Direct3D 10 Shader Model 4.0 Workshop

Collapse All

Summary

This workshop is based upon the Direct3D 10 Shader Model 4.0 Workshop presented at GDC 2006. The workshop guides the student through 7 exercises aimed at teaching the basics of Shader Model 4.0 HLSL. Students are expected to have a working knowledge of the Direct3D 10 graphics pipeline and HLSL. Exercise05 will require knowledge of C programming.

Source

(SDK root)\Samples\C++\Direct3D10\Tutorials\Direct3D10Workshop

General Exercise Guidelines

While the exercises do build upon each other, they are meant to be self-contained. With the exception of Exercise05, the student will be working with .fx files to complete each exercise. Comments are located throughout the code to guide the student. Comments are highlighted by a collection of ascii characters known as "Breakdancin Bob" and take the following form:

```
//--------------------------------------------------------------------
// o/__   <-- BreakdancinBob TODO: Todos are areas where you need
//  |  (\                  to change or implement code. This is where
//                         the actual exercise work will happen.
//--------------------------------------------------------------------

//--------------------------------------------------------------------
// o/__   <-- BreakdancinBob NOTE:  Notes are code snippets of
//  |  (\                  interest. No work needs to be done. Just
//                         look.
//--------------------------------------------------------------------

//--------------------------------------------------------------------
// o/__   <-- BreakdancinBob HINT:  Hints may be found near Todo
//  |  (\                  comments. Hints can help you if you are
//                         stuck with the exercise.
//--------------------------------------------------------------------
```

Solutions to the exercises are found in the folders with _Solved appended to the end.

Exercise00 - Debugging Practice

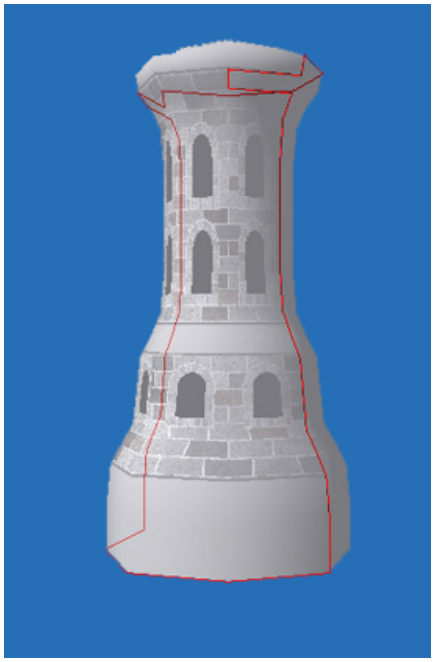
The goal of Exercise00 is to familiarize the user with the act of debugging shader problems. The solution will compile, but during run time, an error box will appear telling the user that the shader could not be compiled. Look in the Visual Studio output window for debug spew

from the shader compiler to determine the cause of the error.

