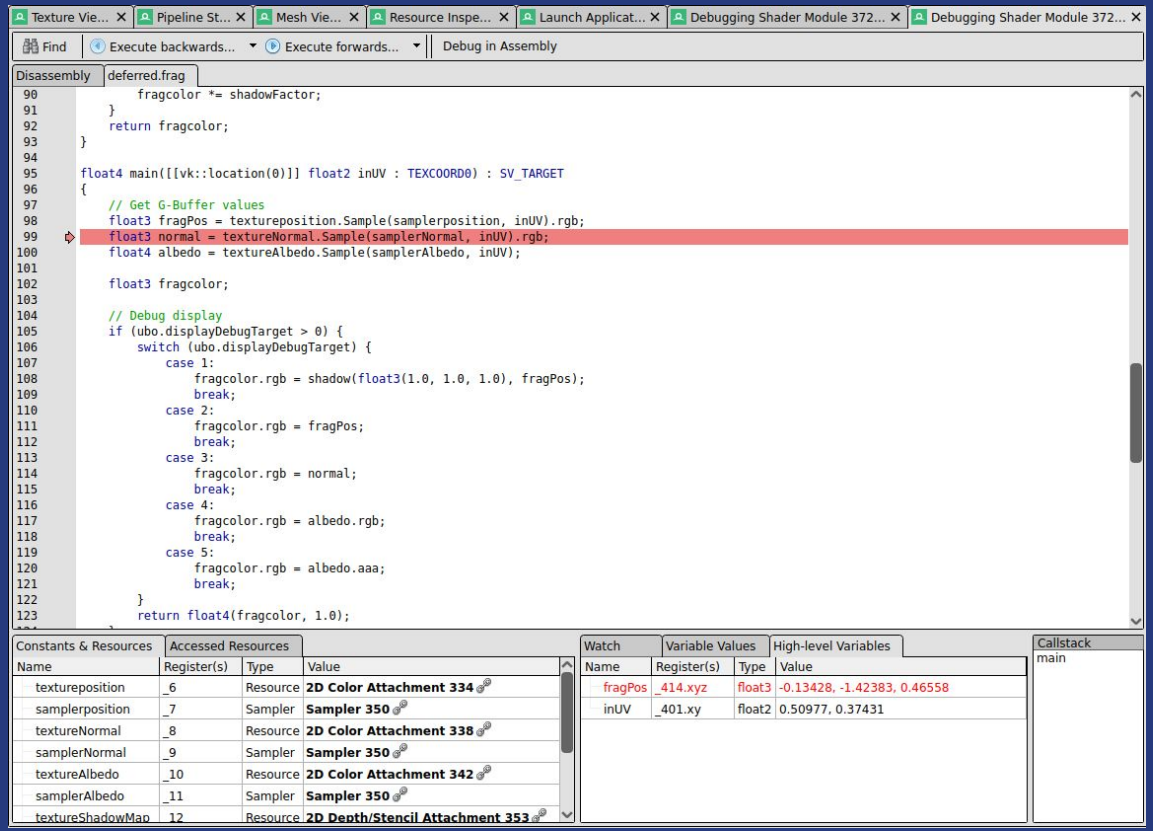
Source-level Shader Debugging in Vulkan with RenderDoc

Jeremy Hayes, LunarG, Inc.
Greg Fischer, LunarG, Inc.
October 27, 2022

KHRONOS GROUP / WEBINARS & MEETUPS

Vulkan®

Shader debugging in Vulkan is here!



Source-level Shader Debugging in Vulkan with RenderDoc

Jeremy Hayes, LunarG, Inc.

Senior Graphics Software Engineer with 20 years of graphics experience (3Dlabs, Intel, LunarG).

Greg Fischer, LunarG, Inc.

Senior Graphics Software Engineer with 38 years of compiler experience across numerous graphics and compute platforms.

Slides are available at:
https://www.lunarg.com/news-insights/white-papers/vulkanise-source-level-shader-debugging-in-vulkan-with-renderdoc/

LUNARG

Agenda

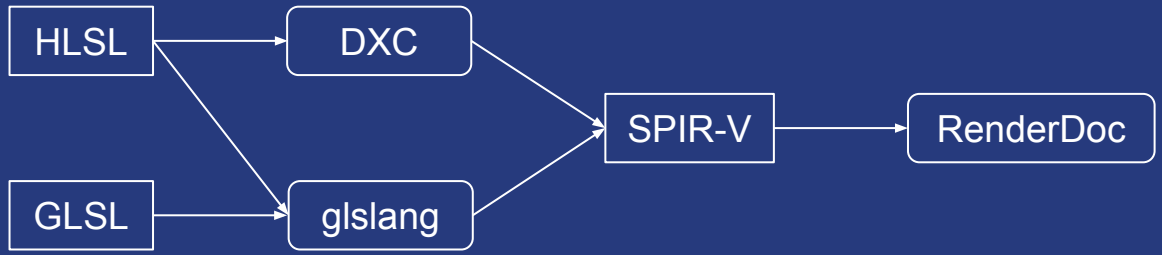
- What is NonSemantic.Shader.DebugInfo.100
- How to generate debug information
- Debug Instruction classifications
- Status of compiler toolchain support
- How to debug shaders in RenderDoc

