

```

using UnityEngine;

using System.Collections;
using System.Collections.Generic;

public class ProceduralTerrain : MonoBehaviour {

    [Range(10, 1000)] public int TerrainSize;
    [Range( 5,  250)] public int CellSize;

    public void GenerateTerrain() {
        int x_segments = TerrainSize / CellSize;
        int z_segments = TerrainSize / CellSize;

        for (int x = 0; x < x_segments; x++) {
            for (int z = 0; z < z_segments; z++) {
                // Build up our terrain mesh
            }
        }
    }

    private float GetHeight(float x, float z, int x_segments, int z_segments) {
        return Mathf.PerlinNoise(x / (float) x_segments, z / (float) z_segments);
    }
}

```

The method that will determine the height of all four corners of each segment

```
using UnityEngine;
```

```
using System.Collections;
```

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

```
public class ProceduralTerrain : MonoBehaviour {
```

```
    [Range(10, 1000)] public int TerrainSize;
```

```
    [Range( 5, 250)] public int CellSize;
```

```
    public void GenerateTerrain() {
```

```
        int x_segments = TerrainSize / CellSize;
```

```
        int z_segments = TerrainSize / CellSize;
```

```
        for (int x = 0; x < x_segments; x++) {
```

```
            for (int z = 0; z < z_segments; z++) {
```

```
                // Build up our terrain mesh
```

```
            }
```

```
        }
```

```
    }
```

```
    private float GetHeight(float x, float z, int x_segments, int z_segments) {
```

```
        return Mathf.PerlinNoise(x / (float) x_segments, z / (float) z_segments);
```

```
    }
```

```
}
```

The logic to construct
our procedural terrain
will go here