// Procedural Terrain.shader

```
Shader "MinTuts/Procedural Terrain" {
  Properties {
    _WaterLimit("Water Limit", Range(0.000001, 0.05)) = 0.01
    _ShoreLimit("Shore Limit", Range(0.05,     0.1 )) = 0.05
  }

  SubShader {
    Pass {
…
       #include "UnityCG.cginc"

       float _WaterLimit;
       float _ShoreLimit;

       struct v2f {

…
```

With _**WaterLimit** and _**ShoreLimit** defined in the Properties section, we must now define them in the **SubShader > Pass > CGPROGRAM** section

This may seem redundant, but it makes sense when you consider that the Properties section exists only to link shader properties to Unity's material inspector

The Properties section does not define anything in the scope of the shader; each line simply says…

*"This shader property…*
*should be displayed in the inspector using this name,*
*the Property Drawer should of this type,*
*and this should be it's default value"*

Nothing on this line in specified in the context of our **CGPROGRAM**, so we must…
define the property being described in **SubShader > Pass > CGPROGRAM**…
so **ShaderLab** can properly link everything up *and*
we have _**WaterLimit** defined and usable in the scope of our shader

## // Procedural Terrain.shader

```
Shader "MinTuts/Procedural Terrain" {
  Properties {
    _WaterLimit("Water Limit", Range(0.000001, 0.05)) = 0.01
    _ShoreLimit("Shore Limit", Range(0.05,     0.1 )) = 0.05
  }

  SubShader {
    Pass {
…
       #include "UnityCG.cginc"

       float _WaterLimit;
       float _ShoreLimit;

       struct v2f {

…
```
~~`if (p < 0.01) {`~~
```
       if (p < _WaterLimit) {

…
```

With all that taken care of we can now get rid of the hard-coded value for the water's edge and replace it with **_WaterLimit**