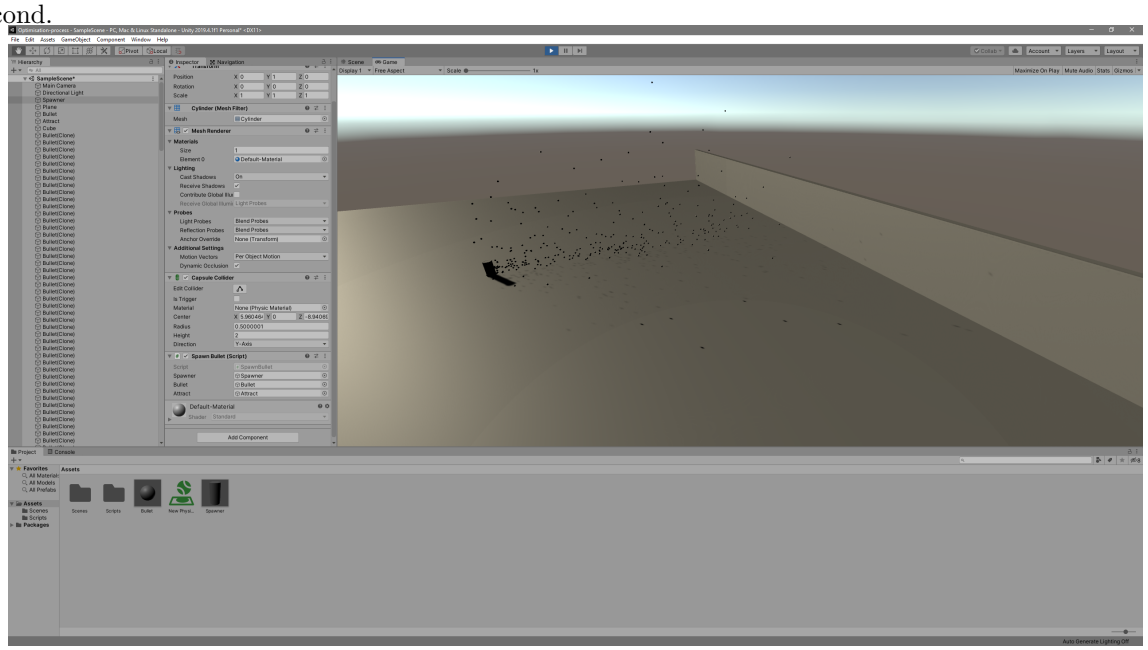
COMP280 Worksheet 3

Philip Hutchings / PH230156

January 3, 2021

1 Introduction

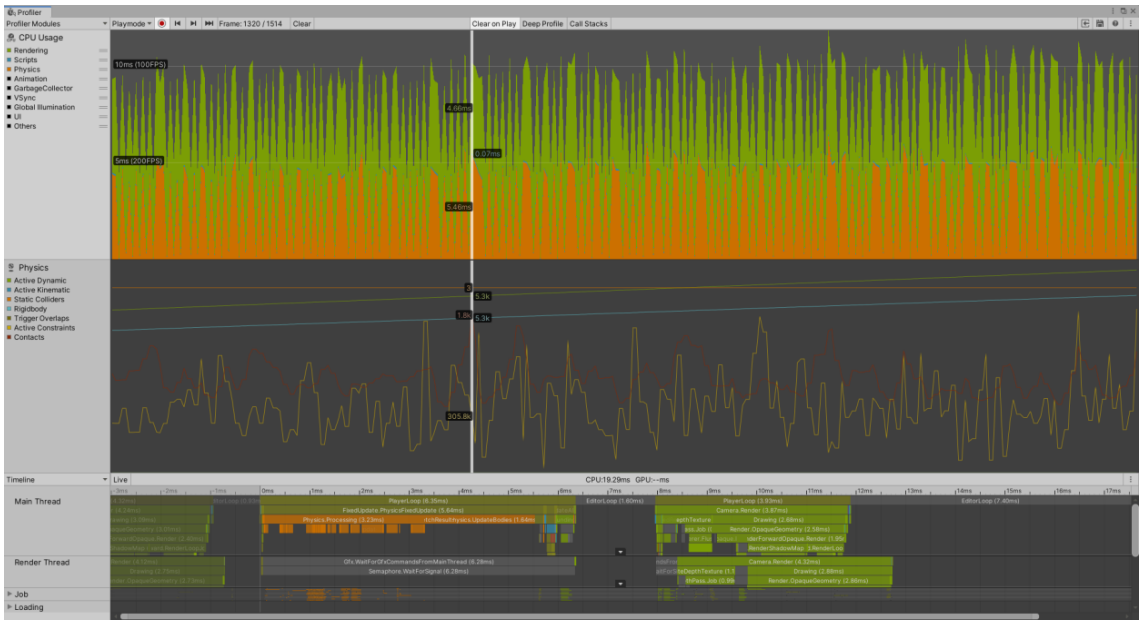
For this assignment, I will be using a basic unity project I made which can be found using the link: (link). It is a basic project where a cylinder game object will spawn several spheres which have a custom physics material and modified rigidbody attached to a sphere prefab. I think that it is relatively unoptimised, although it may be above a standard frames per second e.g. 60 frames per second.



