Custom Shaders

Destructible 2D works by blending the Main Tex with the current Alpha Tex. By default, shaders in Unity don't use a separate Alpha Tex, so if you want to use custom shaders then you will have to make some simple modifications to them.

Step 1 - Update your shader properties

In your shader's Property { ... } block, you need to add the following properties:

```
[PerRendererData] _AlphaTex ("Alpha Tex", 2D) = "white" {}
[PerRendererData] _AlphaScale ("Alpha Scale", Vector) = (1,1,0,0)
[PerRendererData] _AlphaOffset ("Alpha Offset", Vector) = (0,0,0,0)
[PerRendererData] _AlphaSharpness ("Alpha Sharpness", Float) = 1.0
```

Step 2 - Update your variable declarations

In your shader's variable section (e.g. where you should have sampler2D _MainTex; or similar), add the following variables:

```
sampler2D _AlphaTex;
float2 _AlphaScale;
float2 _AlphaOffset;
float _AlphaSharpness;
```

Step 3 - Update your fragment or surface function

Inside your fragment function, e.g. fixed4 frag(v2f IN) : SV_Target { … }

Or inside surface function, e.g. void surf (Input IN, inout SurfaceOutput o) { … }

You need to multiply your final alpha like this:

```
float2 alphaUV  = (i.vertex.xy - _AlphaOffset) / _AlphaScale;
float4 alphaTex = tex2D(_AlphaTex, alphaUV);

myFinalColour.a *= saturate(0.5f + (alphaTex.a  0.5f) * _AlphaSharpness);

return myFinalColour;
```

or like this:

```
float2 alphaUV  = (i.vertex.xy - _AlphaOffset) / _AlphaScale;
float4 alphaTex = tex2D(_AlphaTex, alphaUV);

o.Alpha *= saturate(0.5f + (alphaTex.a  0.5f) * _AlphaSharpness);
```

NOTE: Make sure this is done AFTER setting the initial alpha value, otherwise it will be overwritten.
NOTE: Make sure the UV variable (e.g. i.texcoord) is correct, as it may change depending on the shader.