Vulkan GLSL Shaders

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License
The Shaders’ View of the Basic Computer Graphics Pipeline

• In general, you want to have a vertex and fragment shader as a minimum.

• A missing stage is OK. The output from one stage becomes the input of the next stage that is there.

• The last stage before the fragment shader feeds its output variables into the rasterizer. The interpolated values then go to the fragment shaders.

= Fixed Function

= Programmable

Oregon State University
Computer Graphics
Vulkan Shader Compiling

- You pre-compile your shaders with an external compiler
- Your shaders get turned into an intermediate form known as SPIR-V
- SPIR-V gets turned into fully-compiled code at runtime
- SPIR-V spec has been public for a couple of years—new shader languages are surely being developed
- OpenGL and OpenCL will be moving to SPIR-V as well

**Advantages:**

1. Software vendors don’t need to ship their shader source
2. Syntax errors appear during the SPIR-V step, not during runtime
3. Software can launch faster because half of the compilation has already taken place
4. This guarantees a common front-end syntax
5. This allows for other language front-ends
SPIR-V: 
Standard Portable Intermediate Representation for Vulkan

glslangValidator shaderFile -V [-H][-l<dir>][-S <stage>] -o shaderBinaryFile.spv

Shaderfile extensions:
.vert    Vertex
.tesc    Tessellation Control
.tese    Tessellation Evaluation
.geom    Geometry
.frag    Fragment
.comp    Compute
(Can be overridden by the –S option)

-V   Compile for Vulkan
-G   Compile for OpenGL
-I    Directory(ies) to look in for #includes
-S   Specify stage rather than get it from shaderfile extension
-c   Print out the maximum sizes of various properties

Windows: glslangValidator.exe
Linux:    setenv LD_LIBRARY_PATH /usr/local/common/gcc-6.3.0/lib64/
Vulkan: GLSL Differences from OpenGL

Detecting that a GLSL Shader is being used with Vulkan/SPIR-V:

- In the compiler, there is an automatic
  #define VULKAN 100

Vertex and Instance indices:

- Both are 0-based

  gl_VertexIndex
  gl_InstanceIndex

These are
  gl_VertexID
  gl_InstanceID
In OpenGL. The Vulkan names make more sense.

gl_FragColor:

- In OpenGL, it broadcasts to all color attachments
- In Vulkan, it just broadcasts to color attachment location #0
- Best idea: don’t use it – explicitly declare out variables to have specific location numbers
Shader combinations of separate texture data and samplers:
    uniform sampler s;
    uniform texture2D t;
    vec4 rgba = texture( sampler2D( t, s ), vST );

Descriptor Sets:
    layout( set=0, binding=0 ) . . . ;

Push Constants:
    layout( push_constant) . . . ;

Specialization Constants:
    layout( constant_id = 3 ) const int N = 5;

• Can only use basic operators, declarations, and constructors
• Only for scalars, but a vector can be constructed from specialization constants

Specialization Constants for Compute Shaders:
    layout( local_size_x_id = 8, local_size_y_id = 16 );

• gl_WorkGroupSize.z is still as it was
Vulkan: Shaders’ use of Layouts for Uniform Variables

```c
layout( std140, set = 0, binding = 0 ) uniform matBuf
{
    mat4 uModelMatrix;
    mat4 uViewMatrix;
    mat4 uProjectionMatrix;
    mat3 uNormalMatrix;
} Matrices;

// non-opaque must be in a uniform block:
layout( std140, set = 1, binding = 0 ) uniform lightBuf
{
    vec4 uLightPos;
} Light;

layout( set = 2, binding = 0 ) uniform sampler2D uTexUnit;
```

All opaque (non-sampler) uniform variables must be in block buffers.
You Can Run the SPIR-V Compiler on Windows from a Bash Shell

1. Click on the Microsoft Start icon

2. Type word bash
You Can Run the SPIR-V Compiler on Windows from a Bash Shell

Pick one:

- Can get to your personal folders
- Does not have make

- Cannot get to your personal folders
- Does have make
Running glslangValidator.exe

```
MINGW64:/y/Vulkan/Sample2017
ONID+mjb@pooh MINGW64 /y/Vulkan/Sample2017
$ !85
glslangValidator.exe -V sample-vert.vert -o sample-vert.spv
  sample-vert.vert
ONID+mjb@pooh MINGW64 /y/Vulkan/Sample2017
$ !86
glslangValidator.exe -V sample-frag.frag -o sample-frag.spv
  sample-frag.frag
ONID+mjb@pooh MINGW64 /y/Vulkan/Sample2017
$
```
You can also run SPIR-V from a Linux Shell

```sh
$ glslangValidator.exe -V sample-vert.vert -o sample-vert.spv
$ glslangValidator.exe -V sample-frag.frag -o sample-frag.spv
```
You can also run SPIR-V from a Linux Shell

```
glslangValidator.exe -V sample-vert.vert -o sample-vert.spv
```

Compile for Vulkan ("-G" is compile for OpenGL)

The input file. The compiler determines the shader type by the file extension:
- `.vert` Vertex shader
- `.tccs` Tessellation Control Shader
- `.tecs` Tessellation Evaluation Shader
- `.geom` Geometry shader
- `.frag` Fragment shader
- `.comp` Compute shader

Specify the output file
How do you know if SPIR-V compiled successfully?

Same as C/C++ -- the compiler gives you no nasty messages.

Also, if you care, legal .spv files have a magic number of 0x07230203

So, if you do an `od -x` on the .spv file, the magic number looks like this:

0203 0723 . . .
SPIR-V: More Information

SPIR-V Tools:
http://github.com/KhronosGroup/SPIRV-Tools
Installing bash on Windows

1. Open **Settings**.
2. Click on **Update & security**.
3. Click on **For Developers**.
4. Under "Use developer features", select the **Developer mode** option to setup the environment to install Bash.
5. On the message box, click **Yes** to turn on developer mode.
6. After the necessary components install, you'll need to restart your computer.
7. Once your computer reboots, open **Control Panel**.
8. Click on **Programs**.
9. Click on **Turn Windows features on or off**.
10. Check the **Windows Subsystem for Linux (beta)** option.
11. Click **OK**.
12. Once the components installed on your computer, click the **Restart now** button to complete the task.

After your computer restarts, you will notice that Bash will not appear in the "Recently added" list of apps, this is because Bash isn't actually installed yet. Now that you have setup the necessary components, use the following steps to complete the installation of Bash.

1. Open Start, do a search for **bash.exe**, and press **Enter**.
2. On the command prompt, type **y** and press **Enter** to download and install Bash from the Windows Store.
3. Then you'll need to create a default UNIX user account. This account doesn't have to be the same as your Windows account. Enter the username in the required field and press **Enter** (you can't use the username "admin").
4. Close the "bash.exe" command prompt

Now that you completed the installation and setup, you can open the Bash tool from the Start menu like you would with any other app.

https://www.windowscentral.com/how-install-bash-shell-command-line-windows-10