NonSemantic.Shader.DebugInfo.100

- What?
  - Non-semantic SPIR-V extension (`OpExtension "SPV_KHR_non_semantic_info"`)
    - NonSemantic instructions can be safely ignored by drivers
  - Inspired and derived from DWARF debugging standard and OpenCL.DebugInfo.100
  - Authored by Baldur Karlsson from RenderDoc
- Why?
  - Information useful to debugging is lost during compilation and optimization
  - This extension preserves the lost information
  - Allows debugging of optimized shaders

NonSemantic.Shader.DebugInfo.100

- How?
  - `%result_id = OpExtInstImport “NonSemantic.Shader.DebugInfo.100”`
  - Annotates locations, scopes, types, variables, values, and procedures
  - DXC and glslang compiler support
  - SPIRV-Tools support
    - Optimized shaders can be debugged in addition to unoptimized shaders



Generating Debug Information (DXC)

```
\path\to\dxc.exe -spirv -fspv-target-env=vulkan1.3
      -T <target-profile> -E <entry-point>
      -fspv-extension=SPV_KHR_non_semantic_info
      -fspv-debug=vulkan-with-source
      <hlsl-src-file> -Fo <spirv-bin-file>
```

- `-fspv-extension=SPV_KHR_non_semantic_info` instructs the compiler to use the `SPV_KHR_non_semantic_info` extension which is required to use non semantic extended instruction sets. Not required for Vulkan 1.3.
- `-fspv-debug=vulkan-with-source` instructs the compiler to embed the source string in the `DebugSource` instruction. RenderDoc reads the high-level source from this instruction.

Generating Debug Information (glslang)

HLSL

```
\path\to\glslangValidator.exe -e main -gVS -D -o <spirv-bin-file> <hlsl-src-file>
```

GLSL

```
\path\to\glslangValidator.exe -e main -gVS -V -o <spirv-bin-file> <glsl-src-file>
```

- `-gVS` instructs the compiler to embed the source string in the `DebugSource` instruction (similar to the `-fspv-debug=vulkan-with-source` argument in DXC).
- `-D` tells the compiler that the source is HLSL.
- `-V` tells the compiler that the source is GLSL under Vulkan semantics.

Debug information will increase the size of your SPIR-V (~70% increase)

Debug instruction classifications

- Location
  - Source: DebugSource
  - Line / Column: DebugLine / DebugNoLine
- Types
  - int / floats / bool: DebugTypeBasic
  - vectors: DebugTypeVector
  - matrix: DebugTypeMatrix
  - arrays: DebugTypeArray
  - structure: DebugTypeComposite / DebugTypeMember
  - functions: DebugTypeFunction
- Variables
  - globals: DebugGlobalVariable
  - locals: DebugLocalVariable, DebugDeclare, DebugValue

