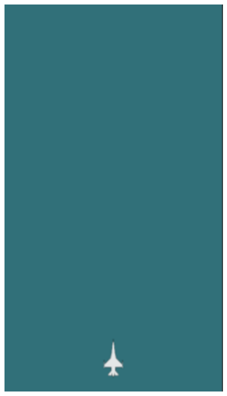
Game Engine’s Documentation

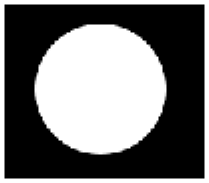
Task 3

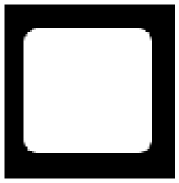
A) Graphical Assets – The **plane** was created in Photoshop using the pen tool and at times the curved pen tool. When I finished the outline of the plane I filled it in white.

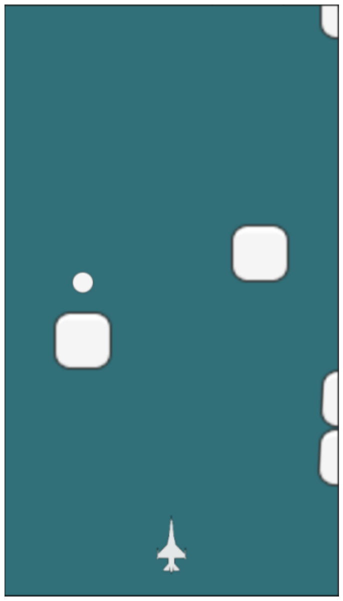


In game the **plane** would be staying in a stationary position with no ways on moving it and the goal is to protect it from any incoming collision that may too into contact with the plane.

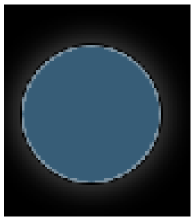
Collisions - The **collisions** in the game were used with the basic shape tool in Photoshop, a **circle** and a **rounded square**. These 2 shapes came in 2 different sizes, one small for each shape and a larger one of each.

