

// 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);
            }

        }
    }
}
```

Here we are defining the **vert** function...
specified by the **#pragma** definition
from earlier

This function has a single argument...
of the type **float4**...

with a **semantic** filter of **POSITION**
(the same semantic meaning as
SV_POSITION)...

returning the **v2f** structured data type
defined earlier

The expanded name of **v2f**, vertex2fragment,
communicates something crucially important
about what this function returns

It is the output of the **vert** function,
and the input to the **frag** function

This data has a single purpose: pass data
from the **vertex** function... to the **fragment**
function

// 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);
            }

        }
    }
}
```

First things first: create our output
data structure