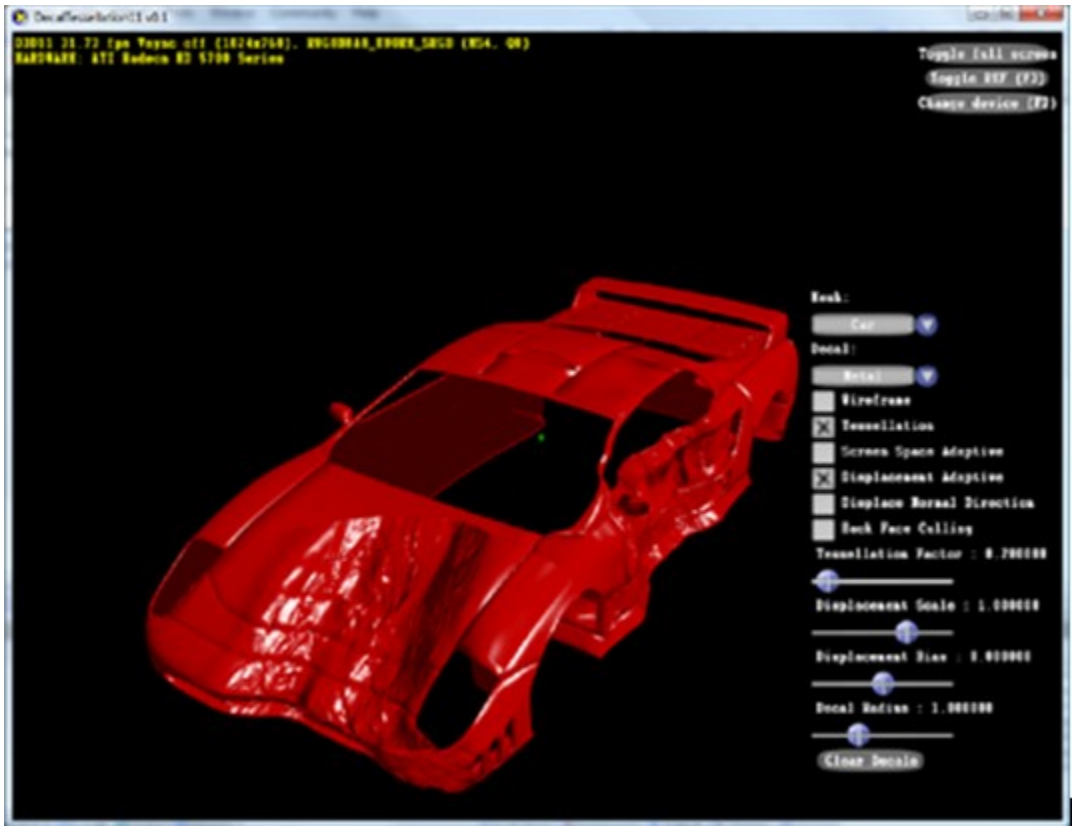
DecalTessellation11 Sample

Collapse All

This sample, contributed by AMD, presents a technique for applying tessellation with displacement mapping using decals. Given an arbitrary mesh, the sample uses a ray cast to find the location of where a ray from the eye point intersects the mesh. A decal is placed at the intersection and the mesh is rendered with tessellation so that the triangle density is high enough for accurate displacement mapping. One possible use for this technique is to represent realistic looking 3D damage on objects.



Path

Source	SDK root\Samples\C++\Direct3D11\DecalTessellation11
Executable	SDK root\Samples\C++\Direct3D11\Bin\x86 or x64\DecalTessellation11.exe

Hull Shader

The control point phase of the hull shader is a simple pass through shader that passes control points data to the domain shader. The patch constant phase of the hull shader performs important optimizations that adaptively modify the tessellation factors to reduce unnecessary tessellation. There are three types of adaptive tessellation techniques used in the hull shader:

1. Displacement Adaptive: This type of adaptive tessellation is specific to the decal tessellation algorithm. For each patch, the hull shader checks to see if a decal will be placed on the patch. If not, the tessellation factor is set to one, which causes no additional triangles to be generated for the patch. The end result is that only patches that have decals on them are tessellated.
2. Back Face Culling: Patches with all edge normals facing backwards, based upon a supplied tolerance, are culled. This can lead to significant performance gains, especially at higher levels of tessellation, since the following Domain Shader work is skipped.
3. Screen Space Adaptive: Rasterization hardware is not efficient at rendering tiny triangles, let alone sub-pixel triangles, therefore it is prudent to ensure that generated triangle sizes are kept to sensible sizes. Supplying a target edge length for a triangle ensures that models keep close to the desired triangle size at varying distance and screen resolution.

Fixed Function Tessellator

The fixed function HW tessellator stage produces new geometry based on the input tessellation factors, as barycentric coordinates in the patch domain.

Domain Shader

The domain shader stage uses the barycentric coordinates combined with the patch control point vertices to generate position, texture coordinates and normal. If a decal is placed on the patch, the vertex position will be translated, using the displacement map associated with the decal. To determine if a decal should be used, the domain shader iterates through the list of decals and if the vertex is within the radius of the decal, it gets displaced. Most mesh texture coordinates are not suitable for displacement mapping. Displacement maps needs a continuous, uniform mapping, otherwise cracks and aliasing artifacts can appear. To ensure a suitable mapping, the domain shader procedurally generates the texture coordinates by projecting the vertex position onto a local decal coordinate space which is stored using three vectors and a position in a constant buffer list.

Pixel Shader

The pixel shader stage is passed a 3D texture coordinate from the domain shader. The x,y values of the texture coordinate are the same as the displacement map texture coordinates generated in the domain shader as explained above. The z coordinate is used to determine if the pixel being lit is part of the decal or not. If the z coordinate = 1, then the pixel shader will use the normal map associated with the decal. Otherwise, the pixel shader will light the pixel as if it would normally, without decal tessellation.

GUI

The GUI allows the user to select between 4 different models, and supply their own model and texture ("user.sdkmesh", "diffuse.dds"). The GUI also allows the user to select between three different decals. In the center of the screen is a small crosshair which indicates where the decal is to be placed. Pressing "space" will add a new decal to the model.

Limitations

The technique works well on a wide variety of models. Some care is needed to make sure the displacement is not too large for the model; otherwise the displacement could penetrate the opposite side. Large displacements can also cause too much stretching of the textures. The algorithm currently doesn't handle overlapping decals, since this can cause objectionable discontinuities in the lighting. To avoid overlaps, the sample resizes the decal, so it fits on the mesh without overlapping any other decals.