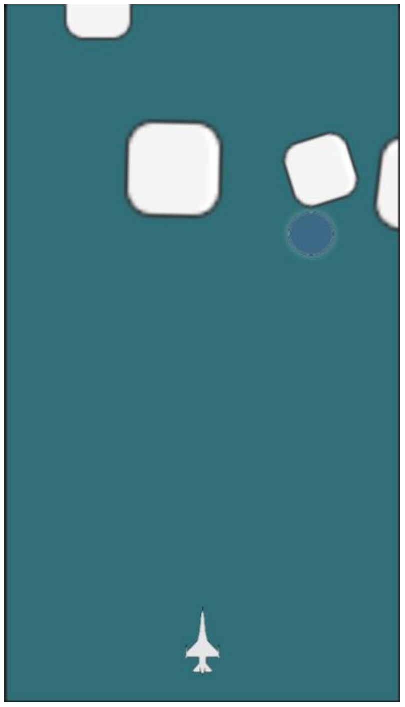




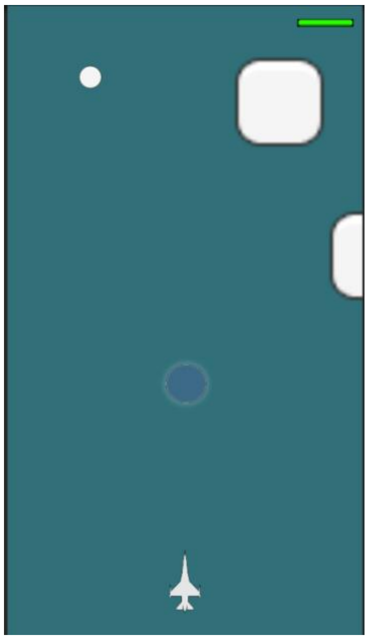


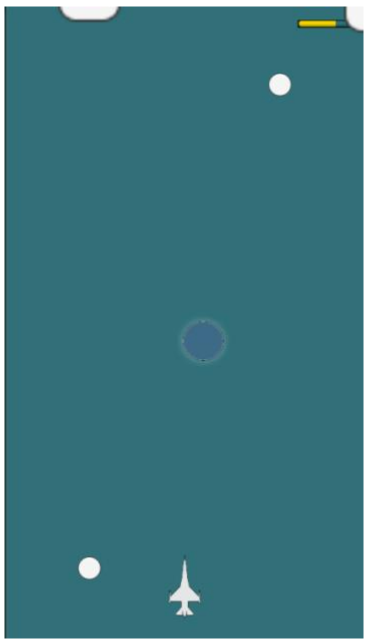
Cursor - As the game didn't provide any functionality with the arrow keys, the **cursor** is displayed as a blue circle with a hint off outer glow to make it stick out more whilst controlling it with the mouse. The cursor is used to knock away any incoming collision coming within the plane.

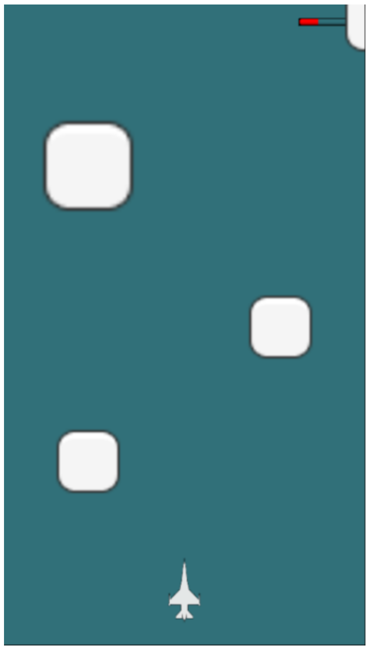




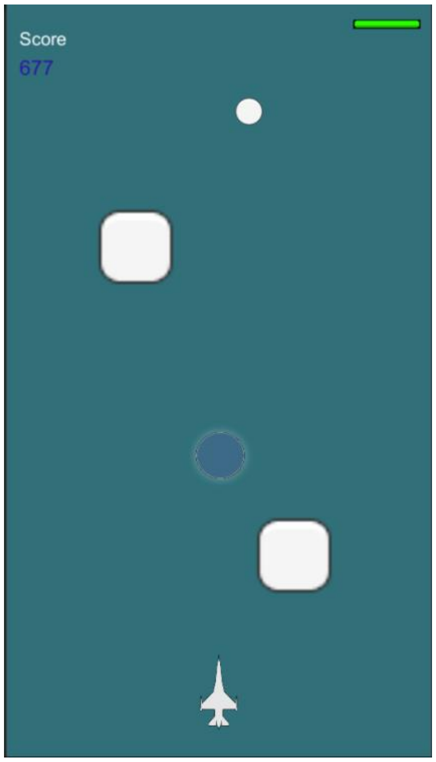
Health Bar - Another important art asset was the **health bar**; it was created in Photoshop using the rounded rectangle tool and adjusted accordingly. 3 vertical lines went across the bar, each space representing a life which counts to 3 lives in total. Green meaning 3 lives', yellow 2 and red being your last life



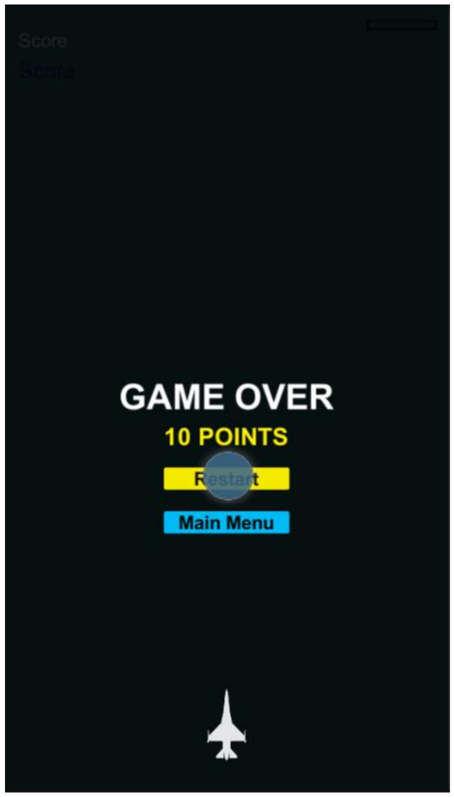




Score System - When the game is being played on the top left corner there will be a score in blue going up, this will increase in number the longer the game lasts.

