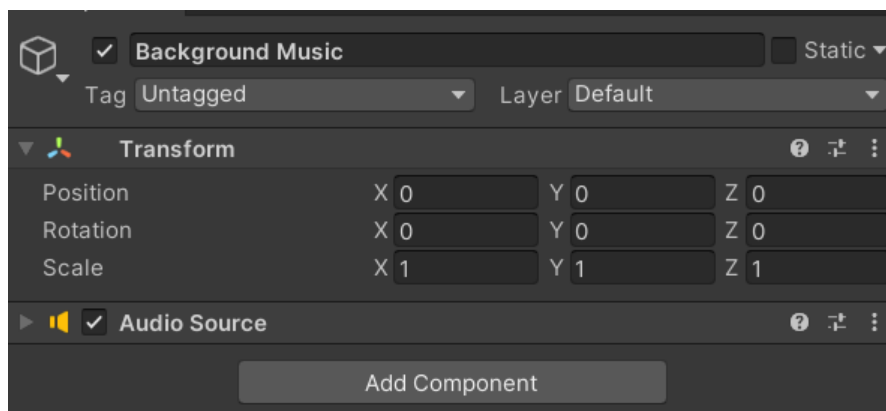


*Fig. 18 Return Button on Game Scene*

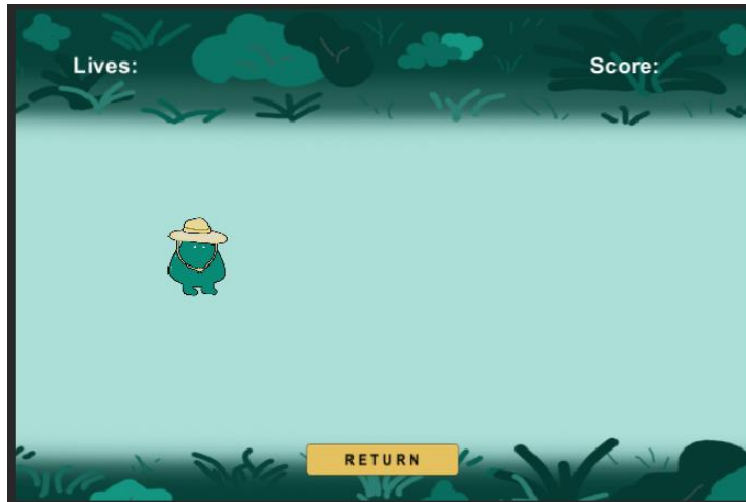


*Fig. 19 Action on the Return Button*

The last element is the background music. The music start to play when you enter the Game Scene.



*Fig. 20 Game object Background music characteristics*



*Fig. 21 Game Scene- final result*

## 6. Future work

What can be improved:

- Add a sound to main menu and stop it when enter the game scene
- Give the possibility to change difficulty level
- Create an “option” button on the main menu
- Add object to collect (the flowers)

## 7. Conclusions

Making this game has been fun and challenging since I've never done so before. It was interesting to do searches and adjust the code written methods for my game. From the result I don't think it's 100% the game, but the fact that I learned to work in Unity and make my first game almost from scratch.

## 8. References:

- [https://www.youtube.com/watch?v=zc8ac\\_qUXQY&t=59s](https://www.youtube.com/watch?v=zc8ac_qUXQY&t=59s)
- [https://www.youtube.com/watch?v=5M7vX\\_z6B9I&t=348s](https://www.youtube.com/watch?v=5M7vX_z6B9I&t=348s)
- <https://www.youtube.com/watch?v=yE0JdtVTnVk>
- <https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html>