Overview

- Shader Toolchain
- Glslang: Updates
- SPIRV-Tools: Updates
- DXC: Goal & Status
- DXC: Feature Coverage
- Using DXC
Shader Toolchain: Projects

SPIR-V
Shader Toolchain: Projects
Shader Toolchain: Projects

- HLSL
- GLSL
- DXC
- Glslang/Shaderc
- SPIR-V
- SPIRV-Reflect
- SPIRV-Tools
  - Assemble (spirv-as)
  - Disassemble (spirv-dis)
  - Optimize (spirv-opt)
  - Validate (spirv-val)
  - ...
- SPIRV-V/Metadata
Shader Toolchain: Projects

- HLSL
  - DXC
  - Glslang/Shaderc
  - SPIR-V
    - SPIRV-Tools
      - Assemble (spirv-as)
      - Disassemble (spirv-dis)
      - Optimize (spirv-opt)
      - Validate (spirv-val)
      - ...

- GLSL
  - SPIR-V
  - SPIRV-Reflect
  - SPIRV-V/Metadata

- MSL

...
Shader Toolchain: HLSL Compilation

- **Glslang**
  - Pioneered 6 months before Microsoft open sourced ...

- **DirectXShaderCompiler (DXC)**
  - Microsoft’s next-gen HLSL compiler open sourced January 2017
  - Google contributing SPIR-V CodeGen (a.k.a. Spiregg) since April 2017
    - Learnt a lot from Glslang’s experience
    - Cooperate with Glslang when features land in both

- **HLSL to SPIR-V compilation**
  - Recommend DXC as the forward path
# Shader Toolchain: Two Compilers

<table>
<thead>
<tr>
<th></th>
<th>Glslang/Shaderc</th>
<th>DXC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-level language</td>
<td>GLSL &amp; HLSL</td>
<td>HLSL</td>
</tr>
<tr>
<td>Intermediate language</td>
<td>SPIR-V</td>
<td>SPIR-V &amp; DXIL</td>
</tr>
<tr>
<td>Supported platform</td>
<td>Windows, Linux, macOS</td>
<td>Windows, Linux, macOS</td>
</tr>
<tr>
<td>Compiler library size</td>
<td>Small</td>
<td>Big (LLVM/Clang)</td>
</tr>
<tr>
<td>HLSL shader model</td>
<td>Up to 5.1, + 6.0 wave ops</td>
<td>Up to 6.2</td>
</tr>
<tr>
<td>HLSL validation</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Glslang: Updates

• Supported lots of new extensions
  - SPV_EXT_descriptor_indexing (for GLSL)
  - SPV_KHR_8bit_storage (for GLSL)
  - GL_EXT_shader_16bit_storage
  - SPV_GOOGLE_hlsl_functionality1 (for HLSL)
  - ...

• Supported linking GLSL compilation units
  - For a single stage

• Better integration with other tools
  - Adopted standard disassembly (spirv-dis)
  - Updated optimization (spirv-opt)
  - More information for debugging

• Incremental improvements & fixes
• Changed to semantic versioning
SPIRV-Tools: Updates

- Legalization works
  - Literal HLSL translation may generate illegal SPIR-V for Vulkan
  - Legalization: transformations to legalize SPIR-V
  - Sharing infrastructure with optimizations

- Stable optimization (spirv-opt)
  - Added loop optimizations
  - Supported many instruction combining cases
  - Better dead code elimination
  - Whole-array copy propagation
  - Greatly improved run time
  - ...

- Better validation (spirv-val)
  - Builtin variable
  - Resource layout
  - Better error message
  - ...
DXC: Goal

Make HLSL for Vulkan Shader Authoring Great
DXC: Current Status

- Stable now!
- Covered ~all native HLSL features
- Covered ~all Vulkan KHR/EXT extensions
- Better optimization support
- Better debugging support
- Supported non-Windows platforms
  - Linux, macOS
Coverage: HLSL Shader Models

- **Shader Model 5.1 and below ✔**
  - Excluding features without Vulkan equivalent
- **Shader Model 6.0 ✔**
  - Wave intrinsics, 64-bit integers
- **Shader Model 6.1 ✔**
  - SV_ViewID, SV_Barycentrics
- **Shader Model 6.2 (WIP)**
  - 16-bit types ✔
  - Denorm mode •
Coverage: HLSL Language Features

- **C-ish features** ✔
  - Math types
  - Various operations
  - Control flows
  - Functions

- **C++-ish features** ✔
  - Resource types and methods
  - Namespaces
  - OO (static) class members
  - OO (static) class methods
  - OO this pointer
  - OO inheritance

- **Common code patterns** ✔
  - Grouping resources in structs
  - Aliasing structured buffers
Coverage: Vulkan/SPIR-V

- **Supported Vulkan 1.0 & 1.1 ✔**
  - Push constant, subpass input, ...
  - Subgroup operations, ...

