## Task 1 – (P1.1)

A Game Engine is an application that is designed to create and develop video games. The current most popular game engines are Unity, Unreal Engine 4 and CryENGINE. These are 3 powerful game engines and each are strong in a certain area of game development. It is important to choose the right game engine when creating a game to get the most out of what you are creating.

**Cost:** The first thing that comes to many people's mind at first is the cost to purchase the game engine. A lot of game engines can be purchased for free but have limited features. Most engines offer a version that includes more features but has to be bought or can be paid monthly at a cheaper price.

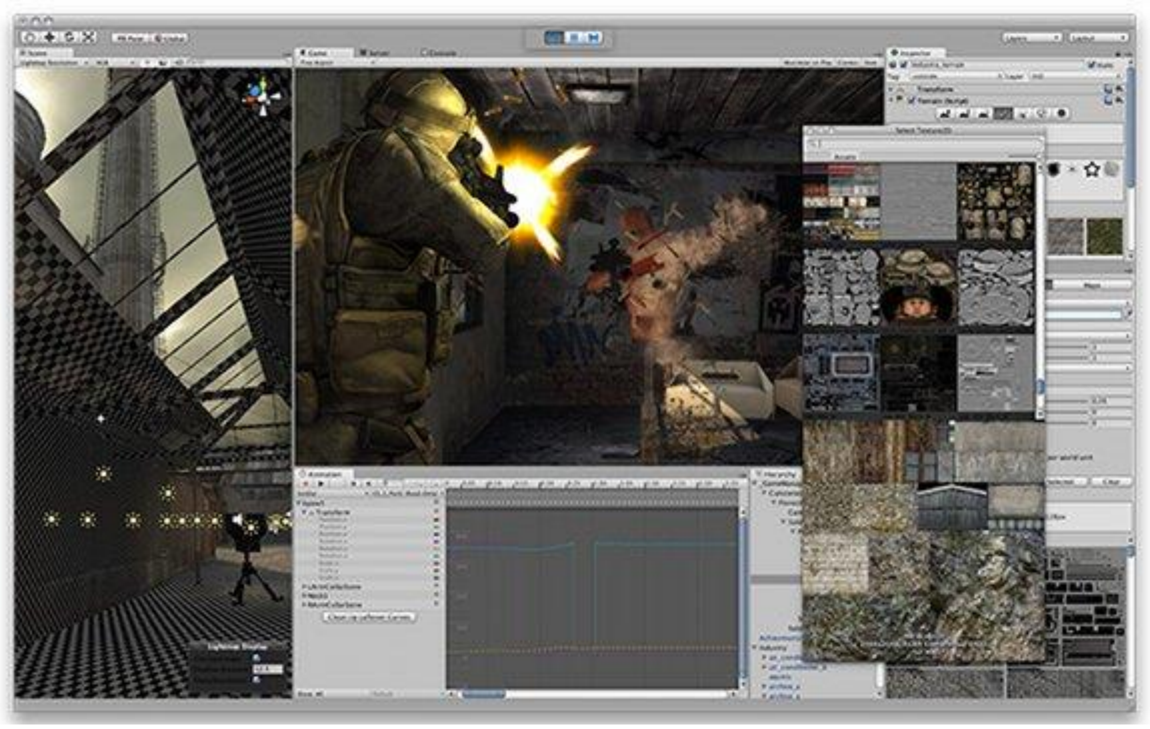
**2D/3D:** Choosing a game engine also depends on the type of game you want to develop.  Although most game engines offer both 2D and 3D options, they usually are more specific to one of the types. So before choosing a game engine it is important to know what game you are going to make.

**Licensing Fees:** If you plan on releasing a game or selling it for an amount of money, you have to take into consideration different licensing fees each game engine offers. Even though these game engines are relatively cheap, there are licensing fees or royalties you must pay.

**Platform:** It is also important to take into consideration on which platform you are going to release the game. Most game engines are cross-platform and allow you to build a game for different platforms such as different mobiles, web and gaming consoles such as Xbox and PlayStation.