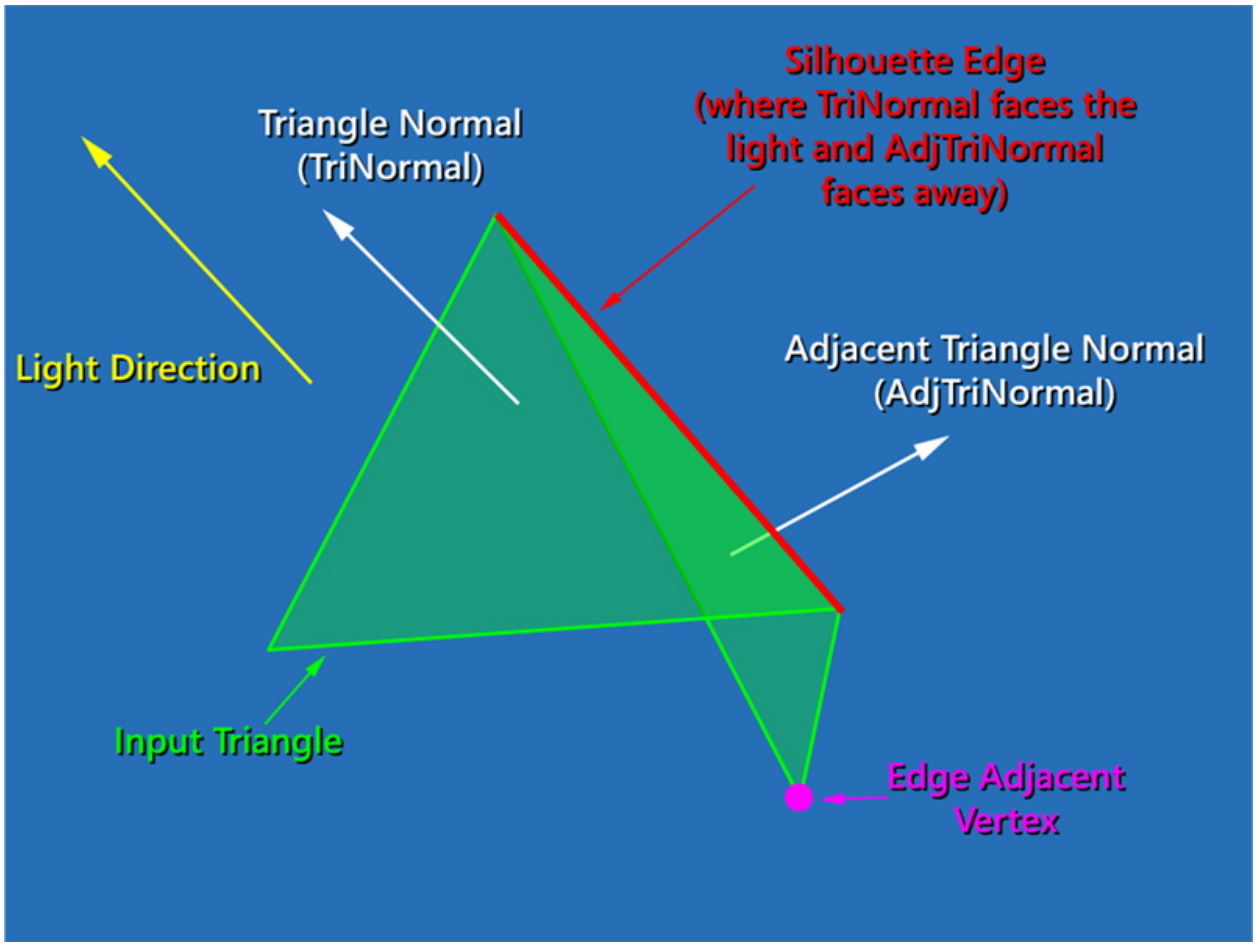
Exercise01 - Introduction to the Geometry Shader

The goal of Exercise01 is to teach basic usage of the Geometry Shader, a new shader type for Direct3d 10. The exercise teaches the student to use the Geometry Shader to append any elements needed by the pixel shader to the output vertex stream.