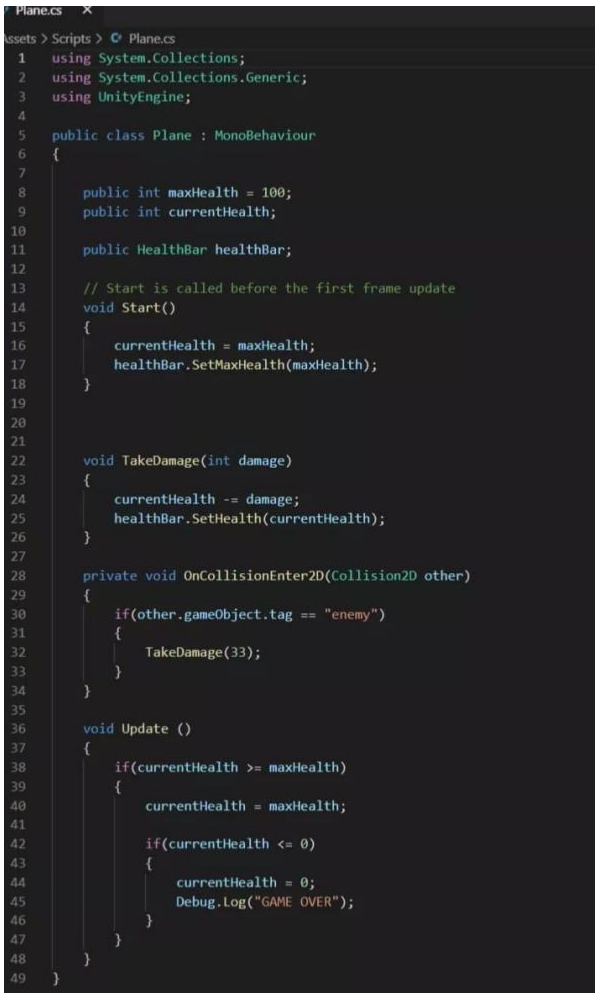


The **game over screen** was designed in unity it's self with the help of the tools that were provided. Unfortunately, the screen didn't appear when the plane reached 0 due to an error in the code.



B) Scripts Overview – Plane

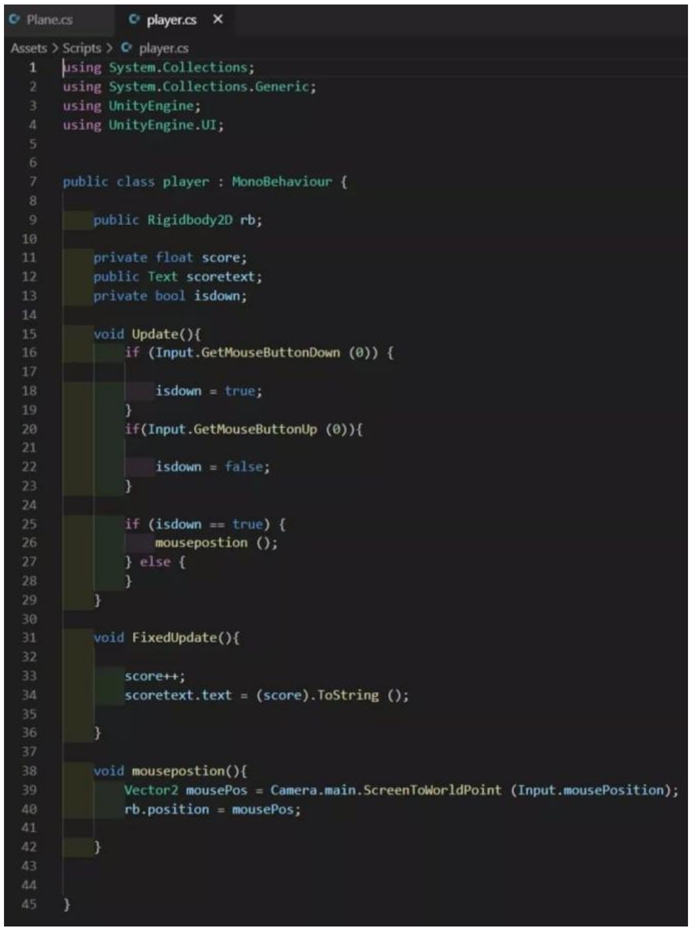
The plane script’s purpose was to assign the plane object which would be at a stationary position with 100hp. Each time the plan was getting hit by any incoming collision, it would take 33 damage. In total would add up to the plane’s full hp.