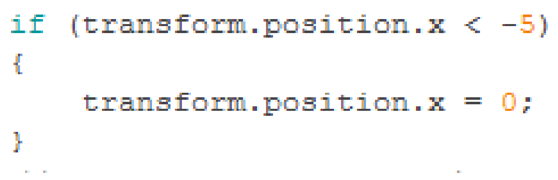
**Editing Capabilities:**  Most game engines allow you to edit a number of features in the game editor such as modeling and building features. On the other hand some game engines are more adapt to editing 2D objects rather than 3D objects, so it is important to know what type of game you are creating beforehand. Some game engines also allow you to download assets with varied prices.

Personally, I would choose Unity as it has a lot to offer. The normal version can be downloaded for free and it is a cross-platform game engine. It also allows you to create both 2D and 3D games that can be built for a large number of platforms. This game engine also supports assets from major 3D applications so there is no real restriction in the formats you can import. On the other hand it does suffer in the amount of editing capabilities it has to offer. However it does have a large asset library where assets can be downloaded or purchased. Apart from its editing capabilities, Unity has everything you need to create any type of game for nearly every type of platform.



**<u>Unity Screenshot</u>**

## Task 2 – (P1.2)

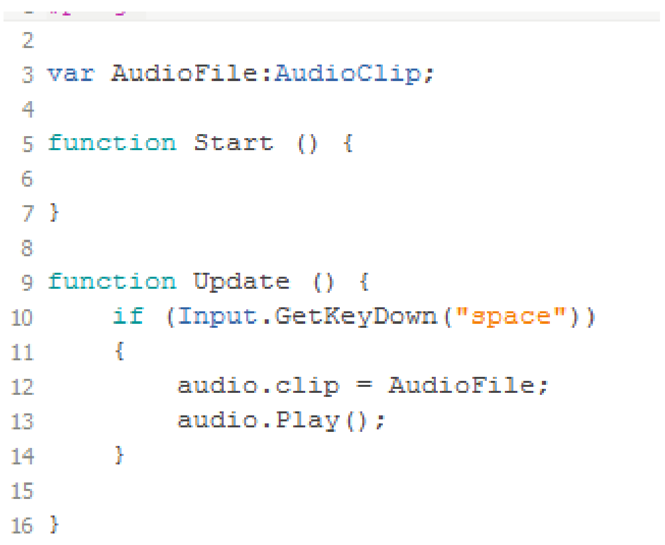


```
if (transform.position.x < -5)
{
    transform.position.x = 0;
}
```

This is the code I used on the Player, so it could not exit the left-hand side of the screen. “transform.position.x” shows the position of the transform along the x-axis. This code is simply saying that if the player tries to go past -5 along the x-axis, the player is positioned back to 0 thus preventing him from escaping the game screen.

## Task 3 – (P2.1)

I started off by creating a prefab where I added an Audio Source (component) and dragged the sound effect into the Audio Clip tab. I then had to attach the following code to the prefab which is basically storing the sound effect in the variable. In the Update function, I am telling the game engine to play the sound effect when ‘space’ is pressed. For the sound effect to work during the game, I had to drag the prefab onto the player (parenting it).



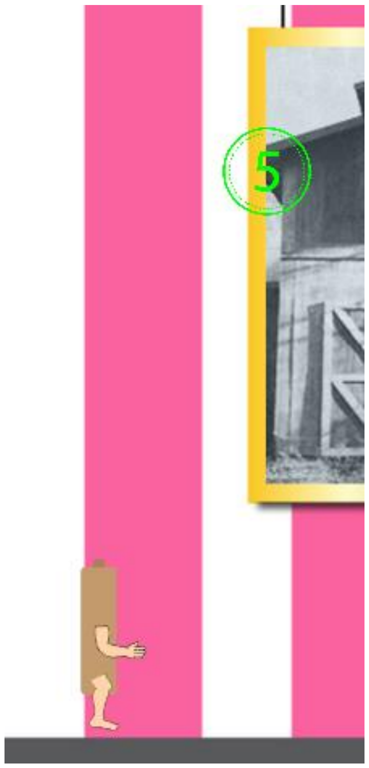
```

var AudioFile:AudioClip;

function Start () {

}

function Update () {
    if (Input.GetKeyDown("space"))
    {
        audio.clip = AudioFile;
        audio.Play();
    }

}
```

