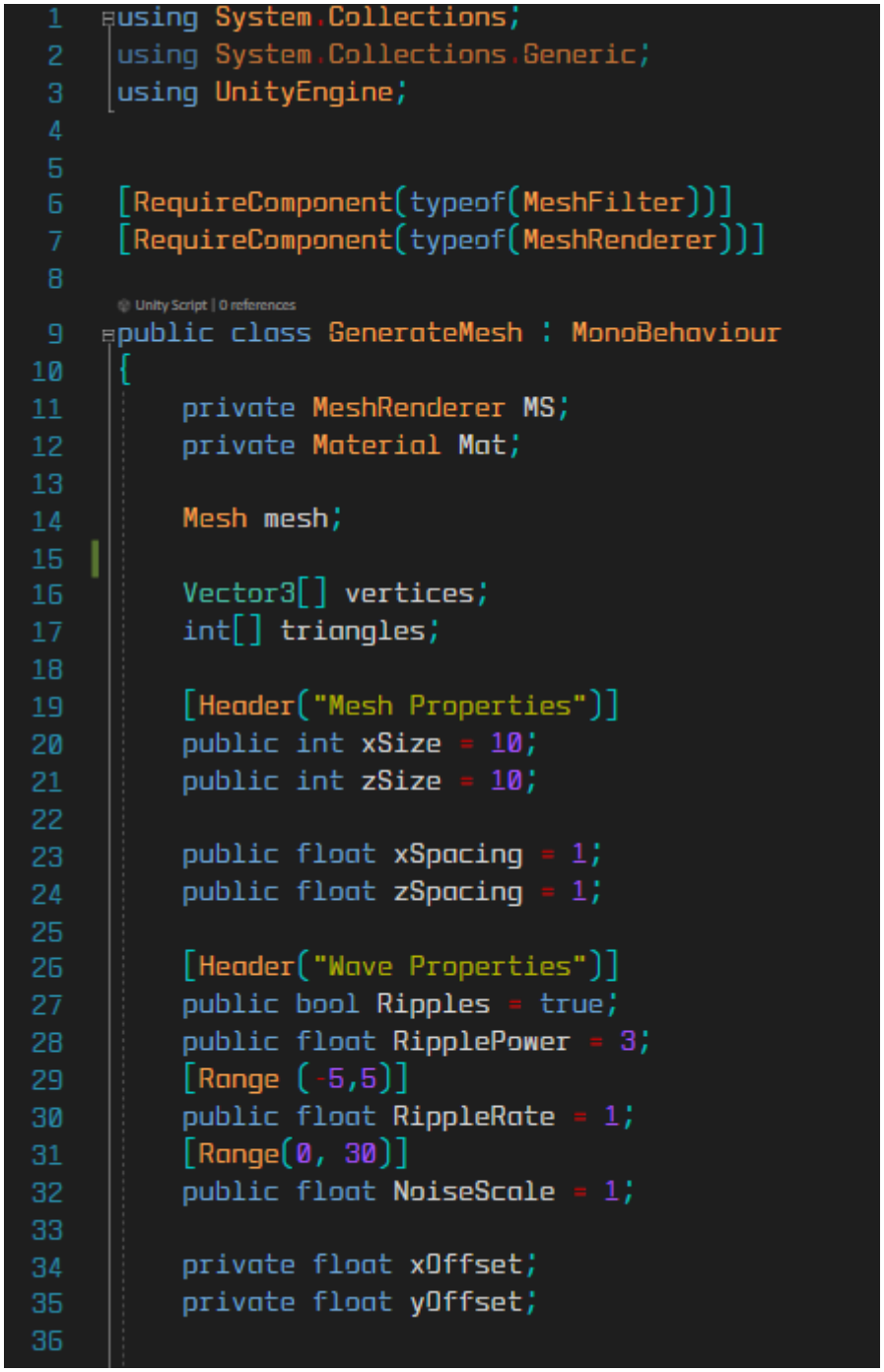
Recap and Usage

Now your wave should be fully working, for reference your plane/wave script should look like this:



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


[RequireComponent(typeof(MeshFilter))]
[RequireComponent(typeof(MeshRenderer))]

Unity Script | 0 references
public class GenerateMesh : MonoBehaviour
{
    private MeshRenderer MS;
    private Material Mat;

    Mesh mesh;

    Vector3[] vertices;
    int[] triangles;

    [Header("Mesh Properties")]
    public int xSize = 10;
    public int zSize = 10;

    public float xSpacing = 1;
    public float zSpacing = 1;

    [Header("Wave Properties")]
    public bool Ripples = true;
    public float RipplePower = 3;
    [Range (-5,5)]
    public float RippleRate = 1;
    [Range(0, 30)]
    public float NoiseScale = 1;

    private float xOffset;
    private float yOffset;

```

```
Unity Message | 0 references
void Start()
{
    MS = GetComponent<MeshRenderer>();
    Mat = MS.material;
    mesh = new Mesh();
    GetComponent<MeshFilter>().mesh = mesh;

    if (Ripples)
    {
        xOffset = Random.Range(0, 1);
        yOffset = Random.Range(0, 1);
    }
    CreateMesh();
    UpdateMesh();
}