Exercise02 - Finding Silhouette Edges

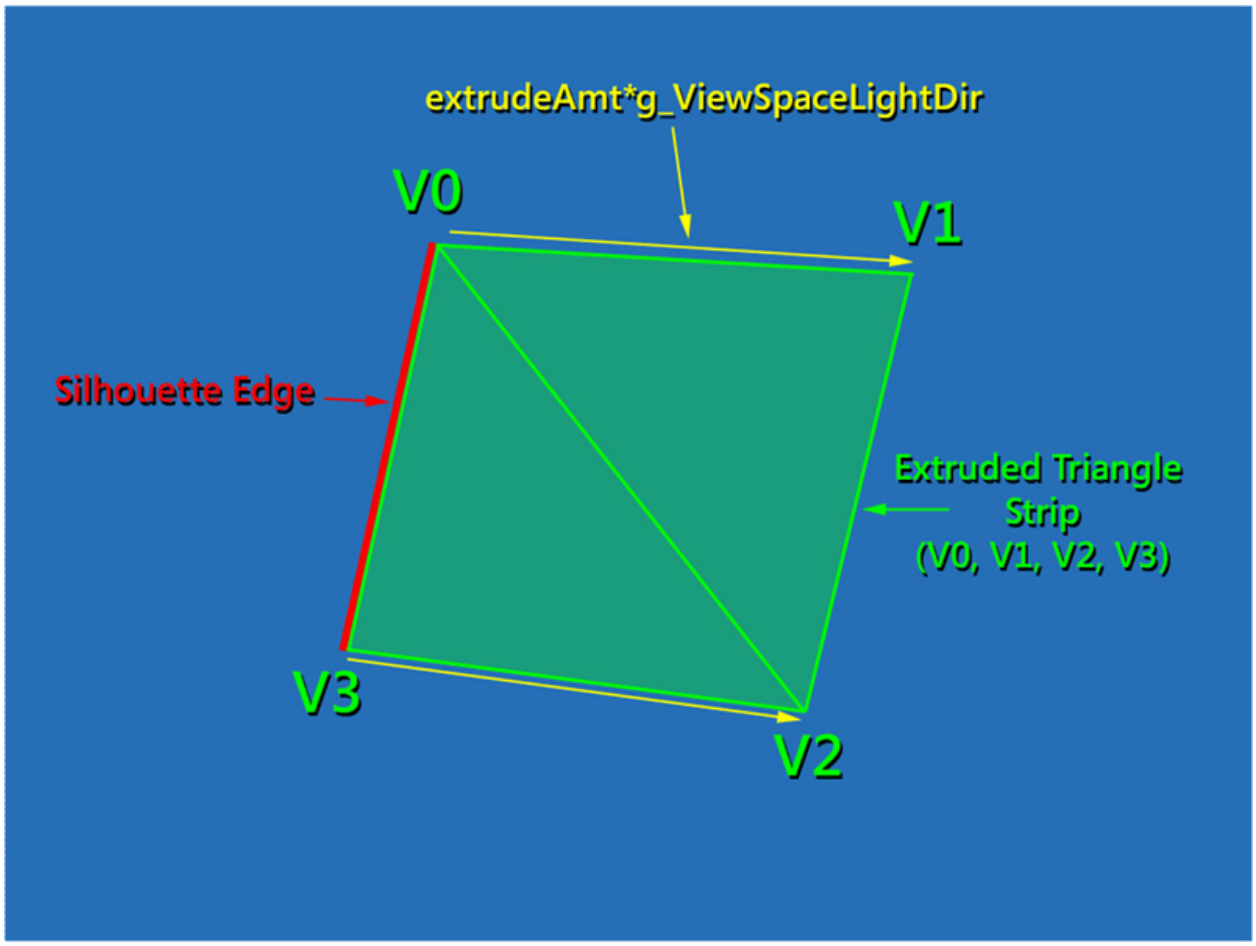


This exercise teaches the principle of finding silhouette edges. When the student opens the exercise, the shader calculates and displays a silhouette with respect to the viewer. Only edges with one face facing the viewer and one face facing away from the viewer are drawn. The goal of Exercise02 is to change this calculation to happen with respect to the light source. This is an essential step for doing stencil shadow volumes. The following diagram illustrates this more clearly.



