

// MinTuts/Procedural Terrain.shader

```
    y = float3(1, 1, 1);  
} else if (p < 0.05) {  
    r = -(p - 0.1);  
    g = r;  
  
    y = float3(1, 1, 1);  
}  
  
return float4(y * float3(r, g, b), 1);
```

The goal of this commit to create a small shoreline between the water and grass

To do that we first need to make sure our previous if didn't match

If it didn't, we check if...

p is less than where we want the top of our shoreline to be

If it is, we subtract 0.1 from p

This will result in a negative number which grows larger as p approaches 0

We then flip the sign of our resulting value

The result of this flip is larger positive numbers the closer p gets to 0.01 - and smaller positive numbers as p approaches 0.05

This gives us a gradient that goes in the opposite direction from the grass gradient: light to dark as p increases

We assign the result of these calculations to the red channel (r)

We then assign r to g (this is what gives us the brown color)

// MinTuts/Procedural Terrain.shader

```
y = float3(1, 1, 1);  
} else if (p < 0.05) {  
    r = -(p - 0.1);  
    g = r;  
  
    y = float3(1, 1, 1);  
}  
  
return float4(y * float3(r, g, b), 1);
```

The goal of this commit to create a small shoreline between the water and grass

To do that we first need to make sure our previous if didn't match

If it didn't, we check if...

p is less than where we want the top of our shoreline to be

If it is, we subtract 0.1 from p

This will result in a negative number which grows larger as p approaches 0

We then flip the sign of our resulting value

The result of this flip is larger positive numbers the closer p gets to 0.01 - and smaller positive numbers as p approaches 0.05

This gives us a gradient that goes in the opposite direction from the grass gradient: light to dark as p increases

We assign the result of these calculations to the red channel (r)

We then assign r to g (this is what gives us the brown color)

As before, we max y out for brightness