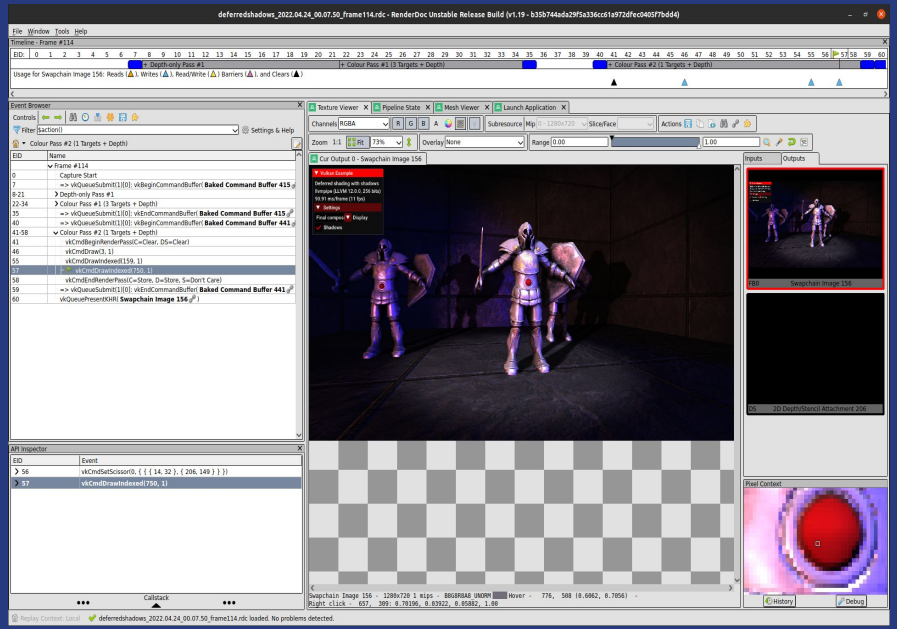
Debug instruction classifications

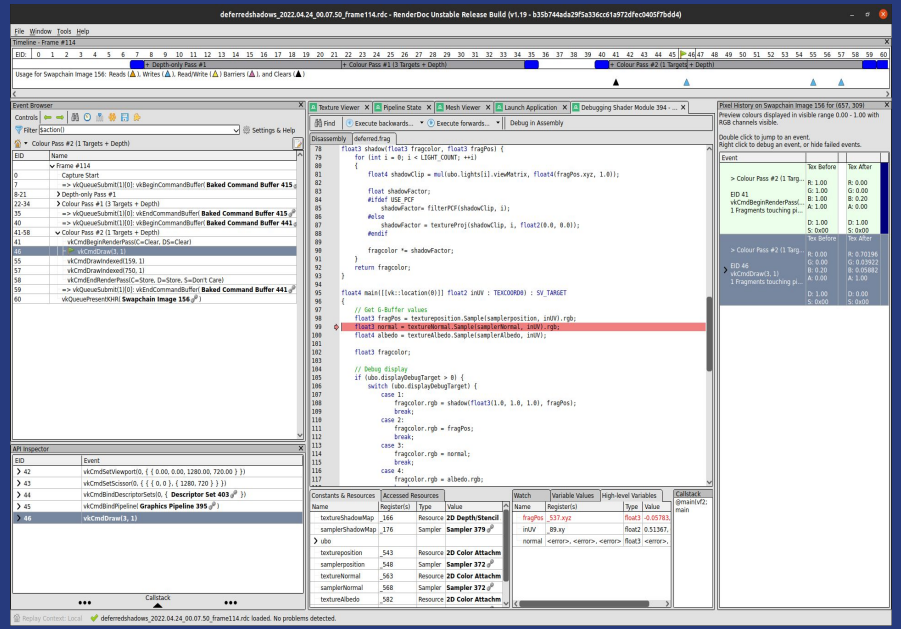
- Procedures
  - DebugTypeFunction, DebugFunction
- Scope
  - DebugScope / DebugNoScope
    - Compilation unit: DebugCompilationUnit
    - Function: DebugFunction
    - Structure: DebugTypeComposite
    - Everything else: DebugLexicalScope

Status

- Supported in the newest LunarG Vulkan SDK (1.3.231.1)
- Valve shaders (thanks to Dan Ginsburg)
- Sascha Willems samples
  - 79 samples
    - 281 HLSL/GLSL Shaders
      - 127 vertex shaders
      - 131 fragment shaders
      - 3 geometry shaders
      - 10 compute shaders
      - 5 tesselation control shaders
      - 5 tesselation evaluation shaders
- Ray tracing not supported
- Mesh shading not supported

Demo


deferredshadows_2022.04.24_00.07.50_frame114.rdc - RenderDoc Unstable Release Build (v1.19 - b35b744ada29f5a336cc61a972dfec0405f7bdd4)
File Window Tools Help
Timeline - Frame #114
Event Browser
Texture Viewer
Pipeline State
Mesh Viewer
Launch Application
API inspector
Callstack
Pixel Context
Replay Context: Local
deferredshadows_2022.04.24_00.07.50_frame114.rdc loaded. No problems detected.


deferredshadows_2022.04.24_00.07.50_frame114.rdc - RenderDoc Unstable Release Build (v1.19 - b35b744ada29f5a336cc61a972dfec0405f7bdd4)
File Window Tools Help
Timeline - Frame #114
Event Browser
Texture Viewer
Pipeline State
Mesh Viewer
Launch Application
Debugging Shader Module 394 - ...
Pixel History on Swapchain Image 156 for (657, 309)
API inspector
Callstack
Replay Context: Local
deferredshadows_2022.04.24_00.07.50_frame114.rdc loaded. No problems detected.

LUNARG

Future work

- Squash bugs
- DebugBuildIdentifier / DebugStoragePath
  - Store debug information in a separate file

Thanks!

- Dan Ginsburg (shaders)
- Baldur Karlsson (NonSemantic.shader.debuginfo.100, RenderDoc)
- Sascha Willems (shaders)
- Google (OpenCL.debuginfo.100, DXC, SPIRV-Tools)

Questions or presentation feedback?

Contact Jeremy Hayes: jeremy@lunarg.com

Report bugs or make feature requests here:

https://github.com/KhronosGroup/glslang

https://github.com/microsoft/DirectXShaderCompiler

For more information:

https://www.lunarg.com/news-insights/white-papers/source-level-shader-debugging-in-vulkan-with-renderdoc/

Thanks
Q&A?

Jeremy Hayes
LunarG, Inc.

Greg Fischer
LunarG, Inc.

KHRONOS GROUP / WEBINARS & MEETUPS

Vulkan®