A screenshot of the spheres firing out of the cylinder, though it might be hard to see.

The first thing we are doing is to look at the profiler to get an idea of how things are performing.

2 Profiler analysis before optimisation



In this screenshot, you can see the profiler window for my project. I have also turned off many of the unnecessary views such as garbage collector, vsync, etc. I have left on the rendering, scripts and physics views since these are the main contributors to my profiler statistics.

The rendering is responsible for displaying the game to the screen, drawing objects, etc. There could be slow performance caused by objects which are being drawn when they dont need to be drawn, badly optimised code, etc.

The scripts view is responsible for showing how many resources are being used to run the scripts. The low performance could be caused by bad code or calling too many functions in one update at once.

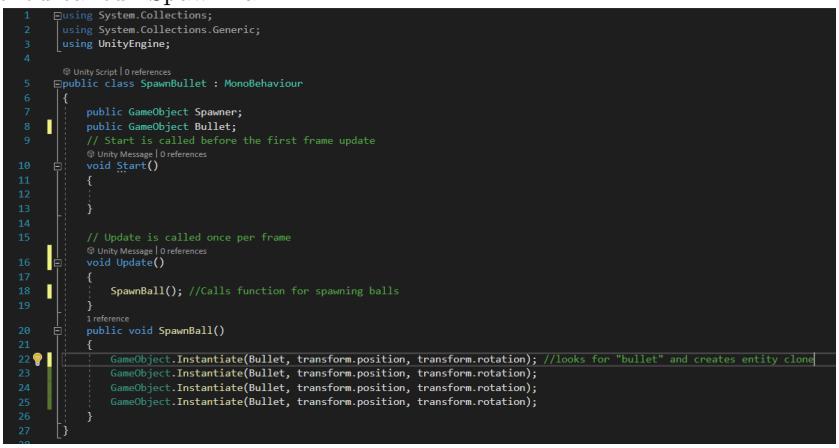
Finally, the physics view is responsible for showing us how many constraints there are, collisions between objects, etc. The low performance from the physics could come from the sheer amount of objects with rigidbodys and physics materials, all interacting with each other at once.

So, first things first, we must identify the biggest contributor and optimise it in a way, in this case its physics processes.

3 First optimisation process

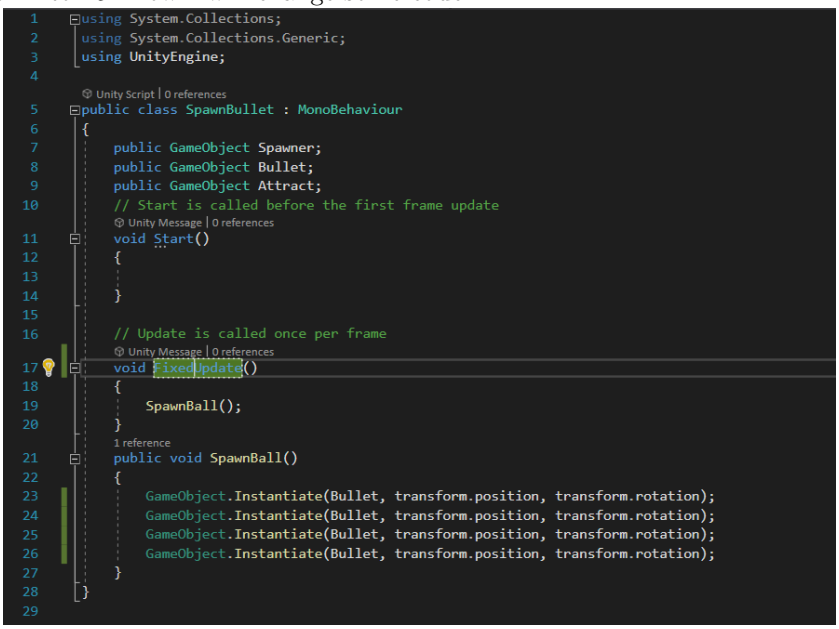
Looking at the timeline at the bottom, we can see that the main thing happening each update is a physics process.

