BasicHLSL Sample

⊟ Collapse All

This sample shows a simple example of the high-level shader language (HLSL) using an effect interface.

Path

Source	*SDK root* \Samples\C++\Direct3D\BasicHLSL
Executable	*SDK root* \Samples\C++\Direct3D\Bin\ *x86 or x64* \BasicHLSL.exe

Sample Overview

This sample simply loads a mesh, creates an effect from a file, and then uses the effect to render the mesh. The effect that is used is a simple vertex shader that animates the vertices based on time.

How the Sample Works

First the sample loads the geometry with **D3DXLoadMeshFromX** and calls **D3DXComputeNormals** to create mesh normals if there aren't any. It then calls OptimizeInplace() to optimize the mesh for the vertex cache, then loads the textures using **D3DXCreateTextureFromFile**. Next, the sample calls **D3DXCreateEffectFromFile** to compile the effect text file to a **ID3DXEffect*** interface called m_pEffect.

Note that the D3DXCreateEffectFromFile takes flags which are needed to debug shaders with the Visual Studio .NET shader debugger. Debugging vertex shaders requires either a reference device or software vertex processing. Debugging a pixel shader requires a reference device.

The D3DXSHADER_FORCE_*_SOFTWARE_NOOPT flags set various source code debugging options. Setting these flags will cause slower rendering since the shaders will be unoptimized and forced into software. All that is needed to turn these flags on with the sample code is to uncomment the #define DEBUG_VS or #define DEBUG_PS line from near the top of the file. This will also force the device into software processing by adding some code to CMyD3DApplication::ConfirmDevices.

In CMyD3DApplication::Render, the sample sets the technique to use with a call to **ID3DXEffect::SetTechnique** and passes a D3DXHANDLE to the technique that was obtained upon initialization with a call to **ID3DXBaseEffect::GetTechniqueByName**. It is possible to pass a string name instead of a handle, but using a handle improves performance since this high frequency call does not have to spend time on a string compare.

The sample then sets variables used by the technique such as the World*View*Projection matrix and the time variable with various ID3DXEffect::Set* calls. To render, the sample calls **ID3DXEffect::Begin** which returns the number of passes required by the technique, then loops through each pass calling m_pEffect->Pass(iPass) and rendering the mesh with **ID3DXBaseMesh::DrawSubset**. When all the passes are done, it calls **ID3DXEffect::End**.