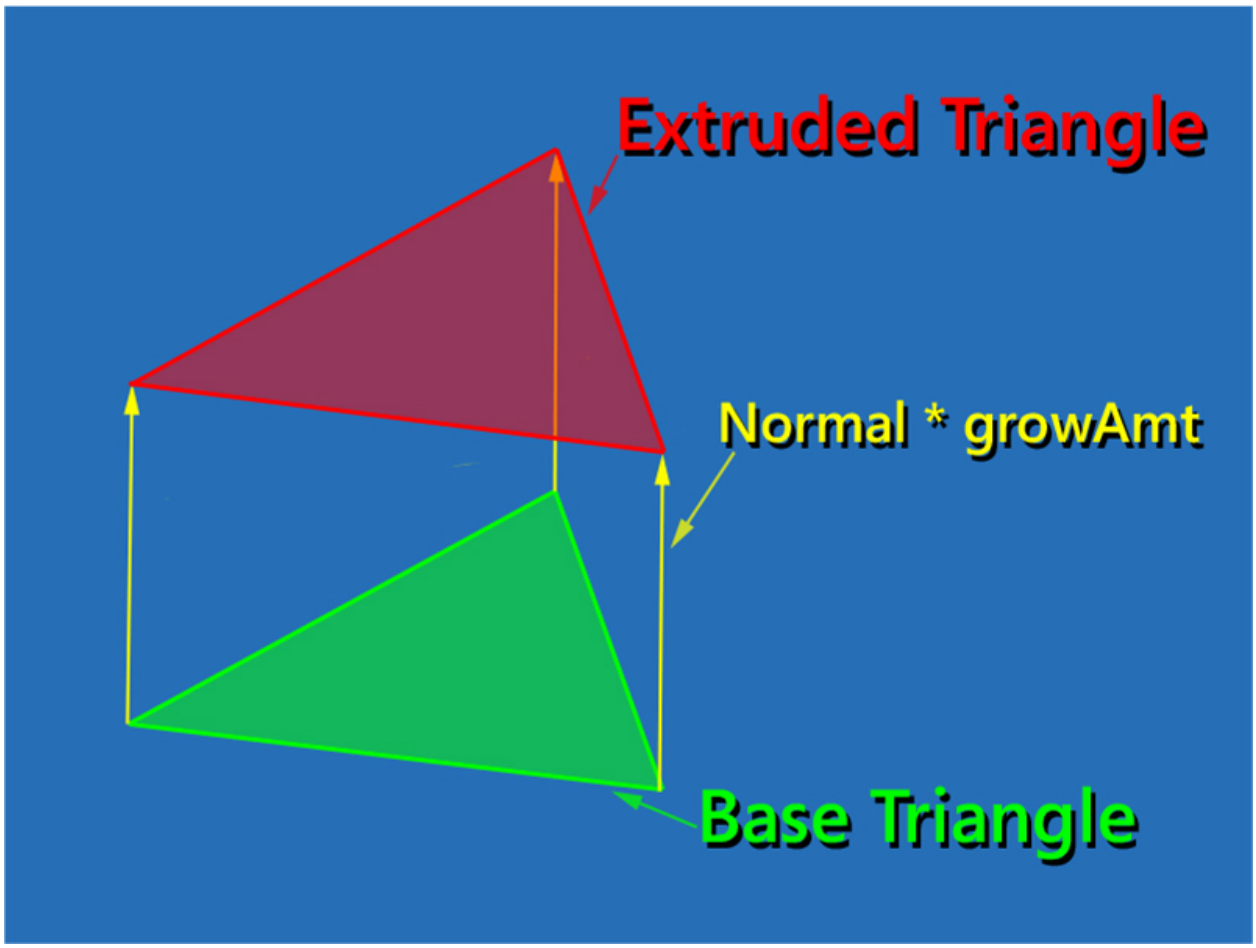
Exercise03 - Shadow Volumes

Exercise03 builds upon the silhouette edges found in Exercise02 to create extruded shadow volumes. While this exercise does not do capping and does not use the stencil buffer to actually do shadowing, it illustrates the second essential step for implementing stencil shadow volumes. The following diagram illustrates how to extrude a shadow volume from a selected edge.



Exercise04 - Creating New Geometry

The previous exercises showed off one of the main strengths of the Geometry Shader, operating on whole primitives with adjacency information. The remaining exercises illustrate two other strengths of the Geometry Shader. These are geometry amplification and destruction, and stream output. In Exercise04, the geometry shader is used to create new triangles based upon information contained in the vertex stream. At specific faces on the mesh, a new triangle is created and moved away from the original face in the direction of the normal.



Exercise05 - Feedback and Stream Out

This exercise demonstrates procedural geometry generation using geometry amplification through the Geometry Shader. The amplified geometry is then fed back through the Geometry Shader during subsequent passes to grow an entire tree from a small piece of seed geometry. This exercise deals only with the cpp file and requires knowledge of C to complete.

Exercise06

Exercise06 is a very simple exercise showing how to add some randomness and variation to tree created in Exercise05. A buffer of random numbers is generated and then sampled in the shader to control the growth of the tree.