If we take a look at my code, specifically the update function, its calling a method called "SpawnBall".



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
   Unity Script | 0 references
5  public class SpawnBullet : MonoBehaviour
6  {
7      public GameObject Spawner;
8      public GameObject Bullet;
9      // Start is called before the first frame update
       Unity Message | 0 references
10     void Start()
11     {
12
13     }
14
15     // Update is called once per frame
       Unity Message | 0 references
16     void Update()
17     {
18         SpawnBall(); //Calls function for spawning balls
19     }
       1 reference
20     public void SpawnBall()
21     {
22         GameObject.Instantiate(Bullet, transform.position, transform.rotation); //looks for "bullet" and creates entity clone
23         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
24         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
25         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
26     }
27 }
```

As you can see, our problem comes from the fact that it is calling the function with the intent of spawning multiple clones at once in one function call from line 22 to 25. Now I will change some code.



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
   Unity Script | 0 references
5  public class SpawnBullet : MonoBehaviour
6  {
7      public GameObject Spawner;
8      public GameObject Bullet;
9      public GameObject Attract;
10     // Start is called before the first frame update
       Unity Message | 0 references
11     void Start()
12     {
13
14     }
15
16     // Update is called once per frame
       Unity Message | 0 references
17     void FixedUpdate()
18     {
19         SpawnBall();
20     }
       1 reference
21     public void SpawnBall()
22     {
23         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
24         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
25         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
26         GameObject.Instantiate(Bullet, transform.position, transform.rotation);
27     }
28 }
29
```

Here I have made a change to the update function, changing it from "Update" to "FixedUpdate".