1 reference
void CreateMesh()
{
    vertices = new Vector3[(xSize + 1) * (zSize + 1)];

    for (int i = 0, z = 0; z <= zSize; z++)
    {
        for (int x = 0; x <= xSize; x++)
        {
            vertices[i] = new Vector3(x * xSpacing, 0, z *zSpacing);
            i++;
        }
    }
```

```
    StartCoroutine(PlaneNoise(false));

    int vert = 0;
    int tris = 0;
    triangles = new int[xSize * zSize * 6];

    for (int z = 0; z < zSize; z++)
    {
        for (int x = 0; x < xSize; x++)
        {
            //triangle1
            triangles[tris + 0] = vert;
            triangles[tris + 1] = vert + xSize + 1;
            triangles[tris + 2] = vert + 1;
            //triangle2
            triangles[tris + 3] = vert + 1;
            triangles[tris + 4] = vert + xSize + 1;
            triangles[tris + 5] = vert + xSize + 2;
            //Quad Generated

            //Move start position of next vertex
            vert++;
            tris += 6;
        }
        // Skips triangle gen at end of row
        vert++;
    }

}
```

```
2 references
void UpdateMesh()
{
    mesh.Clear();
    mesh.vertices = vertices;
    mesh.triangles = triangles;
    mesh.RecalculateNormals();

    StartCoroutine(PlaneNoise(true));
}

2 references
IEnumerator PlaneNoise(bool Repeat)
{
    if (Ripples)
    {
        xOffset += Time.deltaTime * RippleRate;
        yOffset += Time.deltaTime * RippleRate;

        Mat.SetFloat("_MinY", transform.position.y);
        Mat.SetFloat("_MaxY", transform.position.y + RipplePower);

        for (int i = 0; i < vertices.Length; i++)
        {
            vertices[i].y = GenerateNoise(vertices[i].x, vertices[i].z) * RipplePower;
        }
    }
    yield return new WaitForSeconds(Time.deltaTime);
    if (Repeat)
        UpdateMesh();

}

1 reference
float GenerateNoise(float x,float y)
{
    float xCoord = x * NoiseScale + xOffset;
    float yCoord = y * NoiseScale + yOffset;

    return Mathf.PerlinNoise(xCoord, yCoord);

}
}
```

Mesh Properties

X Size and **Z Size** dictate the number of vertices to generate along each axis

X Spacing and **Z Spacing** state the distance between each vertex

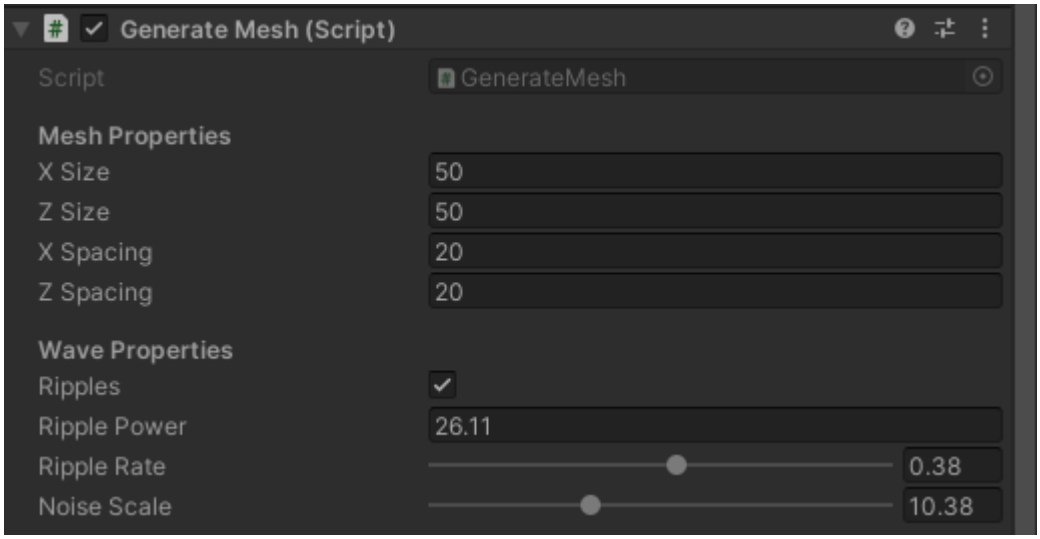
Mesh Properties

Ripples sets whether or not the wave oscillates

Ripple Power sets the max amplitude of a wave

Ripple Rate scrolls the plane at a set speed

Noise Scale spreads out the peaks of each wave from another



To set up a new wave:

Create a new empty game object and add the C# script and a material using the custom Shader and the other components should generate on the object.