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

public class Plane : MonoBehaviour
{

    public int maxHealth = 100;
    public int currentHealth;

    public HealthBar healthBar;

    // Start is called before the first frame update
    void Start()
    {
        currentHealth = maxHealth;
        healthBar.SetMaxHealth(maxHealth);
    }



    void TakeDamage(int damage)
    {
        currentHealth -= damage;
        healthBar.SetHealth(currentHealth);
    }

    private void OnCollisionEnter2D(Collision2D other)
    {
        if(other.gameObject.tag == "enemy")
        {
            TakeDamage(33);
        }
    }

    void Update ()
    {
        if(currentHealth >= maxHealth)
        {
            currentHealth = maxHealth;

            if(currentHealth <= 0)
            {
                currentHealth = 0;
                Debug.Log("GAME OVER");
            }
        }
    }
}
```

Player Script – The player in our case of the game would be controlling the cursor hence the name of the script. The script’s function assigns the cursor art asset with wherever u wish to move the mouse in game.

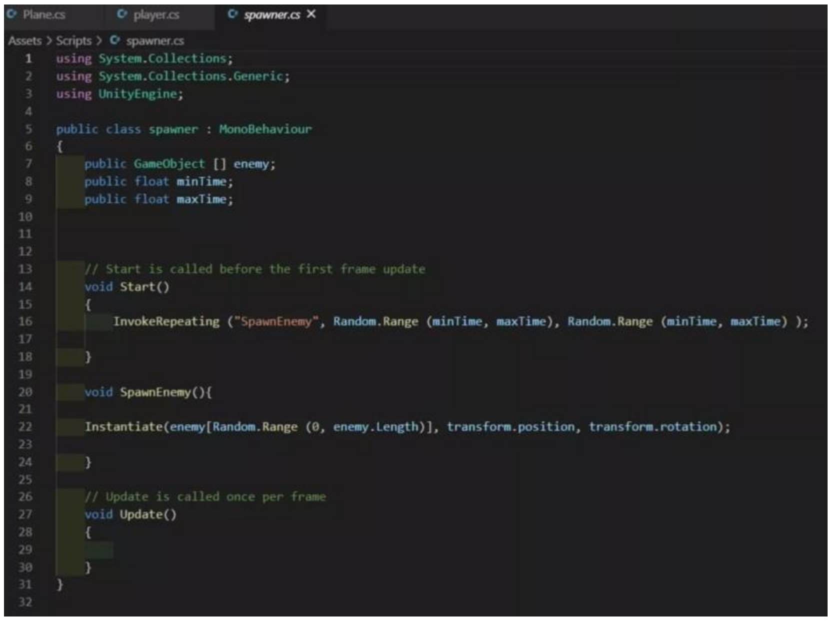
The other part of the script is for the score system to function which depends on how long the game goes for, this depends on the planes lives.



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


public class player : MonoBehaviour {

    public Rigidbody2D rb;

    private float score;
    public Text scoretext;
    private bool isdown;

    void Update(){
        if (Input.GetMouseButtonDown (0)) {

            isdown = true;
        }
        if(Input.GetMouseButtonUp (0)){

            isdown = false;
        }

        if (isdown == true) {
            mousepostion ();
        } else {
        }
    }

    void FixedUpdate(){

        score++;
        scoretext.text = (score).ToString ();

    }

    void mousepostion(){
        Vector2 mousePos = Camera.main.ScreenToWorldPoint (Input.mousePosition);
        rb.position = mousePos;

    }


}
```