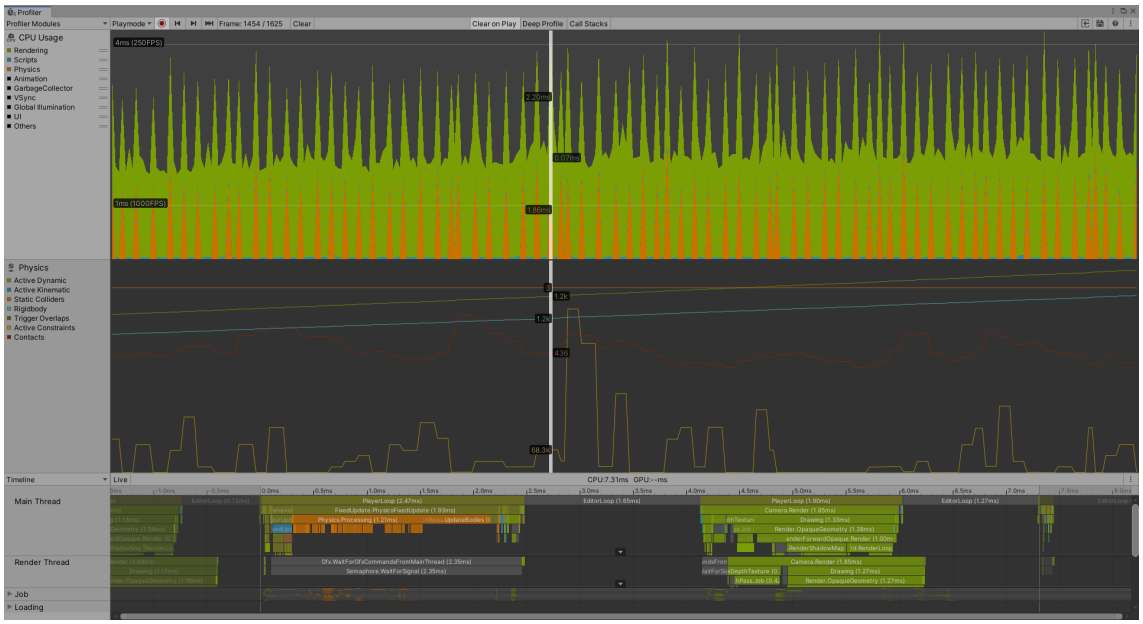
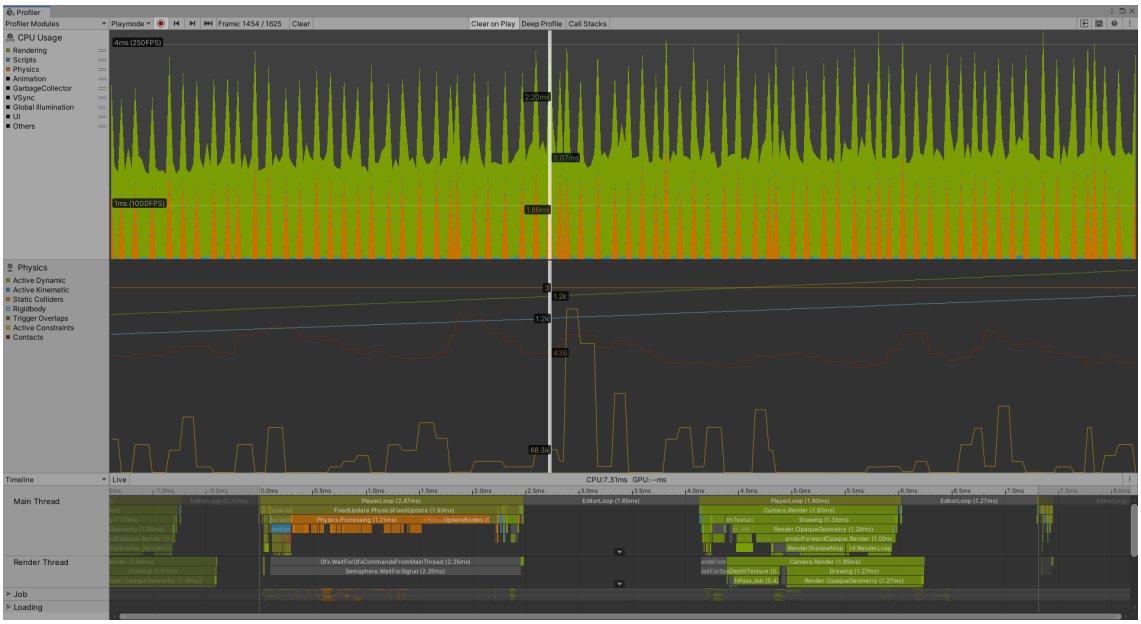


Figure 1: Here is the profiler after code changes

As you can see here, we made a big difference in performance by changing the code in our script. Originally the ms on the physics processes in the CPU view were 5.46ms and after the changes it has dropped to 1.86ms. I ran the project roughly the same time to get the profiler statistics for both images(1514 frames, 1625 frames respectively). This is roughly 65 percent faster with the new code.

The next thing we will look at is the rendering processes and how we can reduce the impact on performance.

4 Second Optimisation process



Looking at the new profiler, we can see that the rendering processes are still having a big impact on our performance, this could be happening because even if objects aren't in view, they are still being drawn. To fix this we will implement a function that removes the balls if they go below or above a certain y or x axis value. For now, the camera.render is sitting at 1.85ms in the timeline.

