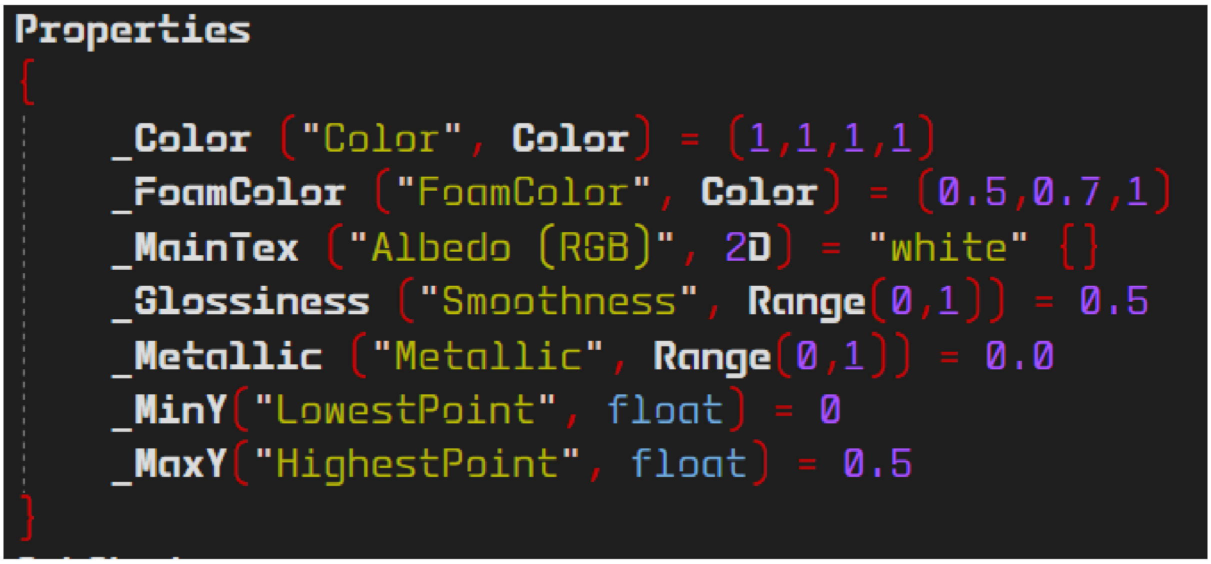
Shading Waves Between Two Colours Based On Height

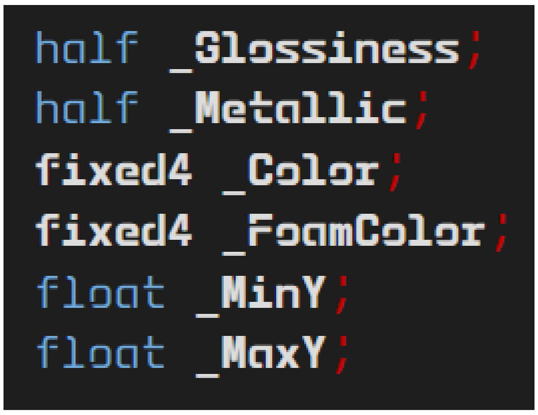
***This tutorial uses a custom Shader written in CG using Unity ShaderLab.**

First we need to define all our variable we need to use in a void called properties in the format: *VariableName*(*InspectorName*,*DataType*).



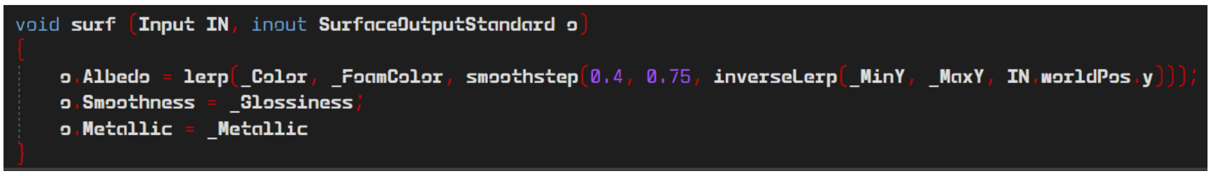
```
Properties
{
    _Color ("Color", Color) = (1,1,1,1)
    _FoamColor ("FoamColor", Color) = (0.5,0.7,1)
    _MainTex ("Albedo (RGB)", 2D) = "white" {}
    _Glossiness ("Smoothness", Range(0,1)) = 0.5
    _Metallic ("Metallic", Range(0,1)) = 0.0
    _MinY("LowestPoint", float) = 0
    _MaxY("HighestPoint", float) = 0.5
}
```

Now we need to redefine our variables to be used in our Subshader Class.



