

HLSLSymbols 样本

*\*此示例与2016年8月的Xbox One XDK兼容。*

# 描述

此样本演示了从 HLSL 着色器生成并处理 pdb 符号的各种选项。PIX 工具的各种有用功能需要着色器符号，例如编辑并继续以及着色器调试。

# 构建示例

解决方案包含两个 Visual Studio 项目

* **MyD3DCompiler：**一个简单的 Win32 命令行工具，使用 D3DCompile API 系列编译着色器。
* **HLSLSymbols：**一个在 Xbox 上运行的 XDK 项目，使用相同像素着色器的多个副本，其中一些使用 MyD3DCompiler.exe 进行编译。

解决方案设置为自动构建两个项目，并尊重 HLSLSymbols 对 MyD3DCompiler 的依赖性。

# 使用样本

样本没有运行时控件。预期用途为运行样本、观察屏幕文本、进行 PIX 捕获，以及查看 PIX 中各种三角形的像素着色器。

## HLSLSymbols



|  |  |  |
| --- | --- | --- |
| 操作 | 游戏手柄 | 键盘 |
| 退出 | 查看按键 | 退出 |

# 实施说明

MyD3DCompiler 项目中的代码为样本核心。此代码演示了在资产构建时操作 HLSL 符号的各种方法。样本中演示的案例如下：

* **EmbeddedPdb：**使用带有 /Zi 选项的 fxc 编译着色器。将符号嵌入着色器二进制文件中。这是最简单的选项，但会增加运行时着色器的尺寸。
* **ManualPdb：**使用带有 D3DCOMPILE\_DEBUG 标志的 MyD3DCompiler 编译着色器。然后，符号将保存至磁盘，显式选择的 pdb 文件名以匹配输入 HLSL 文件的名称。然后，从着色器二进制文件中删除符号，并将 pdb 的路径嵌入着色器二进制文件中。建议使用此方法来平衡便利性和运行时着色器的尺寸。
* **AutoPdb：**这种情况与前一种情况相同，除了 MyD3DCompiler 自动选择 pdb 文件名以与着色器语义哈希相同。这种方法可以解决选择 pdb 文件名时可能出现的某些困难。例如，输入 HLSL 可能来自内存中的缓冲区，没有自然关联的文件名。或者，开发人员可能会多次编译同一个 HLSL 文件，并带有多个参数，从而导致潜在的命名冲突。
* **AutoPdbNoPath：**这种情况与前一种情况相同，只是从着色器二进制文件中省略了 pdb 的路径以节省额外的内存。在这种情况下，开发人员必须确保 pdb 位于 PIX 符号路径中，可从 PIX 设置视图进行控制。
* **StrippedPdb：**这种情况为错误情况。开发人员已生成符号，但随后未将其保存到磁盘就将其删除。PIX 无法检索此着色器的符号。

# 已知的问题

无

# 更新历史记录

2014/07 - 首次发布

2018/08 - 移至 D3D12 和新的样本框架

# 隐私声明

编译和运行示例时，示例可执行文件的文件名将发送给Microsoft以帮助跟踪示例使用情况。要选择退出此数据收集，您可以删除Main.cpp中标记为“Sample Usage Telemetry”的代码块。

有关 Microsoft 隐私政策的更多信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/zh-cn/privacystatement/)。