## Task 4 – (P2.2)

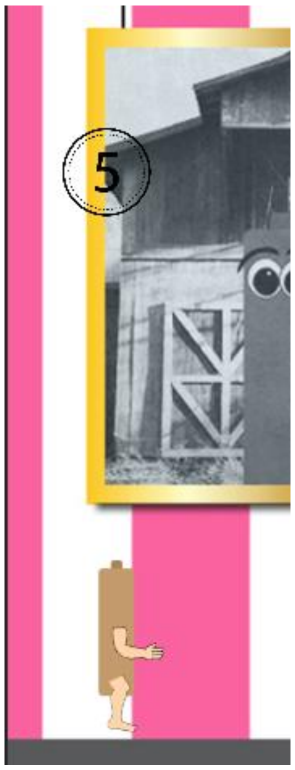
I recorded a noise which could be used when the Player fires his weapon/lightning bolt. I added some effects to the sound such as noise reduction and fade out as the noise became smoother. I also reversed the sound and added a vocoder effect to get my desired noise. The sound effect was recorded and edited in Audacity.

The original and edited sound effects can be found here (Luke_Camilleri_2HND1 > Documentation > Sound).

## Task 5 – (P3.1)

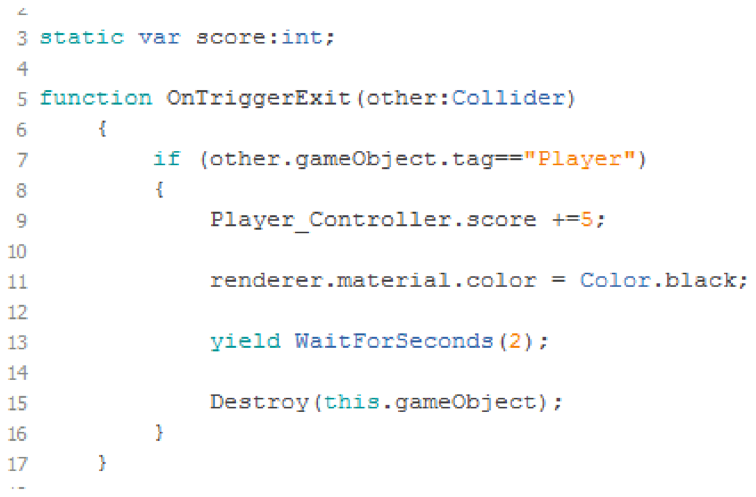


**<u>Bonus Icon is Green</u>**



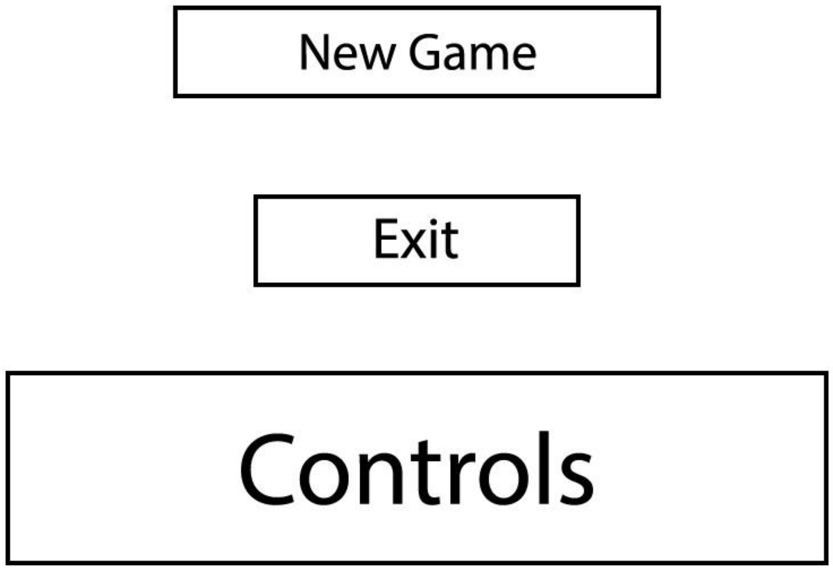
**<u>Bonus Icon changes to Black after the Player passes through it</u>**

