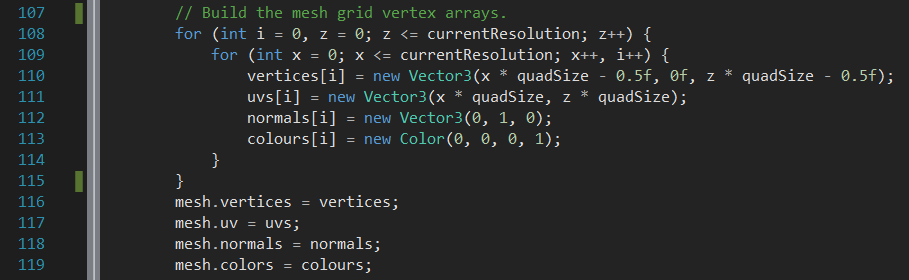
**Terrain Genesis**

**Introduction**

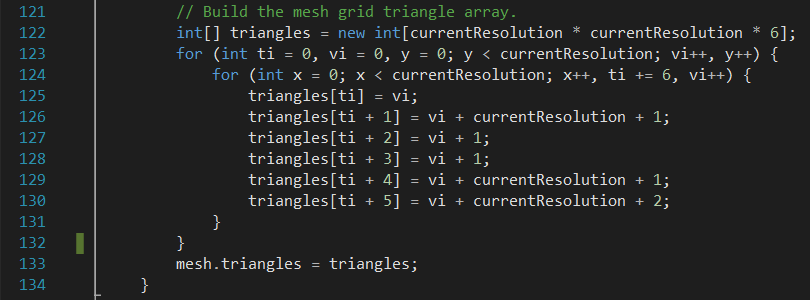
For this coursework I focused on generating a scalable mesh that could be combined with noise methods to create different terrains. This report will act as a guide to the code and will cover the key areas of the project.

**Mesh Generation – SurfaceGenerator.cs – CreateMeshGrid()**

I began by first generating a grid of quads to act as the mesh that the noise would be applied to form the terrain. This is done by first building arrays for the mesh’s vertices, uvs, and normals. I also created a colour array to distinguish the mesh from the rest of the scene. This array was a temporary measure as I went on to colour each quad corresponding to their noise value and thus height.



Similarly, the second step involves constructing the triangle array for the mesh. All of these arrays are assigned to the mesh.

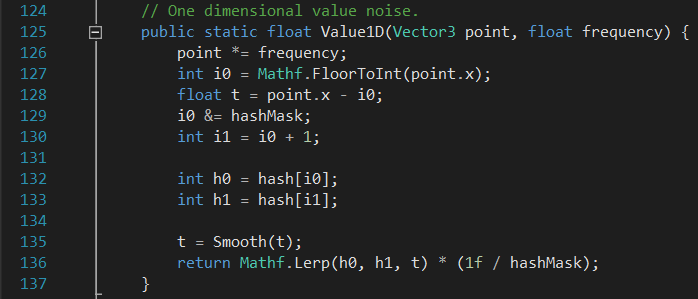


This results in a mesh surface which is scalable by altering the resolution variable.

**Noise Generation – Noise.cs**

**Value Noise**

The noise generation code can be found in the ‘Noise.cs’ class. I began by implementing value noise as a quick and easy method to test on the mesh. For one dimensional value noise this was achieved by assigning random values to the x coordinates of the mesh, and interpolating between the adjacent lattice points. I assigned ‘random’ values through the use of a permutation table.



It was then trivial to expand this to greater dimensions.

**Perlin Noise**

I wanted to include a type of gradient noise as well as simple value noise. The combination of only intending to implement this gradient noise up to the third dimension and having previously covered Simplex noise in detail meant that Perlin noise was good fit for this application.