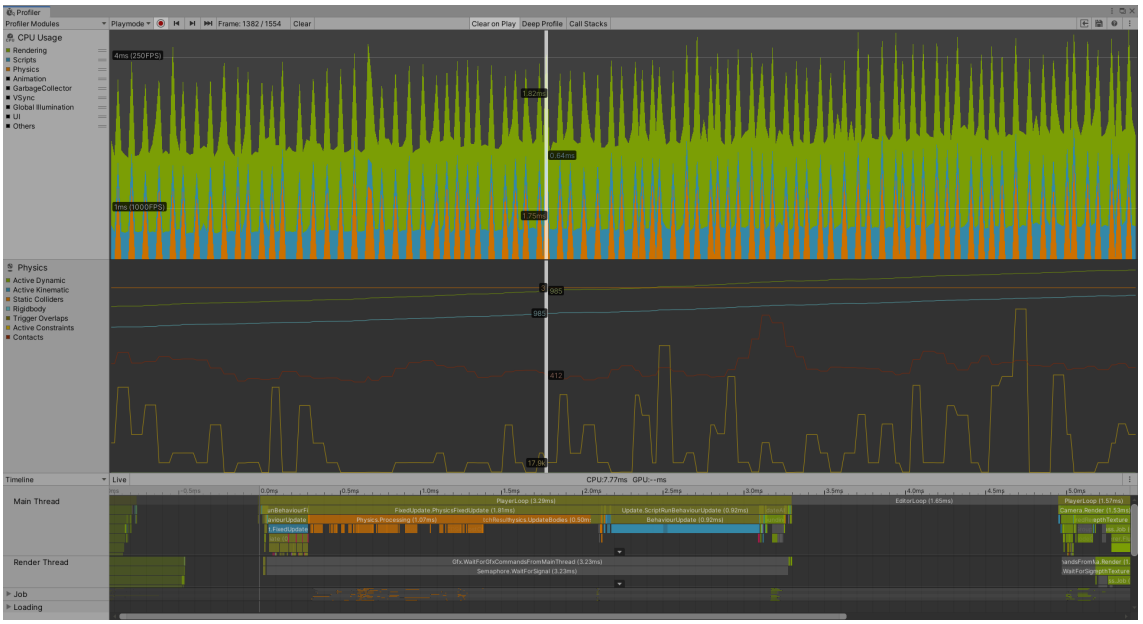
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class DestroyBullet : MonoBehaviour
{
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if(transform.position.y < 0 || transform.position.y > 15 || transform.position.x < -50)
        {
            Destroy(gameObject);
        }
    }
}
```

Here I have made another script to add to the ball prefab, I have made it so if a ball goes below y 0, above y 15 and beyond x -50, the object will be deleted.

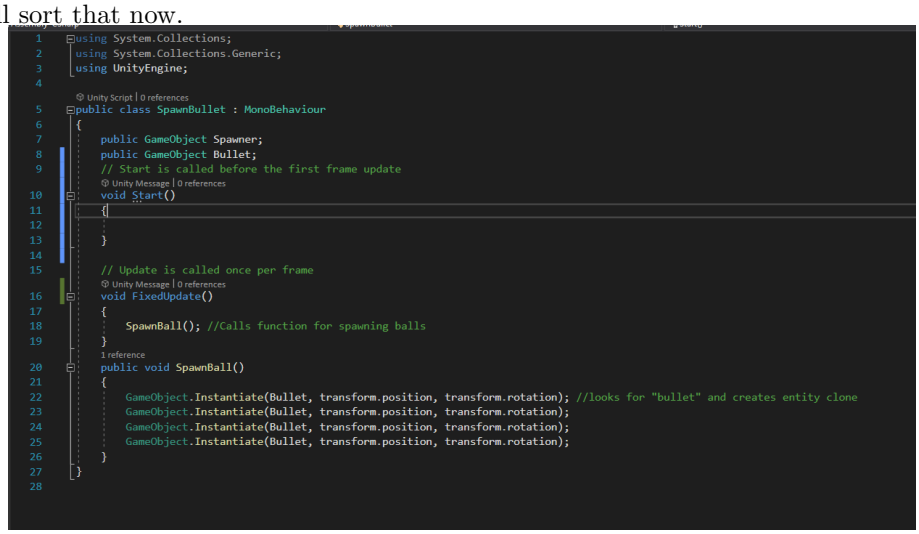


And as you can see in this new profiler window, the camera.render has dropped to 1.53ms(bottom right, might be hard to see), not a big difference but nonetheless it's better than before.

We can further improve the performance, not by much but in larger projects it may make a very noticeable difference.

5 Third optimisation process

In this third iteration, we will make small improvements to our performance, to do this we can look in our code and remove any empty functions that we aren't using. The scripts is still taking a bit of performance away from us so we will sort that now.



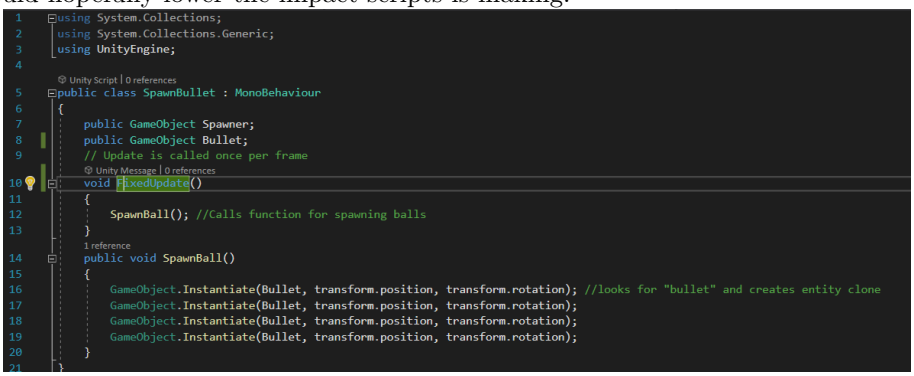
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class SpawnBullet : MonoBehaviour
{
    public GameObject Spawner;
    public GameObject Bullet;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void FixedUpdate()
    {
        SpawnBall(); //Calls function for spawning balls
    }
    1 reference
    public void SpawnBall()
    {
        GameObject.Instantiate(Bullet, transform.position, transform.rotation); //looks for "bullet" and creates entity clone
        GameObject.Instantiate(Bullet, transform.position, transform.rotation);
        GameObject.Instantiate(Bullet, transform.position, transform.rotation);
        GameObject.Instantiate(Bullet, transform.position, transform.rotation);
    }
}
```