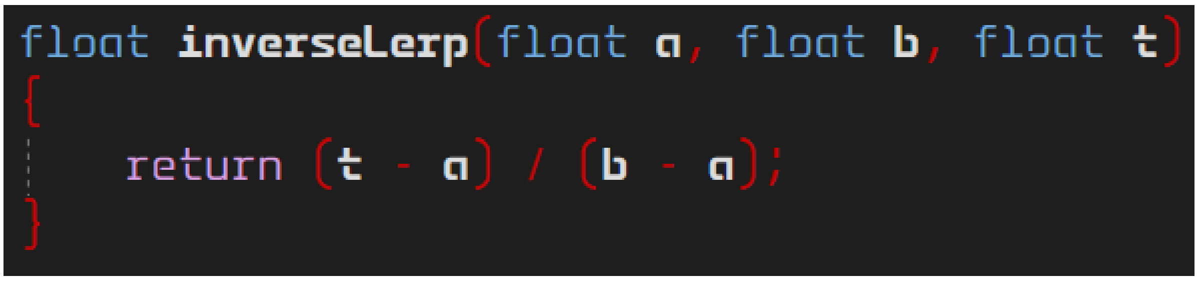
```
half _Glossiness;
half _Metallic;
fixed4 _Color;
fixed4 _FoamColor;
float _MinY;
float _MaxY;
```

In our main function, we set the color of our object to smoothstep between two colour starting at the y component of world position



```
void surf (Input IN, inout SurfaceOutputStandard o)
{
    o.Albedo = lerp(_Color, _FoamColor, smoothstep(0.4, 0.75, inverseLerp(_MinY, _MaxY, IN.worldPos.y)));
    o.Smoothness = _Glossiness;
    o.Metallic = _Metallic
}
```

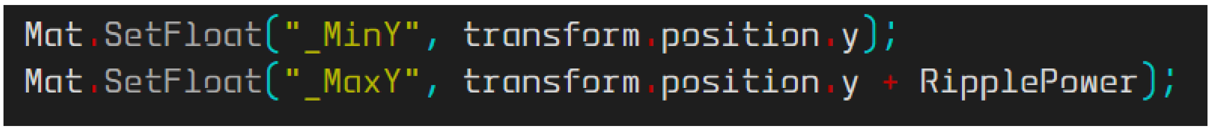
Our InverseLerp function returns a value to step between based on the lowest point, highest point and mid point of our object.



```
float inverseLerp(float a, float b, float t)
{
    return (t - a) / (b - a);
}
```

Inside of our plane noise script

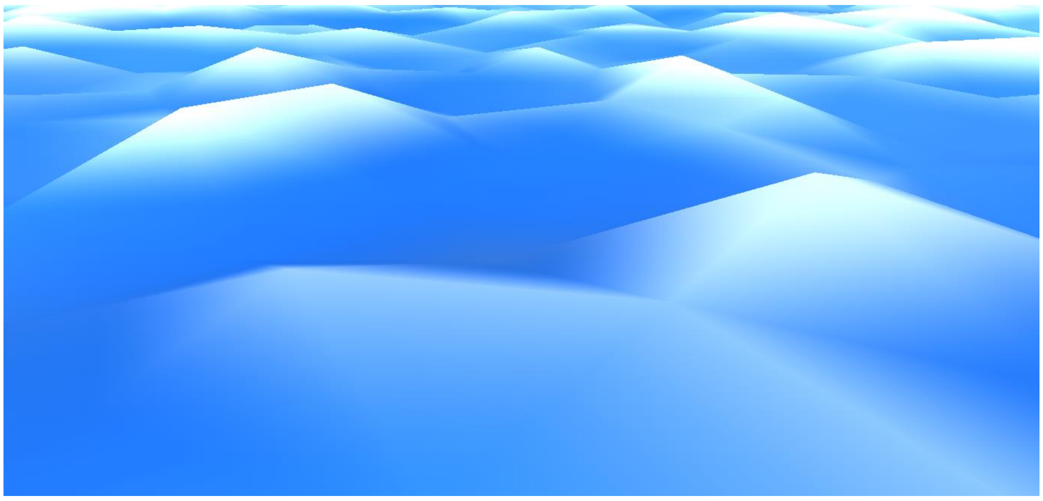
Now we need to set the top and bottom of our plane which should be the world position y value and maximum possible amplitude of a wave.

