

HLSLSymbols Sample

*\* This sample is compatible with the August 2016 Xbox One XDK.*

# Description

This sample demonstrates various options for generating and handling pdb symbols from HLSL shaders. Various useful features of the PIX tool require shader symbols, for instance edit-and-continue and shader debugging.

# Building the Sample

The solution contains two Visual Studio Projects

* **MyD3DCompiler:** A simple Win32 command line tool which compiles shaders using the D3DCompile API family.
* **HLSLSymbols:** An XDK project which runs on the Xbox, and which uses several copies of the same pixel shader, some of which are compiled using MyD3DCompiler.exe.

The Solution is set up to build both projects automatically, and to respect the dependency of HLSLSymbols on MyD3DCompiler.

# Using the sample

The sample has no runtime controls. The intended usage is to run the sample, observe the onscreen text, take a PIX capture, and view the pixel shaders for the various triangles within PIX.

## HLSLSymbols



|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Exit | View Button | Esc |

# Implementation notes

The code in the MyD3DCompiler project is the heart of the sample. This code demonstrates various methods for manipulating HLSL symbols at asset build time. The cases demonstrated in the sample are as follows:

* **EmbeddedPdb:** The shader was compiled using fxc with the /Zi option. Symbols are embedded in the shader binary. This is the simplest option, but it bloats runtime shader size.
* **ManualPdb:** The shader was compiled using MyD3DCompiler with the D3DCOMPILE\_DEBUG flag. Then, symbols were saved to disk, to a pdb filename chosen explicitly to match the name of the input HLSL file. Then, symbols were removed from the shader binary, and the path to the pdb was embedded in the shader binary. This approach is recommended to balance convenience with runtime shader size.
* **AutoPdb:** This case is the same as the previous case, except the pdb filename is chosen automatically by MyD3DCompiler to be identical to the shader semantic hash. This approach gets around certain difficulties which may arise when choosing pdb filenames. For instance, the input HLSL may come from a buffer in memory, with no natural associated filename. Or, a developer may compile the same HLSL file multiple times, with multiple arguments, leading to potential naming collisions.
* **AutoPdbNoPath:** This case is the same as the previous case, except that the path to the pdb is omitted from the shader binary to save additional memory. In this case, the developer must ensure that the pdb is located in the PIX symbol path, which can be controlled from the PIX Settings view.
* **StrippedPdb:** This case is an error case. The developer has generated symbols, but then stripped them without saving them to disk. There is no way for PIX to retrieve symbols for this shader.

# Known issues

None

# Update history

2014/07 – initial release

2018/08 – port to D3D12 and to new sample framework

# Privacy Statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).