

// 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 a data structure

This data structure is named **v2f**; which is short for *vertex2fragment*

It has two properties; **pos** (of type **float4**) and **wpos** (of type **float3**)

These properties have **semantics** along with their types

The best way to think about **semantics** is as a filter on top of a type

For example: **pos** has a type of **float4** with a **semantic** filter of **SV_POSITION**

The **semantic** meaning of **SV_POSITION** is: *clip space position*, or where the *current vertex* is located *relative to the camera's field of view*

Another example: **wpos** has a type of **float3** with a **semantic** filter of **POSITION1**

The **semantic** meaning of **POSITION1** is: *world space position*, or where the *current vertex* is located *in the world*

NOTE: We don't use the **pos** property, but a **semantic** of **SV_POSITION** must be included to work with **DirectX**

NOTE: A semicolon must terminate each line defining a property as well as the line terminating the definition of the **struct**

// 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);
                }

            ENDCG
        }
    }
}
```

Here we are defining the **vert** function...