

HlslCompile Sample

*This sample is compatible with the Microsoft Game Development Kit (October 2021)*

# Description

The sample compiles the same pixel shader in a number of different ways, to illustrate different options for PC-side asset build. The shader compiler is under active development, and we will update the sample as functionality changes.

The sample builds shaders using two different compiler interfaces:

* Dxc.exe – command line interface for the new shader compiler front end
* DxCompiler\_x[s].dll callable interface for the new shader compiler front end

There are separate copies of both the exe and the dll for Xbox One and for Xbox Series X|S. (The Xbox Series X|S copy of the dll is DxCompiler\_xs.dll.)

Shader symbols provide important information for PIX, in the same way that C++ symbols provide context for Visual Studio and other tools. The shader compiler interfaces support several options for storage of symbols:

* Embedded in the binary – this method is the simplest, but it is generally too costly in terms of runtime memory usage. Embedded symbols are deprecated for the new shader compiler front end.
* Stripped to a manually chosen file name – for example, the name may be chosen by the caller to be a recognizable variant of the source file name.
* Stripped to an automatically chosen file name – the name will be chosen based on a hash of the compiled shader. This method is recommended, since PIX can compute the same shader hash without a hint.

# Building the sample

If using an Xbox One devkit, set the active solution platform to Gaming.Xbox.XboxOne.x64.

If using an Xbox Series X|S devkit, set the active solution platform to Gaming.Xbox.Scarlett.x64.

*For more information, see* Running samples*, in the GDK documentation.*

# Using the sample

The sample is non-interactive. Each row in the screen image below contains a triangle. Each triangle is rendered using a copy of the same pixel shader, with each copy compiled in a different manner. The size of each pixel shader binary is listed in cyan (the numbers in the screenshot may be out of date). The remainder of each line’s text describes how the shader was compiled, and how symbols are stored.

To verify that symbols were generated correctly, we recommend that you take a PIX GPU capture of the sample and attempt to retrieve symbols within PIX for the pixel shader of each triangle. In some cases, PIX will automatically retrieve the proper symbols, while in other cases, manual action on the user’s part is required.



# Known issues

[none]

# Update history

Initial release April 2019

Updated for Microsoft GDK November 2019

# 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/).