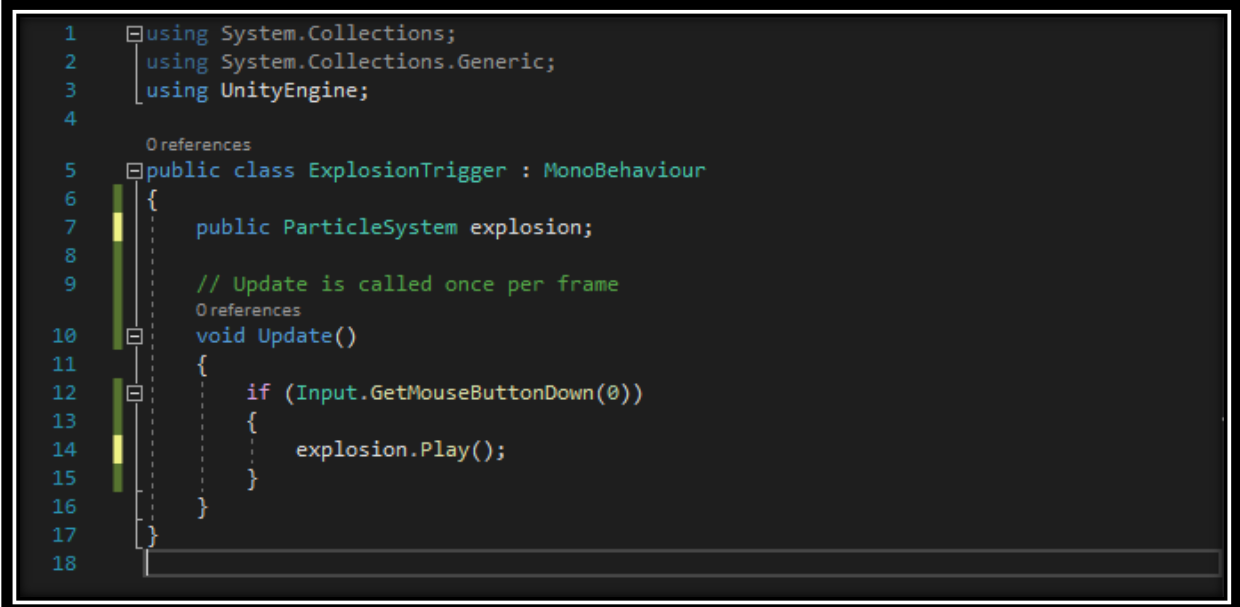

TUTORIAL FOUR – CAMERA SHAKE (Basic)

Before beginning the Camera Shake tutorial, it's important to understand [Coroutines](#) and to also have set up a [Particle System](#). Finally, you'll need the particle system to have this *ExplosionTrigger* script attached:



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ExplosionTrigger : MonoBehaviour
6  {
7      public ParticleSystem explosion;
8
9      // Update is called once per frame
10     void Update()
11     {
12         if (Input.GetMouseButtonDown(0))
13         {
14             explosion.Play();
15         }
16     }
17 }
18
```

This is a simple script that means when in play mode, whenever we click our left-mouse button the particle system will play, simulating an explosion. The point of this tutorial is to make this explosion feel better with some camera shake.

Once you have all these things setup, you can go ahead and create a C# script by right-clicking in the assets section and create > C# script. Rename it to CameraShake, drag and drop it onto our main camera within the hierarchy and open it up in visual studios.

We'll begin by deleting our start and update functions; instead of using either of these we're going to create our own function. After reading through the Coroutines documentation, you'll now understand the benefits of using this instead. So, we'll need to set up our IEnumerator function, call it 'Shake' and then also take into consideration its parameters for how long and how strong. Type as follows beneath the MonoBehaviour where start would have been:

```
public IEnumerator Shake (float duration, float magnitude)
{
}
```

Once we have our function set up, the first thing we need to do is record the original position of our camera so that unity knows where to return our camera once our shake has ended. This will follow with setting up our duration parameter and we'll use the term 'elapsed' for this. Put simply, we'll tell unity to check how much time has elapsed, or past, since our camera started shaking. And our camera can continue to shake if the time past does not exceed our set duration. So, within our curly brackets we'll type:

```
{
    Vector3 originalPos = transform.localPosition;
    float elapsed = 0.0f;
    while (elapsed < duration)
}
```

Now, we can set up the calculations for our offset, meaning we can set up how exactly our camera shake along which axis. We're only going to be using our x and y axis and so when speaking about our z axis we'll say 'originalPos.z' so that it stays as it is during the shake. We need to create some new curly brackets after our last line to hold our calculations, and then enter as follows:

```
{
    float x = Random.Range(-1f, 1f) * magnitude;
    float y = Random.Range(-1f, 1f) * magnitude;
    transform.localPosition = new Vector3(x, y, originalPos.z);
}
```

This translates to us wanting our x and y axis to shake randomly between these two constants on the axis, multiplied by the shake strength. Higher the strength, the more our camera we'll look like it's shaking. We then apply these calculations to our camera.

We will now reference back to our Coroutine through three words; yield return null. What this does is lets us run our 'while' loop of camera shake calculations alongside the update function. We'll enter:

```
{
    float x = Random.Range(-1f, 1f) * magnitude;
    float y = Random.Range(-1f, 1f) * magnitude;
    transform.localPosition = new Vector3(x, y, originalPos.z);
    yield return null
}
```

Which then tells unity, after our calculations have been called, to wait until the next frame (when the update function also is called again) before calling upon our script. Before this however, we also want to update our elapsed time, so we'll fit this in just before we wait for our next frame. Every frame we increase the elapsed time value by the time that's gone by; so:

```
{
    float x = Random.Range(-1f, 1f) * magnitude;
    float y = Random.Range(-1f, 1f) * magnitude;
    transform.localPosition = new Vector3(x, y, originalPos.z);
    elapsed += Time.deltaTime;
    yield return null
}
```

After completing our camera shake, we want unity to return our camera to its original position, so to finish our script off we'll type:

```
elapsed += Time.deltaTime;  
    yield return null  
}  
transform.localPosition = originalPos;
```

Our camera shake script is complete, but we still need to reference it in our explosion script so that it is called when appropriate; in this case when we activate our explosion through our mouse button.

Note: When I was first putting together this script, I had a usual error from unity though nothing in my script was incorrect. To fix this, I simply deleted the line that was causing the error and retyped it. Unity was then entirely okay with my script, so this is a possible solution if you come across the same error I did.

Save and return to unity. If you haven't already, ensure your camera shake script is attached to your Main Camera and then double-click on the pre-prepared ExplosionTrigger script within the assets to open it up.

Here we need to add a couple lines of code so that when we press down our left-mouse button, our CameraShake script will be called along with our Trigger script. We first need to reference our script and so beneath our particle system reference type:

```
public CameraShake cameraShake;
```

Make sure to double-check your capitalisations here.

And now within our update function, we'll need to add to explosion.Play our Coroutine. To do this, type:

```
StartCoroutine(cameraShake.Shake(.15f, 4f));
```

Our scripts are complete. Assuming there are no errors, save your script, enter back into unity, hit play, and now when you click your mouse, you'll have both your particle system explosion play along side your camera shake.

Your final scripts should look as follows:

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ExplosionTrigger : MonoBehaviour
6  {
7      public ParticleSystem explosion;
8      public CameraShake cameraShake;
9
10     // Update is called once per frame
11     void Update()
12     {
13         if (Input.GetMouseButtonDown(0))
14         {
15             explosion.Play();
16             StartCoroutine(cameraShake.Shake(.15f, 4f));
17         }
18     }
19 }
20
21

```

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class CameraShake : MonoBehaviour
6  {
7      public IEnumerator Shake (float duration, float magnitude)
8      {
9          Vector3 originalPos = transform.localPosition;
10
11          float elapsed = 0.0f;
12
13          while (elapsed < duration)
14          {
15              float x = Random.Range(-1f, 1f) * magnitude;
16              float y = Random.Range(-1f, 1f) * magnitude;
17
18              transform.localPosition = new Vector3(x, y, originalPos.z);
19
20              elapsed += Time.deltaTime;
21
22              yield return null;
23          }
24
25          transform.localPosition = originalPos;
26      }
27 }
28

```