

## // Procedural Terrain.shader

```
Shader "MinTuts/Procedural Terrain" {
```

```
    Properties {
```

```
        _WaterLimit("Water Limit", Range(0.000001, 0.05)) = 0.01
```

```
        _ShoreLimit("Shore Limit", Range(0.05, 0.1)) = 0.05
```

```
    }
```

```
    SubShader {
```

```
        Pass {
```

```
...
```

```
            #include "UnityCG.cginc"
```

```
            float _WaterLimit;
```

```
            float _ShoreLimit;
```

```
            struct v2f {
```

```
...
```

```
            if (p < 0.01) {
```

```
            if (p < _WaterLimit) {
```

```
...
```

With all that taken care of we can now get rid of the hard-coded value for the water's edge and replace it with **\_WaterLimit**

Now, whenever **"Water Limit"** is updated via the inspector, this test will stay in sync and correct

## // Procedural Terrain.shader

```
Shader "MinTuts/Procedural Terrain" {
```

```
    Properties {
```

```
        _WaterLimit("Water Limit", Range(0.000001, 0.05)) = 0.01
```

```
        _ShoreLimit("Shore Limit", Range(0.05, 0.1 )) = 0.05
```

```
    }
```

```
    SubShader {
```

```
        Pass {
```

```
...
```

```
            #include "UnityCG.cginc"
```

```
            float _WaterLimit;
```

```
            float _ShoreLimit;
```

```
            struct v2f {
```

```
...
```

```
            if (p < 0.01) {
```

```
            if (p < _WaterLimit) {
```

```
...
```

With all that taken care of we can now get rid of the hard-coded value for the water's edge and replace it with **\_WaterLimit**

Now, whenever **"Water Limit"** is updated via the inspector, this test will stay in sync and correct

**NOTE:** Since our shader is executed for every vertex and fragment every frame we don't need to add an auto-update feature like we did when changing properties of our terrain mesh