Making custom Curved World shaders is very easy.
Everything what is need - transform vertex before use.

```
fragmentInput vert(vertexInput i)
{
    fragmentInput o;
    UNITY_INITIALIZE_OUTPUT(fragmentInput,o);

    //CurvedWorld vertex transform
    V_CW_TransformPoint(i.vertex);

    o.position = mul (UNITY_MATRIX_MVP, i.vertex);
    o.uv = i.texcoord0.xy;
    return o;
}
```

(Simple vertex/fragment shader)

```
void vert (inout appdata_full v, out Input o)
{
    UNITY_INITIALIZE_OUTPUT(Input,o);

    //CurvedWorld vertex transform
    V_CW_TransformPointAndNormal(v.vertex, v.normal, v.tangent);
}
```

(Surface shader)

Available two functions for vertex transformation:
`inline void V_CW_TransformPoint(inout float4 vertex)`
Transforms only vertex, suitable for unlit shaders.

`inline void V_CW_TransformPointAndNormal(inout float4 vertex, inout float3 normal, float4 tangent)`
Transforms vertex and normal, suitable for shaders that require correct bended normal for calculating: light, shadow, reflection etc.

Both function and all required variables are provided inside CurvedWorld_Base.cginc file.
Variables inside CurvedWorld_Base.cginc file are updated by CurvedWorld_Controller script.

Curved World bend type is defined in CurvedWorld_Base.cginc file:
`/*DO NOT DELETE - CURVED WORLD BEND TYPE*/ #define V_CW_BENDTYPE_UNIVERSAL`
It is updated by changing projects bend type from - Menu / Edit / Preferences / Curved World

Check two example shaders inside Shaders/Example folder:

1. `"Custom/Example_Unlit"`
2. `"Custom/Example_Surface"`

Do not forget to provide correct fallback for custom shaders or use Curved World's fallback:

1. `Fallback "Hidden/VacuumShaders/Curved World/VertexLit/Diffuse"`
2. `Fallback "Hidden/VacuumShaders/Curved World/VertexLit/Cutout"`
3. `Fallback "Hidden/VacuumShaders/Curved World/VertexLit/Transparent"`

Do not modify original .shader and .cginc files!