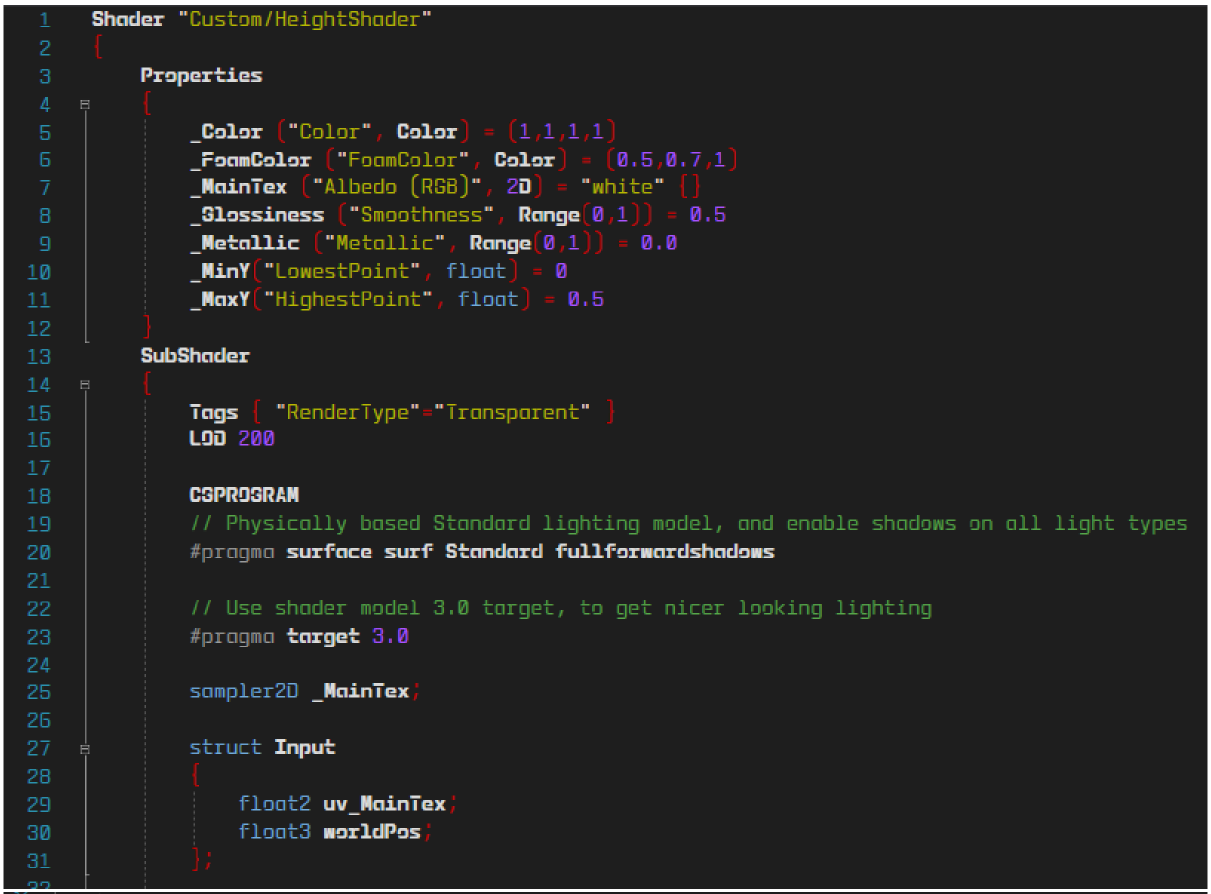


```
Mat.SetFloat("_MinY", transform.position.y);
Mat.SetFloat("_MaxY", transform.position.y + RipplePower);
```

Recap:

Along with the default Shader code your complete Shader and wave should look like this,