In order to this I used the OnTriggerExit although it could have been done using the OnTriggerEnter too. Line 5 and 7 show when the icon's collider passes through the Player. Line 9 adds to the Player's score. Line 11 changes the icon's material color from green to black. Line 13 and 15 mean that after 2 seconds after the icon changes color it is destroyed.

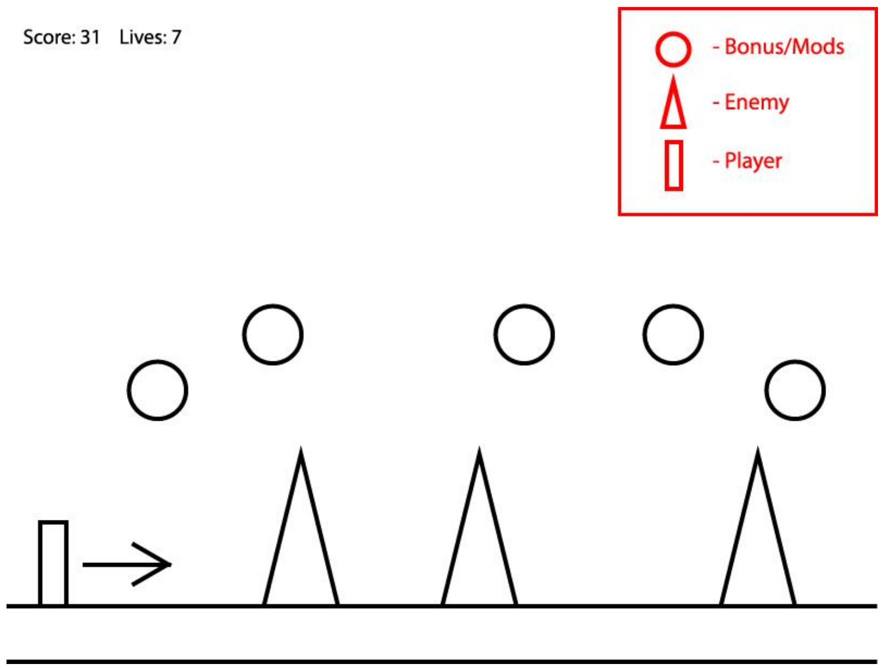


```
 3 static var score:int;
 4
 5 function OnTriggerExit(other:Collider)
 6     {
 7         if (other.gameObject.tag=="Player")
 8         {
 9             Player_Controller.score +=5;
10
11             renderer.material.color = Color.black;
12
13             yield WaitForSeconds(2);
14
15             Destroy(this.gameObject);
16         }
17     }
```

