// Procedural Terrain.shader

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

    _ShoreMultiplier    ("Shore Multiplier",     Range(1,      4   )) = 2
    _IntensityMultiplier("Intensity Multiplier", Range(0.0001, 0.02)) = 0.015
}

…
        float _ShoreLimit;

        float _ShoreMultiplier;
        float _IntensityMultiplier;

…
```

As with the **Limit** properties, we tell **ShaderLab**…
what shader properties to look for…
what the labels to display for the properties in the inspector should be…
what **Property Drawer** type to use…
and the default values to assign to the properties

NOTE: For **_ShoreMultiplier** the range min, range max,
and default value…
could be either integers or floating point numbers

How does **ShaderLab**/Unity know which of these types to use for this range?

We tell it explicitly when we define **_ShoreMultiplier** in the **SubShader > Pass > CGPROGRAM** section

// Procedural Terrain.shader

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

    _ShoreMultiplier    ("Shore Multiplier",     Range(1,      4   )) = 2
    _IntensityMultiplier("Intensity Multiplier", Range(0.0001, 0.02)) = 0.015
}

…
        float _ShoreLimit;

        float _ShoreMultiplier;
        float _IntensityMultiplier;

…
```

As with the **Limit** properties, we tell **ShaderLab**…
what shader properties to look for…
what the labels to display for the properties in the inspector should be…
what **Property Drawer** type to use…
and the default values to assign to the properties

NOTE: For **_ShoreMultiplier** the range min, range max,
and default value…
could be either integers or floating point numbers

How does **ShaderLab**/Unity know which of these types to use for this range?

We tell it explicitly when we define **_ShoreMultiplier** in the **SubShader > Pass > CGPROGRAM** section

Without this type declaration **ShaderLab**/Unity would have no way to determine which data type is correct in situations like this