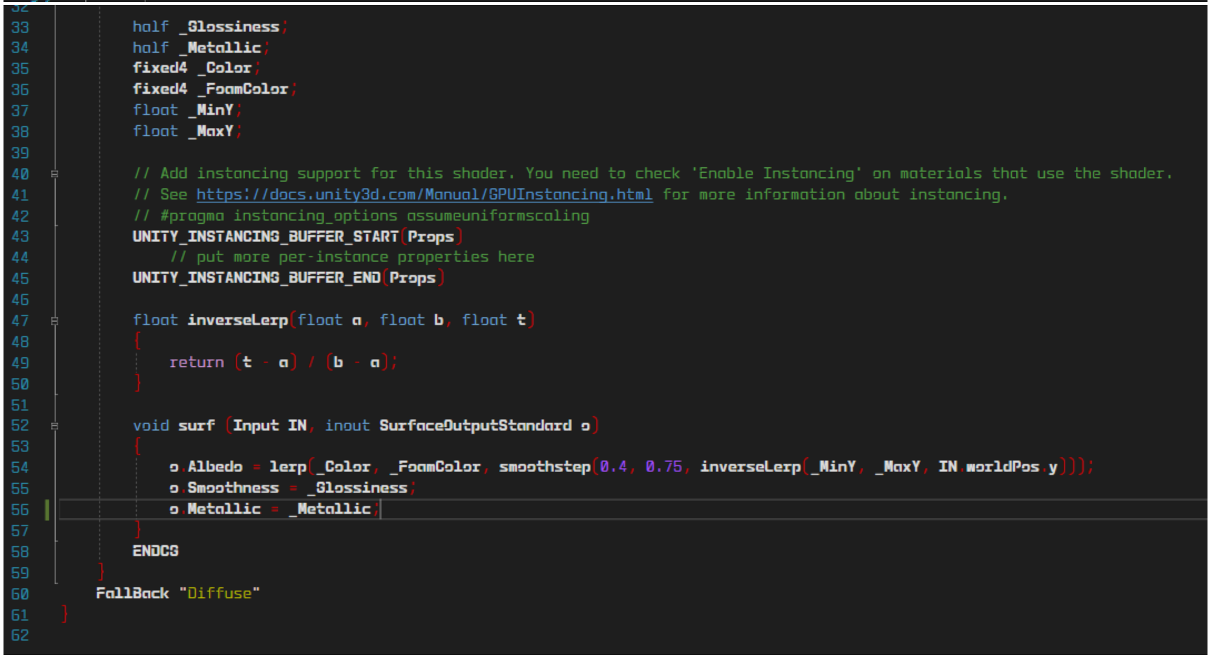
```
Shader "Custom/HeightShader"
{
    Properties
    {
        _Color ("Color", Color) = (1,1,1,1)
        _FoamColor ("FoamColor", Color) = (0.5,0.7,1)
        _MainTex ("Albedo (RGB)", 2D) = "white" {}
        _Glossiness ("Smoothness", Range(0,1)) = 0.5
        _Metallic ("Metallic", Range(0,1)) = 0.0
        _MinY("LowestPoint", float) = 0
        _MaxY("HighestPoint", float) = 0.5
    }
    SubShader
    {
        Tags { "RenderType"="Transparent" }
        LOD 200

        CGPROGRAM
        // Physically based Standard lighting model, and enable shadows on all light types
        #pragma surface surf Standard fullforwardshadows

        // Use shader model 3.0 target, to get nicer looking lighting
        #pragma target 3.0

        sampler2D _MainTex;

        struct Input
        {
            float2 uv_MainTex;
            float3 worldPos;
        };
```



```
        half _Glossiness;
        half _Metallic;
        fixed4 _Color;
        fixed4 _FoamColor;
        float _MinY;
        float _MaxY;

        // Add instancing support for this shader. You need to check 'Enable Instancing' on materials that use the shader.
        // See https://docs.unity3d.com/Manual/GPUInstancing.html for more information about instancing.
        // #pragma instancing_options assumeuniformscaling
        UNITY_INSTANCING_BUFFER_START(Props)
            // put more per-instance properties here
        UNITY_INSTANCING_BUFFER_END(Props)

        float inverseLerp(float a, float b, float t)
        {
            return (t - a) / (b - a);
        }

        void surf (Input IN, inout SurfaceOutputStandard o)
        {
            o.Albedo = lerp(_Color, _FoamColor, smoothstep(0.4, 0.75, inverseLerp(_MinY, _MaxY, IN.worldPos.y)));
            o.Smoothness = _Glossiness;
            o.Metallic = _Metallic;
        }
        ENDCG
    }
    FallBack "Diffuse"
}
```