

## // 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 **frag** function...  
specified by the **#pragma** definition  
from earlier

This function has a single argument...  
of the type **v2f**...

with a **semantic** filter of **COLOR**...  
and returns a **float4** (representing the 4  
color channels: red, green, blue, and  
alpha/transperancy)

## // 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 first thing we do is convert **i.wpos.y** (**i.wpos**' vertical axis) to a value between 0 and 1

Well... between 0 and 1.5 in our case

To “normalize” (convert to a value between 0 and 1) **i.wpos.y** we would need to multiply it by 0.01 (the minimum value for our vertical axis is 0 and the maximum value is 100, so multiplying i.wpos.y by 0.01 shrinks the value space to the range 0 - 1)

I chose 0.015 because I think it looks better in this case

**NOTE:** Cases like this are pretty rare; typically you want to ensure your **normalized** value never goes above 1