For each vertex we:

1. Generate perlin noise

2. Shift the value range from 0 - 1 to -0.5 - 0.5

3. Determine the difference between the falloff value and the height value

4. Clamp that value between 0 and 1

```
    frequency *= Lacunarity;
}

if (UseFalloffMap) {
    float falloff_00 = Mathf.PerlinNoise(x,      z     ) - 0.5f;
    float falloff_01 = Mathf.PerlinNoise(x,      z + 1f) - 0.5f;
    float falloff_10 = Mathf.PerlinNoise(x + 1f, z     ) - 0.5f;
    float falloff_11 = Mathf.PerlinNoise(x + 1f, z + 1f) - 0.5f;

    height00 -= Mathf.Clamp01(height00 - falloff_00) * 0.5f;
    height01 -= Mathf.Clamp01(height01 - falloff_01) * 0.5f;
    height10 -= Mathf.Clamp01(height10 - falloff_10) * 0.5f;
    height11 -= Mathf.Clamp01(height11 - falloff_11) * 0.5f;
}

int x0 =  x      * CellSize;
```

For each vertex we:

1. Generate perlin noise

2. Shift the value range from 0 - 1 to -0.5 - 0.5

3. Determine the difference between the falloff value and the height value

4. Clamp that value between 0 and 1

5. Half the clamped value

```
    frequency *= Lacunarity;
}

if (UseFalloffMap) {
   float falloff_00 = Mathf.PerlinNoise(x,      z     ) - 0.5f;
   float falloff_01 = Mathf.PerlinNoise(x,      z + 1f) - 0.5f;
   float falloff_10 = Mathf.PerlinNoise(x + 1f, z     ) - 0.5f;
   float falloff_11 = Mathf.PerlinNoise(x + 1f, z + 1f) - 0.5f;

   height00 -= Mathf.Clamp01(height00 - falloff_00) * 0.5f;
   height01 -= Mathf.Clamp01(height01 - falloff_01) * 0.5f;
   height10 -= Mathf.Clamp01(height10 - falloff_10) * 0.5f;
   height11 -= Mathf.Clamp01(height11 - falloff_11) * 0.5f;
}

int x0 =  x      * CellSize;
```