- **Supported Extensions ✔**
  - SPV_KHR_16bit_storage
  - SPV_KHR_device_group
  - SPV_KHR_multiview
  - SPV_KHR_post_depth_coverage
  - SPV_KHR_shader_draw_parameters
  - SPV_EXT_descriptor_indexing
  - SPV_EXT_fragment_fully_covered
  - SPV_EXT_shader_stencil_support
  - SPV_AMD_shader_explicit_vertex_parameter
  - SPV_GOOGLE_hlsl_functionality1
Using DXC: Downloads and Docs

- **Pre-built binaries**
  - Rolling release build from latest master branch: http://khr.io/dxcappveyorbuild

- **User manual**
  - How HLSL and Vulkan language features are translated: http://khr.io/hlsl2spirv

- **Compiler internals**
  - Detailed blog posts on translation difficulties and design choices of selected topics: antiagainst.github.io/categories/hlsl-for-vulkan/ (WIP)
Using DXC: Linux & macOS Support

- Windows specific techniques
  - COM, SAL, etc.
  - Introduced to solve technical issues on the Windows platform
  - No longer compilable/runnable on non-Windows platforms

- Implemented adapters for other platforms

- Master branch fully supported Linux and macOS now!
  - Travis CI running for all commits and pull requests
Using DXC: Resource Descriptor Assignment

- **If able to change source code:**
  - ```[[vk::binding(<binding#>, <set#>)]]
  - ```[[vk::counter_binding(<binding#)>)]] (for associated counter)

- **If unable to change source code:**
  - Using `:register(xX, spaceY)`
    - `x`: ignored, `x`: binding#, `y`: set#
  - With command-line shift
    - `-fvk-b-shift <shift-amount> <set#>
    - `-fvk-t-shift <shift-amount> <set#>
    - `-fvk-s-shift= <shift-amount> <set#>
    - `-fvk-u-shift= <shift-amount> <set#>
    - Shift for all sets: `<set#>` ← “all”
  - Or specifying 1:1 mapping
    - `-fvk-bind-register xX Y <binding#> <set#>`
Using DXC: Resource Memory Layout

- **Supported three sets of layout rules**
  - Vulkan, DirectX, OpenGL

<table>
<thead>
<tr>
<th>Rules</th>
<th>CL Option</th>
<th>Uniform Buffer</th>
<th>Storage Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulkan</td>
<td>(default)</td>
<td>“vector-relaxed” std140</td>
<td>“vector-relaxed” std430</td>
</tr>
<tr>
<td>DirectX</td>
<td>-fvk-use-dx-layout</td>
<td>fxc behavior</td>
<td>fxc behavior</td>
</tr>
<tr>
<td>OpenGL</td>
<td>-fvk-use-gl-layout</td>
<td>std140</td>
<td>std430</td>
</tr>
</tbody>
</table>

- **Supported**: packoffset()
  - Native HLSL feature, only for cbuffer

- **Supported**: \([\text{vk::offset}(X)]\)
  - Vulkan specific, for all structs
Using DXC: Optimization

- **-O**: running spirv-opt standard performance recipe
  - Running by default
  - All optimization levels are the same right now

- **-Oconfig**=: specifying your own passes
  - Same as spirv-opt -Oconfig=
  - E.g., -Oconfig=-O,--loop-unroll,-O
Using DXC: Debugging and Reflection

- **-Zi:** emitting debug information
  - Full path for main source file
  - Preprocessed entry point
  - Line information for certain instructions

- **-fspv-reflect:** emitting reflection information
  - Using the SPV_GOOGLE_hls1_functionality1 extension
  - Original HLSL semantic for input/output variables
  - Relation between main buffer and associated counter
DXC: What’s Coming

- More Vulkan extensions
- More debugging information
Credits

• Individuals (apologies if missing anyone)
  - DXC: Greg Roth, Hai Nguyen (Google)
  - spirv-opt: Diego Novillo (Google), Steven Perron (Google)
    Alan Baker (Google), Greg Fischer (LunarG)
  - Guidance: David Neto (Google), John Kessenich (Google)

• Companies
  - Microsoft
  - AMD, Intel, LunarG, Nvidia, Valve

• The community
Hiring

Thank you!

:)  

We're Hiring!

Contact Kevin Lusby (kevinlusby@google.com)!

Come visit us at the Google booth!