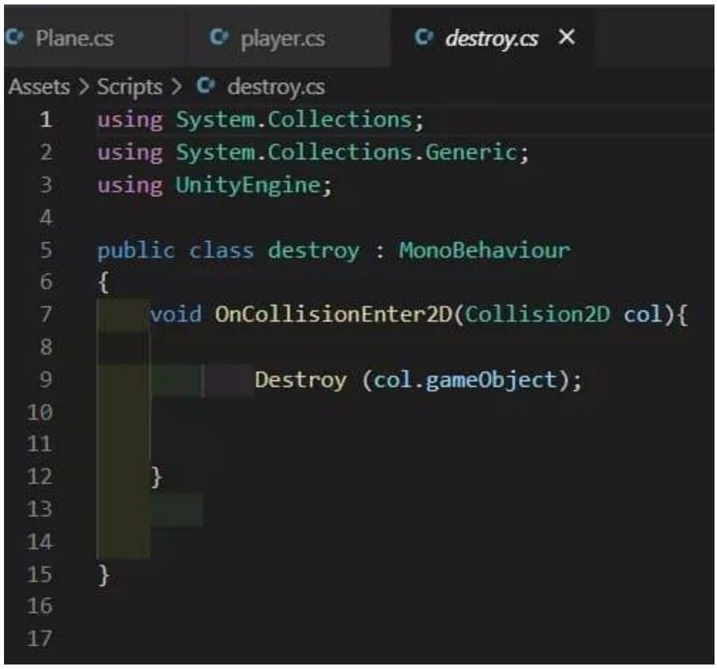
Spawner Script – Without the spawner script there wouldn’t be any collisions, the script allowed for the art assets collisions to spawn wherever the spawners were located.

This allowed for random positions, rotations and sizes of collisions to be spawned in from the script.



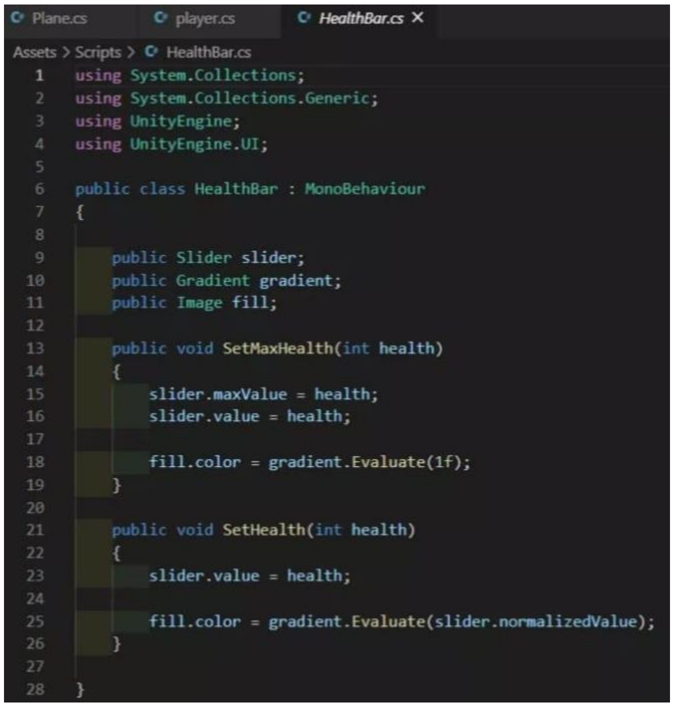
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class spawner : MonoBehaviour
{
    public GameObject [] enemy;
    public float minTime;
    public float maxTime;



    // Start is called before the first frame update
    void Start()
    {
        InvokeRepeating ("SpawnEnemy", Random.Range (minTime, maxTime), Random.Range (minTime, maxTime) );

    }

    void SpawnEnemy(){

    Instantiate(enemy[Random.Range (0, enemy.Length)], transform.position, transform.rotation);

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Destroyer Script – The script’s purpose was for any of the collisions falling out of the viewer’s screen would be completely deleted. This basically assigned a death barrier for any collisions coming intact with the barrier. With the help of this, the collisions weren’t stacking in the hierarchy which made the game run without any spikes.