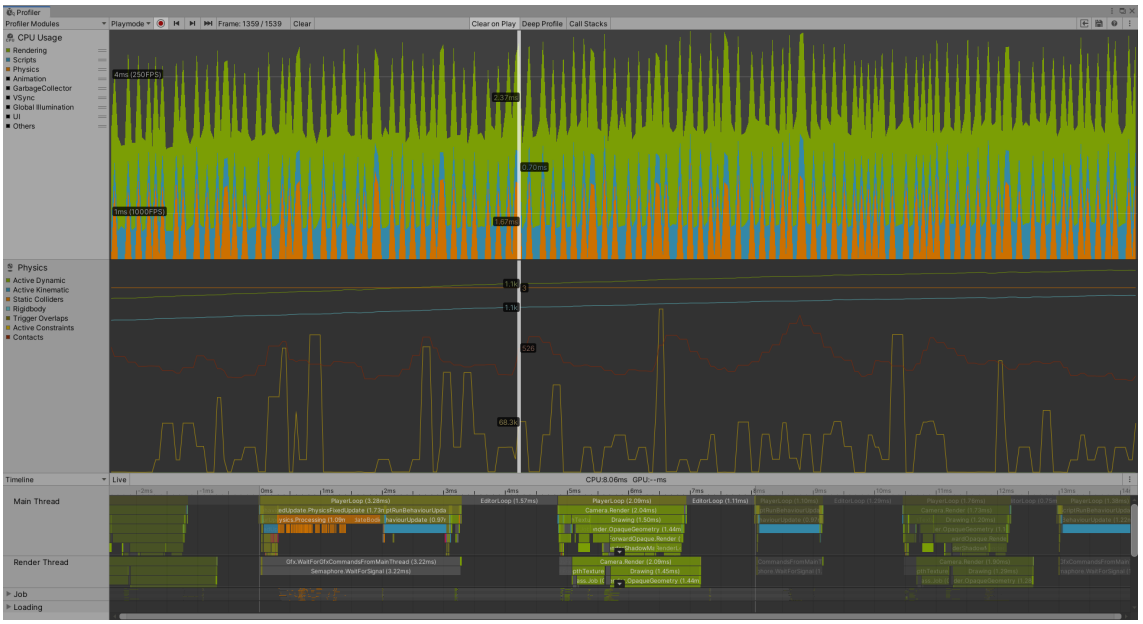
As you can see here, I have an empty function which I can remove which should hopefully lower the impact scripts is making.



