

// MinTuts/Procedural Terrain.shader

```
Shader "MinTuts/Procedural Terrain" {
    SubShader {
        Pass {
            CGPROGRAM

            #pragma vertex    vert
            #pragma fragment  frag

            #include "UnityCG.cginc"

            struct v2f {
                float4 pos    : SV_POSITION;
                float3 wpos   : POSITION1;
            };

            v2f vert(float4 vertex : POSITION) {
                v2f o;

                o.pos = UnityObjectToClipPos(vertex);
                o.wpos = mul(unity_ObjectToWorld, vertex);

                return o;
            }

            float4 frag(v2f i) : COLOR {
                float  p = i.wpos.y * 0.015;
                float3 y = float3(p, p, p);

                return float4(y, 1);
            }

        }
    }
}
```

The **Pass** section(s) specify the logic for a **Shader**

If multiple **Pass** sections are specified they are executed in order from top to bottom

NOTE: Specifying multiple **Pass** sections gets expensive quick; whenever possible limit your **SubShader** to a single **Pass** (transparency effects are one of the few cases where multiple **Pass** sections are required)

// MinTuts/Procedural Terrain.shader

```
Shader "MinTuts/Procedural Terrain" {
  SubShader {
    Pass {
      CGPROGRAM

      #pragma vertex   vert
      #pragma fragment frag

      #include "UnityCG.cginc"

      struct v2f {
        float4 pos      : SV_POSITION;
        float3 wpos     : POSITION1;
      };

      v2f vert(float4 vertex : POSITION) {
        v2f o;

        o.pos  = UnityObjectToClipPos(vertex);
        o.wpos = mul(unity_ObjectToWorld, vertex);

        return o;
      }

      float4 frag(v2f i) : COLOR {
        float  p = i.wpos.y * 0.015;
        float3 y = float3(p, p, p);

        return float4(y, 1);
      }
    }
  }
}
```

CGPROGRAM/ENDCG specify *where* the **Cg** code **Unity** needs to compile is

Everything between **CGPROGRAM/ENDCG** is **NVIDIA's Cg** (aka *C for graphics*)

You can mix **HLSL** (*High-Level Shader Language* or *High-Level Shading Language*; aka **Microsoft's DirectX**) and **Cg**