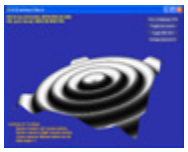
HLSLwithoutEffects Sample

Collapse All

HLSLwithoutEffects demonstrates using HLSL to write vertex shaders without using the D3DX effect interfaces. HLSL is a language that closely resembles C syntax and constructs. By writing shaders in HLSL instead of assembly language, developers can take advantage of not only the features and elements of the language they are already familiar with, but also the great optimization capabilities offered by the D3DX shader compiler.



Path

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

How HLSLwithoutEffects Works

The scene that this sample renders consists of a square grid of triangles lying on the XZ plane. In each frame, the vertices of this grid will move up or down along the Y direction based on a function of their distance to the origin and time. The vertices are also lit using another function of the distance and time. The time is incremented for each frame. Because the Y coordinate and color of the vertices are generated in each frame, they do not need to be stored. Therefore, the vertex declaration only contains a **D3DXVECTOR2** for the X and Z coordinates.

During initialization, the sample calls **D3DXCompileShaderFromFile** to read the shader function from a file and compile it into binary shader code. After this process, an **ID3DXBuffer** containing the shader code and a **ID3DXConstantTable** that allows the application to get and set the global variables of the shader are created. Then, the sample calls **IDirect3DDevice9::CreateVertexShader** with the shader code to obtain a **IDirect3DDevice9** object that can be passed to the device.

In FrameMove, the sample uses the **ID3DXConstantTable** interface to set the shader's global variables mViewProj and fTime in order to update the view + projection transformation and time parameter for the shader. At render time, the sample calls **IDirect3DDevice9::SetVertexShader** to enable vertex shader rendering on the device. When **IDirect3DDevice9::DrawIndexedPrimitive** is called after that, the vertex shader will be invoked once for every vertex processed.

In the sample, the vertex shader is written in a text file called HLSLwithoutEffects.vsh. In this file, there are two global variables and a shader function called Ripple. Ripple takes a float2 as input (for X and Z of the vertex) and outputs two float4 representing the screen-space position and vertex color. The Y coordinate is generated using this formula:

```
Y = 0.1 * sin( 15 * L * sin(T) );
```

L is the distance between the vertex and the origin before the Y adjustment, so it has the value of sqrt(X^2 + Z^2), because the vertices are lying on the XZ plane. T is the fTime global variable.

The color of the vertex will be some shade of gray based on this formula:

```
C = 0.5 - 0.5 * cos( 15 * L * sin(T) );
```

C will be the value used for all red, green, blue, and alpha and will range from 0 to 1, giving the vertex a color of black to white, respectively. The result looks like ripples with width changing back and forth between narrow and wide and with proper shading based on the slope.

General controls defined in the sample:

Key	Action
W/S	Move forward/backward

Q,E,A,D	Strafe
F2	Change device
Esc	Quit