Compiling Shaders to SPIR-V for use in Vulkan

Vulkan uses SPIR-V for shader modules. Standard Portable Intermediate Representation (SPIR) is an open standard to allow parallel compute and graphics operations to be represented in a cross platform fashion. SPIR-V is the newest version, a binary intermediate representation, with native graphics and compute constructs.

The standard vertex, fragment, compute, geometry and tessellation shaders can be compiled into SPIR-V binary blobs using glslang, provided by the Khronos Group on github in source form. It can be found at https://github.com/KhronosGroup/glslang . Further information can be found on the Khronos reference compiler page https://www.khronos.org/opengles/sdk/tools/Reference-Compiler/ .

Glslang provides a validator binary (glslangValidator) that can compile shaders of various type into SPIR-V which can be provided when creating vulkan pipeline objects. We recommend rebuilding this binary from the glslang source code to obtain the latest version which supports new Vulkan constructs.

```
Usage: glslangValidator [option]... [file]...

Where: each 'file' ends in .<stage>, where <stage> is one of
    .conf   to provide an optional config file that replaces the default
configuration
            (see -c option below for generating a template)
    .vert   for a vertex shader
    .tesc   for a tessellation control shader
    .tese   for a tessellation evaluation shader
    .geom   for a geometry shader
    .frag   for a fragment shader
    .comp   for a compute shader

Compilation warnings and errors will be printed to stdout.

To get other information, use one of the following options:
Each option must be specified separately.
  -V          create SPIR-V binary, under Vulkan semantics; turns on -l;
              default file name is <stage>.spv (-o overrides this)
              (unless -o is specified, which overrides the default file
name)
  -G          create SPIR-V binary, under OpenGL semantics; turns on -l;
              default file name is <stage>.spv (-o overrides this)
  -H          print human readable form of SPIR-V; turns on -V
  -E          print pre-processed GLSL; cannot be used with -l;
              errors will appear on stderr.
  -c          configuration dump;
```

```
              creates the default configuration file (redirect to a .conf
file)
  -d          default to desktop (#version 110) when there is no shader
#version
              (default is ES version 100)
  -h          print this usage message
  -i          intermediate tree (glslang AST) is printed out
  -l          link all input files together to form a single module
  -m          memory leak mode
  -o  <file>  save binary into <file>, requires a binary option (e.g., -V)
  -q          dump reflection query database
  -r          relaxed semantic error-checking mode
  -s          silent mode
  -t          multi-threaded mode
  -v          print version strings
  -w          suppress warnings (except as required by #extension : warn)
```

To compile a vertex shader `simple.vert` to a SPIR-V binary contained as a file in `simple_vert.spv` one would invoke `glslangValidator` as so:

```
> glslangValidator –V –x simple.vert –o simple_vert.spv
```

simple_vert.spv can be loaded in its entirety and provided as a single buffer to the Vulkan APIs. There is no file header to parse, the whole binary blob is provided to `vkCreateShaderModule`.

As an alternative, the VkSampleFramework provides functions to load shader modules from memory and from files within the Assets folder of android projects.

```
VkShaderModule VkSampleFramework::CreateShaderModuleFromAsset(const char*
asset)
{
    // Get the file from the Android asset manager
    AAsset* file = AAssetManager_open(GetAssetManager(), asset,
                                      AASSET_MODE_BUFFER);
    assert(file);

    const uint32_t file_size = AAsset_getLength(file);
    const uint32_t* file_buffer = (const uint32_t*)AAsset_getBuffer(file);
    assert(file_buffer);

    VkShaderModule module = CreateShaderModule(file_buffer, file_size);
    AAsset_close(file);

    return module;
}

VkShaderModule VkSampleFramework::CreateShaderModule(const uint32_t* code,
uint32_t size)
{
    VkShaderModule module;
    VkResult  err;
```

```
    // Creating a shader is very simple once it's in memory as compiled
    // SPIR-V.
    VkShaderModuleCreateInfo moduleCreateInfo = {};
    moduleCreateInfo.sType = VK_STRUCTURE_TYPE_SHADER_MODULE_CREATE_INFO;
    moduleCreateInfo.pNext = nullptr;
    moduleCreateInfo.codeSize = size;
    moduleCreateInfo.pCode = code;
    moduleCreateInfo.flags = 0;
    err = vkCreateShaderModule(m_device, &moduleCreateInfo, nullptr,
                               &module);
    VK_CHECK(!err);

    return module;
}
```

Android Studio Gradle Scripts

The Android Studio projects associated with the samples are supplied with build tasks written into the gradle scripts. There is a Clean task which deletes all .spv SPIR-V binaries from the assets folder of the project, and a Build task which incrementally builds modified shader files in the jni/shaders folder. For builds to succeed, glslangValidator needs to be on the system path.