SubD11

<https://github.com/walbourn/directx-sdk-samples>

This is the DirectX SDK's Direct3D 11 sample updated to use Visual Studio 2012 and the Windows SDK 8.0 without any dependencies on legacy DirectX SDK content. This sample is a Win32 desktop DirectX 11.0 application for Windows 10, Windows 8.1, Windows 8, Windows 7, and Windows Vista Service Pack 2 with the DirectX 11.0 runtime.

**This is based on the legacy DirectX SDK (June 2010) Win32 desktop sample. This is not intended for use with Windows Store apps, Windows RT, or universal Windows apps.**

*This sample requires Feature Level 11.0 or better.*

# Description



The SubD11 sample implements the algorithm described in the paper "[Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches](http://research.microsoft.com/~cloop/msrtr-2007-44.pdf)" by Charles Loop and Scott Schaefer. This sample is similar to the DirectX SDK **SubD10** sample, except that it has been enhanced to take advantage of three new Direct3D 11 pipeline stages: the hull shader, the tessellator, and the domain shader.

## Sample Performance and Architecture Notes

The Catmull-Clark content flows through the graphics pipeline in the following steps:

1. The input to the pipeline is a Catmull-Clark control mesh, which consists of quads and adjacency information. The control mesh is stored in a specially constructed SDKMESH content file. The adjacency information is computed offline using the Samples Content Exporter.
2. The vertex shader is used to deform the control mesh using skeletal animation.
3. The hull shader is used for basis conversion between the control mesh and a 4×4 Bezier patch.
4. The tessellator generates domain points and triangles that are based on the partitioning style and tessellation factor.
5. The domain shader is used to compute tangent patch data for the patch and then to evaluate all of the patches into a position and normal value for each domain point. Displacement mapping is also applied within the domain shader. Note that the tangent patch data is only used to compute the surface normal of the patch at the domain point. Texture tangent data from the content is merely interpolated in the domain shader and passed on to the pixel shader for use in normal mapping.
6. Finally, the pixel shader evaluates the normal map and produces a lit pixel.

Since this sample depends on the new Direct3D 11 pipeline stages, it can only operate by using the reference rasterizer at this time. As a result, it takes several seconds to render each frame. When Direct3D 11 hardware is available, we expect this sample to run thousands of times faster.

The architecture of the hull shader is currently designed for maximum readability, and it does not generate a compiled shader that is expected to be efficient on hardware. Specifically, the switch statements that are used to select different code paths for each hull shader output are inefficient. In future releases of this sample, the hull shader will be rewritten to avoid these inefficient constructs, at the expense of readability.

Currently the sample features tessellation using integer partitioning. Future releases will also demonstrate fractional and pow2 partitioning.

## Features

The following settings can be changed at run time:

* The bump height can be adjusted, which demonstrates displacement mapping.
* The tessellation factor can be adjusted to increase or decrease the level of tessellation.
* A wireframe overlay can be enabled to show the triangles that are generated by the tessellator.
* Mesh animation can be enabled to demonstrate deformable subdivision surfaces.

# Dependencies

DXUT-based samples typically make use of runtime HLSL compilation. Build-time compilation is recommended for all production Direct3D applications, but for experimentation and samples development runtime HLSL compilation is preferred. Therefore, the D3DCompile\*.DLL must be available in the search path when these programs are executed.

* When using the Windows 8.x SDK and targeting Windows Vista or later, you can include the D3DCompile\_46 or D3DCompile\_47 DLL side-by-side with your application copying the file from the REDIST folder.

%ProgramFiles(x86)%\Windows kits\8.0\Redist\D3D\arm, x86 or x64

%ProgramFiles(x86)%\Windows kits\8.1\Redist\D3D\arm, x86 or x64

%ProgramFiles(x86)%\Windows kits\10\Redist\D3D\arm, x86 or x64

# More Information

[Direct3D 11 Tessellation](https://walbourn.github.io/direct3d-11-tessellation/)

[Where is the DirectX SDK (2015 Edition)?](https://walbourn.github.io/where-is-the-directx-sdk-2015-edition/)

[DXUT for Win32 Desktop Update](https://walbourn.github.io/dxut-for-win32-desktop-update/)

[Games for Windows and DirectX SDK blog](https://walbourn.github.io/)