```
// ProceduralTerrain



                                              Changing these values will move
                                              the falloff floor up and down
   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;
```

Changing these values will move the falloff floor up and down

As the value approaches 1 the floor moves down along the Y axis

```
        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;
```