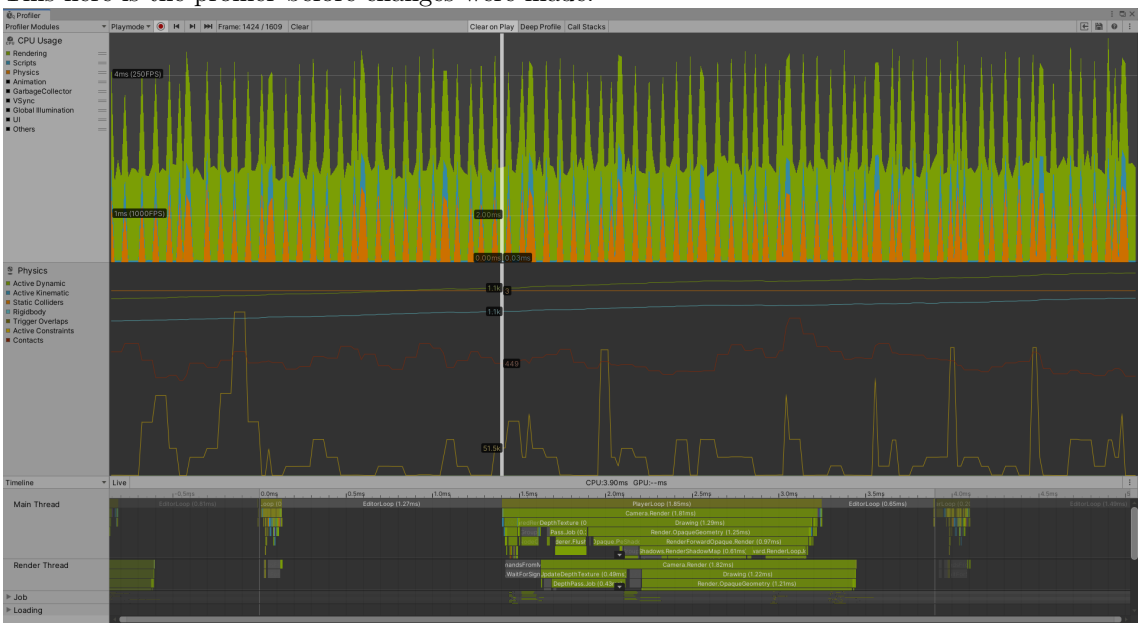
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class SpawnBullet : MonoBehaviour
{
    public GameObject Spawner;
    public GameObject Bullet;
    // Update is called once per frame
    Unity Message | 0 references
    void FixedUpdate()
    {
        SpawnBall(); //Calls function for spawning balls
    }
    1 reference
    public void SpawnBall()
    {
        GameObject.Instantiate(Bullet, transform.position, transform.rotation); //looks for "bullet" and creates entity clone
        GameObject.Instantiate(Bullet, transform.position, transform.rotation);
        GameObject.Instantiate(Bullet, transform.position, transform.rotation);
        GameObject.Instantiate(Bullet, transform.position, transform.rotation);
    }
}
```

Here, you can see that I have removed the start method, now lets take a look at the profiler.



This here is the profiler before changes were made.



And this here is the profiler after the changes, and as you can see the scripts view has dropped between each update from roughly 0.70ms to about 0.03ms.

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

Unity Script | 0 references
public class DestroyBullet : MonoBehaviour
{
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if(transform.position.y < 0 || transform.position.y > 15 || transform.position.x < -50)
        {
            Destroy(gameObject);
        }
    }
}
```

As for my other scripts "DestroyBullet", I will also do the same adjustments.

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

Unity Script | 0 references
public class DestroyBullet : MonoBehaviour
{
    // Update is called once per frame
    Unity Message | 0 references
    void FixedUpdate()
    {
        if(transform.position.y < 0 || transform.position.y > 15 || transform.position.x < -50)
        {
            Destroy(gameObject);
        }
    }
}
```

Here in this snippet, you can see that I have removed the start method but I have also changed "Update" to "FixedUpdate". Although these optimisations didn't have much of an impact in this project, I believe that in larger projects these optimisations will have a much bigger impact.