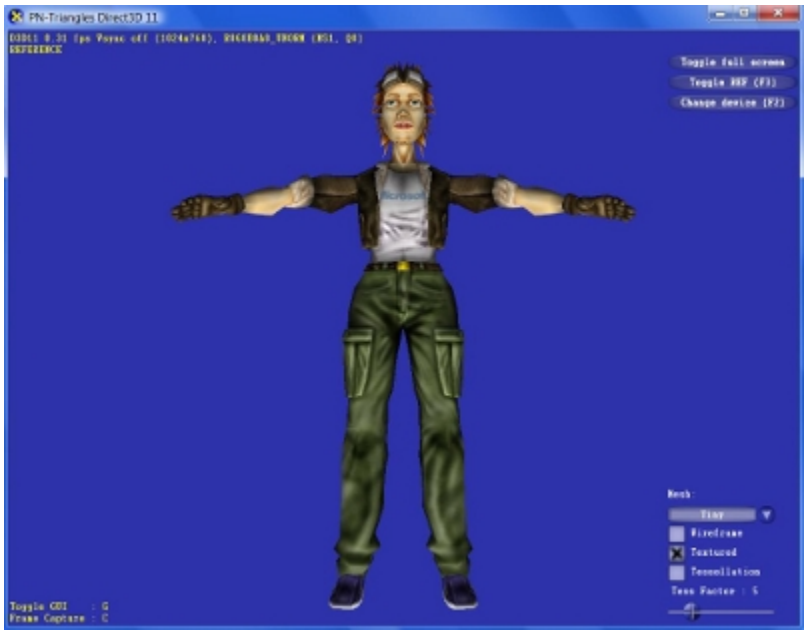
PNTriangles11 Sample

Collapse All

This sample, contributed by AMD, presents a technique for achieving smooth surfaces from the position and normal data of a low-density mesh. It utilizes Direct3D 11 APIs and hardware to make use of the new tessellation stages of the pipeline.

The algorithm employed here is taken directly from the "Curved PN Triangles" white paper written by Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell.



Path

Source	SDK root\Samples\C++\Direct3D11\PNTriangles11
Executable	SDK root\Samples\C++\Direct3D11\Bin\x86 or x64\PNTriangles11.exe

Hull Shader

The hull shader stage performs the usual pass-through steps that you'd expect, such as tessellation factors and input patch control points. In addition to this, the shader generates the seven cubic positional control points, and the three quadratic normal control points using the algorithm described in the above white paper. Also, this shader stage optionally handles various culling and adaptive tessellation techniques, all of which are contained in the AdaptiveTessellation.hlsl file.

1. 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.
2. View Frustum Culling: Patches with all control points outside the view frustum, based upon a supplied tolerance, are culled. This can also lead to very significant performance gains, especially for models close to the camera.
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.
4. Distance Adaptive: Providing a distance based range over which to scale tessellation factors by, is another way in which we can keep triangles sizes sensible. This is a simple technique, however it does not take into account the current monitor resolution.
5. Screen Resolution Adaptive: It may be sensible to globally scale tessellation factors based upon the current screen resolution, especially if distance based adaptive tessellation is being used.

6. Orientation Adaptive: Patch edges near to the silhouette of a model, based upon a supplied tolerance, can be given increased levels of tessellation. This greatly helps to improve the distribution of triangles towards a models silhouette, and therefore makes the most of the extra triangles.

Fixed-Function Tessellator

The fixed-function hardware 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 positional and normal control points to weight the barycentric coordinates of the generated geometry, using the algorithm described in the above white paper.

GUI

The GUI allows the user to select between three different models, and supply their own model and texture ("user.sdkmesh", "diffuse.dds").

Limitations

This technique works extremely well on models that were authored at a low density, but with normals that define the high-density curvature. Models with normals that do not follow the intended curvature will not generate the correct control points, and therefore the resulting tessellated mesh will not be correct.