## Task 6 – (P3.2)



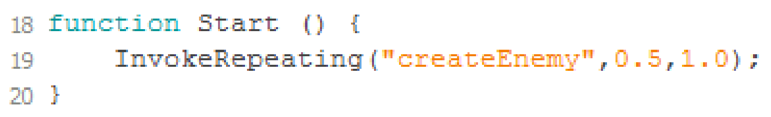
**Main Menu Screen**



**Game Screen**

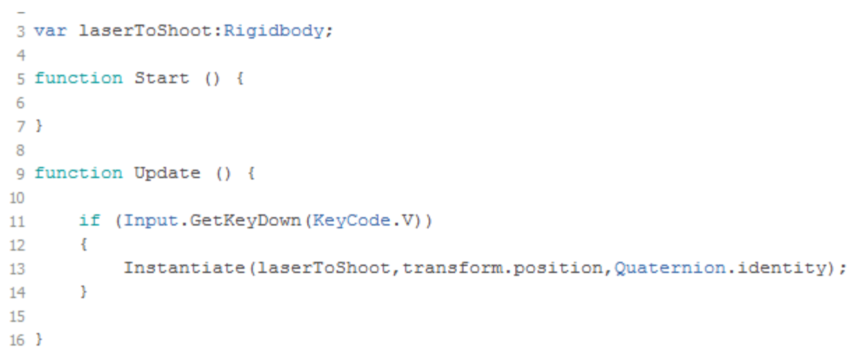
## Task 7 – (P4.1)

**Start** is called when a script is enabled before any Update methods are called for the first time. I used the Start method to generate enemies in the game. I did this by calling a previous method using the InvokeRepeating. This was done in the start method so the enemies are generated from the beginning of the game.



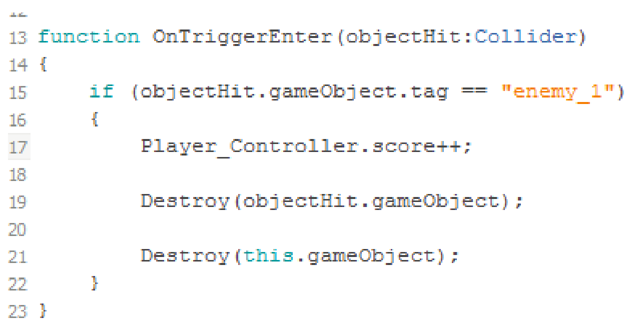
```
function Start () {
    InvokeRepeating("createEnemy",0.5,1.0);
}
```

**Update** is the most commonly used function to implement any kind of game behaviour. I used the Update method for my laser prefabs which are shot when I press a certain key. The lasers are cloned and their movement controlled from the Instantiate function. This has to be done in the Update method as the player controls this type of behaviour.



```
var laserToShoot:Rigidbody;

function Start () {

}

function Update () {

    if (Input.GetKeyDown(KeyCode.V))
    {
        Instantiate(laserToShoot,transform.position,Quaternion.identity);
    }

}
```