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

public class destroy : MonoBehaviour
{
    void OnCollisionEnter2D(Collision2D col){

            Destroy (col.gameObject);


    }


}
```

Health Bar Script – Since the healthbar was made in Unity, it was easier to assign the scipt directly with the tools. The script was handling the slider, gradient and fill which deppended on how many lives were remianing. Each time the plane was getting hit by a collision, the healthbar script would take notice and scale down the slider according to how many times the plane was hit.



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

public class HealthBar : MonoBehaviour
{

    public Slider slider;
    public Gradient gradient;
    public Image fill;

    public void SetMaxHealth(int health)
    {
        slider.maxValue = health;
        slider.value = health;

        fill.color = gradient.Evaluate(1f);
    }

    public void SetHealth(int health)
    {
        slider.value = health;

        fill.color = gradient.Evaluate(slider.normalizedValue);
    }

}
```

C) Process -

I started the scene by creating a greenish blue background for the canvas to try to display the sky background which will always be the background for the scene. After I finished that, I grabbed the Plane art asset and made sure to align it to the middle of the screen but positioned it to the bottom so the player would have time to react. I added an edge collider 2D to give the plane a hit box that I can work with. I assigned a script to the plane by giving it 3 lives by assigning it with 100hp.

I dragged the cursor asset onto the hierarchy and gave it a component containing a rigidbody2D. I changed the gravity scale to 0 and changed the collision detection to continuous for it to constantly push objects away.

I added a circle collider 2D and adjusted it around the cursor accordingly to match with the size. After that was done, I added a script to the player to give functionality depending on the where the mouse was moved.

Afterwards I added some sprites which were the 2Dcollisions; these were the rounded square shape and the normal circle shape. Same thing I did with giving a hit box to the cursor and plane I had to the same with the collisions. Instead, I added a simple box collider 2D to the boxes and added a circle collider 2D to the circles which gave my collisions a barrier.

At this stage the collisions still didn't have any way to appear on the screen. To fix this, I created a spawner script to assign the 2D collision sprites to the script. This allowed me to change the spawns, angles, sizes and positions of the collisions falling.

As I had collisions falling outside the screen they were piling up and filling my hierarchy with collisions. To fix this I added an empty component with a Box Collider 2D and dragged it outside the screen, the goal was to make a working death barrier. I added a destroyer script to this for any collisions going out of zone to be erased.

For player acknowledgment I went to add on a score system by adding text to the canvas and scaled it to the screen size. One displaying the name score and the other would be taking note the actual score. On the player script I made a private float and void update for the score to increase depending on how long the plane lived for.

When it came to the health bar, I made it inside Photoshop and added it as a sprite; I added a fill for the health bar by using images. To make better, I added a slider component to the health bar so I could adjust it whilst the game isn't working. For it to function I added a health bar script to make the slider function whenever the collisions hit the plane. For the final touch, I added a gradient script to the health bar. I made this were when the plan was at 100 the bar would stay green, 66 getting to yellow and to its final life it would be at red.

D) Testing

Functionality	Expected Output	Actual Output	Status
Moving the mouse moves the cursor at any difrection that the user inputs.	Cursor moves at the direction of the input	Cursor moves at the direction of the input	🟩
Hitting the incoming collision with the cursor re-directs the object the opposite direction	Collision gets pushed the direction it was hit	Collision gets pushed the direction it was hit	🟩
Collisions fall at random angels once the game starts	Collisions fall at randoms angels	Collisions fall at a straight angel	🟧
Once the game starts, score starts to increase	Points increase upon on the game starting	Points increase upon on the game starting	🟩
If a collision hits the plane, the health bar will decrease by 33%	HealthBar is decreased by 33% when hit by an incomming collision	HealthBar is decreased by 33% when hit by an incomming collision	🟩
The healthbar changes color upon on the plane getting hit by an incoming collision	Healthbar changes color upon on the plane getting hit by a collision	Healthbar changes color upon on the plane getting hit by a collision	🟩
Once the plane reaches 0HP the game ends	Game ends once the planes reaches 0HP	Game keeps playing without ending when the plane reaches 0HP	🟥
Game Over screen is shown when the plane reaches 0HP with the total amount of score aquired during the play time	Game Over screen is shown upon on the plane reaching 0 HP displaying the amount of score aquired	Game Over Screen doesn't get shown and the game keep's playing when the plane reaches 0Hp	🟥
The collision that's coming into contact wih the plane vanishes	The collision that hits the plane vanishes	The collision that hits the plane, rolls/bounces off the plane and leaves the game view	🟧