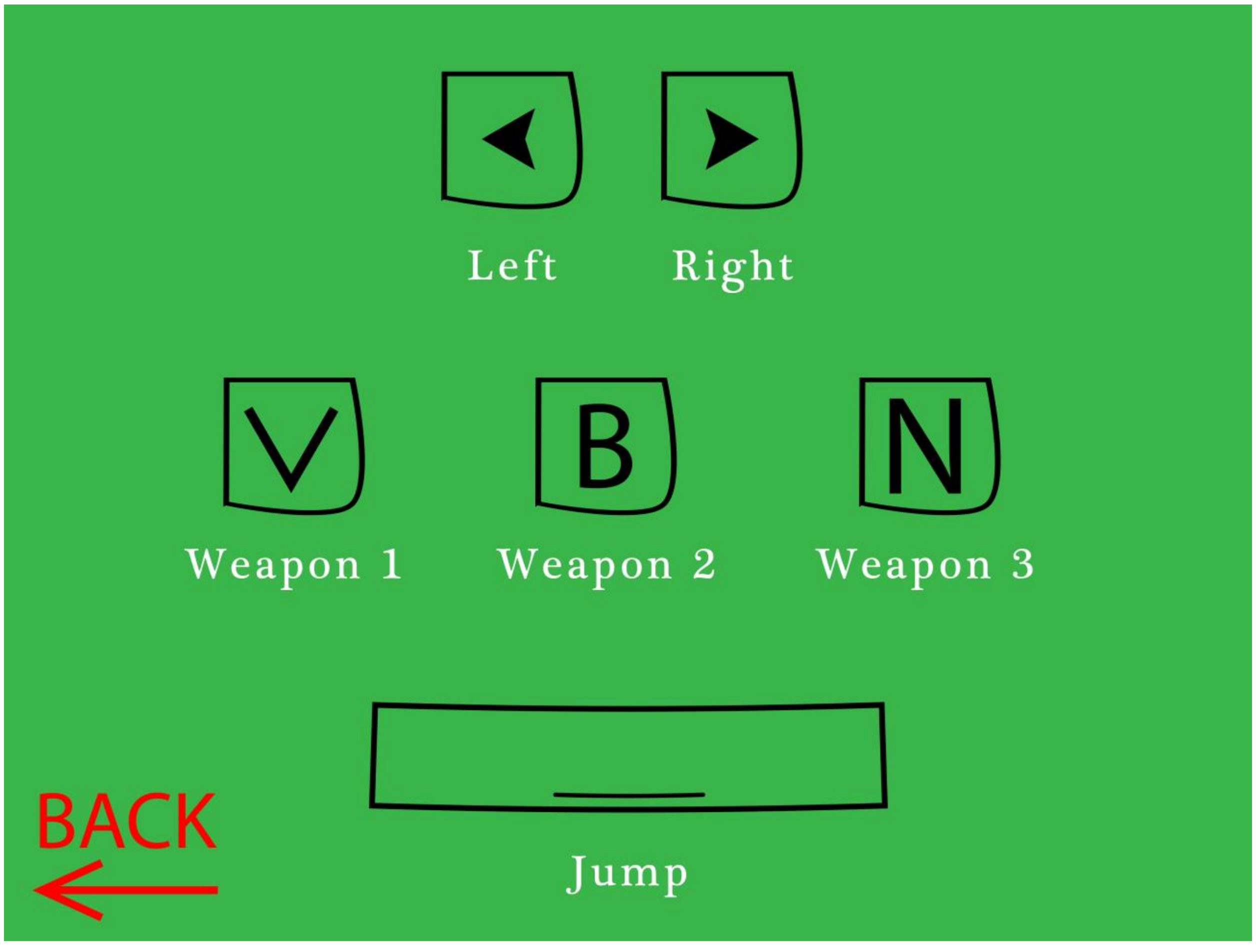
**OnTriggerEnter** is called when the Collider enters the Trigger. This function was used in my game when my laser hits an enemy. When the laser's collider triggers the enemy's collider, my score goes up and the enemy is destroyed. This is always done using the OnTriggerEnter function.



```
function OnTriggerEnter(objectHit:Collider)
{
    if (objectHit.gameObject.tag == "enemy_1")
    {
        Player_Controller.score++;

        Destroy(objectHit.gameObject);

        Destroy(this.gameObject);
    }
}
```

## Task 8 – (P4.2)

In my game you are a battery who got lost and has to face various enemies who shoot water (his worst nightmare). The game is a sidescroller so the battery has to move to the right although it can move back to the left. The battery can also jump and shoots 3 different types of electric shocks. Enemies can reduce the battery's lives by either touching the battery or by shooting water. In each level the battery starts off with no points and 10 lives. During the game the battery can collect various power ups which include extra lives and points. The aim is to reach 100 points without dying to reach the next level. If the battery dies, the game starts again from the main menu. The game's instructions can be found below. Hint: The longer the space bar is pressed, the higher the battery jumps.



## Task 9 – (P4.3)

As a designer I am never really happy with what I do and always come up with improvements after I have created something. The first thing I would like to have improved in my game was to reduce the amount of duplicate code and scripts as it would be an easier job to troubleshoot my game. On the visual side of the game I would like to have had more time to design the backgrounds and icons and continue on one style instead of having multiple colours and styles. As for gameplay I would have liked the player to earn new weapons instead of having 3 weapons that work with different keys. Another improvement would also have been adding platforms in different places so the player has more obstacles to pass through. My final improvement would be adding more sound effects and of better quality.