ARB_gl_spirv implementation on Mesa: status update

igalia

Introduction

Topics

- What we are talking about: some specs
- Previous presentation, status then
- Evolution
- Testing
- Current status, future

GLSL

- Open**GL S**hading **L**anguage.
- C-Like high-level language shading language
- ARB_vertex/fragment_program too low level. Several vendor alternatives.
- Introduced as an OpenGL 1.4 extension on 2003, part of the OpenGL 2.0 core on 2002
- The original shader is included on your program.

SPIR-V

- Introduced as SPIR in 2011
  - **S**tandard **P**ortable **I**ntermediate **R**epresentation
  - Initially for OpenCL
  - Binary format based on LLVM IR
- SPIR-V announced in 2015
  - Part of OpenCL 2.1 core and Vulkan core
  - Not based on LLVM IR anymore

OpenGL and Vulkan: common ecosystem

- Some applications are porting from one to the other, or want to support both
- Some interoperability are desired
- But one use GLSL, other SPIR-V as shading language
- Fist step: GL_KHR_vulkan_glsl

GL_KHR_vulkan_glsl

- Modifies GLSL to be used for Vulkan
  - Announced on Dec 2015
- GLSL is compiled down to SPIR-V, which the Vulkan API consumes
  - Not a driver extension, but frontend one.
- Removes several features, add others
  - “Just GLSL” is not suitable for Vulkan consumption

GL_ARB_gl_spirv

- Allow SPIR-V modules to be loaded on OpenGL
- Modifies GLSL to be a source for creating SPIR-V modules for OpenGL consumption
  - Based on the previous, but not exactly the same
  - Also adds/removes features
- Driver + frontend extension
- “Just GLSL” (through SPIR-V) not suitable for OpenGL consumption either.

Previous presentation: FOSDEM 2018

FOSDEM status summary (I)

- Mesa already has a way to handle SPIR-V shaders, thanks to Vulkan
- SPIR-V to NIR pass
- NIR is one of the Intermediate Representations used at Mesa (IR, NIR, TGSI, etc)
- Problem: shader linking was based on IR

FOSDEM status summary (II)

- First workaround: partial conversion to IR
  - “Shadow variables”
  - Good for bootstrap, discarded soon
- First proof-of-concept was implemented, good enough to pass all the CTS tests
- Cleaned up version of this implementation was used on a downstream version of the Intel driver to pass the conformance
- Challenge: was decided a new linker for ARB_gl_spirv, based on NIR

FOSDEM status summary (III)

- Main reason: ARB_gl_spirv linking needs to work without names

- But current GLSL, so it’s linker, is based on names

```
/* This hash table will track all of the uniform blocks that have been
 * encountered.  Since blocks with the same block-name must be the same,
 * the hash is organized by block-name.
 */
```

Evolution: FAQ mode

The NIR linker is needing a lot of time, do you regret starting it?

- No.

The NIR linker is needing a lot of time, do you regret starting it?

- No.
- During this time even more difference between the GLSL linker and what an ARB_gl_spirv linker needs were found
- Specifically, even although the general rule is trying to behave as similar as possible as OpenGL, some things are done “in the Vulkan way”

Why not just extend the shadow variables idea? NIR to IR pass?

- We would still have the problem of not having a name
- GLSL linking rules, specially validations, are mostly based on names

- Bold statement: even if we had a SPIR-V to IR pass, separate code/linking for ARB_gl_spirv would make sense.

Why not just make-up a name and go with the IR linking?

- How do you make up the name?
- Sometimes you would need the binding, other the location, offset, etc
  - You would mean the semantics of what you are using for linking at a given moment
- The spec itself encourages to “link the vulkan-way”: built-in decorations, locations, etc
- “Make-up a name” is not only hacky, sounds hacky too

Why a NIR linker so tailored to ARB_gl_spirv, why not for both?

- Again, GLSL linking are mostly based on the ubo/uniforms/etc names
- Initial stages tried to design a solution for both, and also share as much code as possible for existing linker.
- Outcome: block
- We still think that the differences on the spec justifies a different linker
- Still trying to share as much as possible

But why it is taking so long? Isn't SPIR-V more straight-forward?

- In the end, it is needed to feed up the same internal structures that with a GLSL linker
- Uniforms need to be enumerated from their block, among other things, to know how many we have
- Ditto for info related to them, like ubo/ssbo buffer size

Wait there! Number of uniforms? Buffer sizes? I thought that SPIRV din’t require introspection?

- That’s not inherent to SPIR-V
- It is just how Vulkan uses it, or requires from it

Wait, I have read the spec, I found several “no reflection required”.

- Yes, issue (8) says “First start without reflection”
- And then issue (12) says “No reflection required”

Wait, I have read the spec, I found several “no reflection required”.

- Yes, issue (8) says “First start without reflection”
- And then issue (12) says “No reflection required”
- But on those, it refers to “name reflection APIs”, not any introspection at all (yes can be confusing)

Wait, I have read the spec, I found several “no reflection required”.

- Yes, issue (8) says “First start without reflection”
- And then issue (12) says “No reflection required”
- But on those, it refers to “name reflection APIs”, not any introspection at all (yes can be confusing)
- From issue (19):

  “C) Allow as much as possible to work "naturally". You can query for the number of active resources, and for details about them. Anything that doesn't query by name will work as expected."

So this linker needs to re-implement all what GLSL does?

- No. The scope is really smaller.
- Mesa GLSL linker need to support different versions of GLSL, with several rules/features added for years.
- We can assume that the SPIR-V module has been validated. Undefined behaviour allowed for wrong SPIR-V
- Several features were removed, or become simpler (example: no specific memory layouts)

Testing

Testing

- Passing CTS is not enough to be considered production ready
- Were good to test the ARB_gl_spirv specifics, shallow for testing the full range of features allowed
- We use extensively Piglit

Piglit

- Open-source test suite for OpenGL implementations
- Heavily used by Mesa developers
- Several of their tests uses shader_runner
  - Individual test uses a text format
  - This includes the shader source, data values and expected values

shader_test example

```
[require]
GL >= 3.3
GLSL >= 4.50

[vertex shader passthrough]

[fragment shader]
#version 450

layout(location = 0) uniform vec4 color;

layout(location = 0) out vec4 outcolor;

void main() {
    outcolor = color;
}

[test]
clear color 1.0 0.0 0.0 0.0
clear

#color
uniform vec4 0 0.0 1.0 0.0 1.0

verify program_query GL_ACTIVE_UNIFORMS 1

draw rect -1 -1 2 2
probe all rgba 0.0 1.0 0.0 1.0
```

Piglit (barebone tests)

- Nicolai added support for ARB_gl_spirv on shader_runner
- Also added a script that parses the shader_test, and call glslang to create the SPIR-V source
- This allowed to add tests for the extension
- We are adding them as we implement features, but even more is needed

Piglit (borrowed tests)

- The script tries to be able to reuse tests written originally for other specs
- In some cases, it does some fixing
  - Like setting explicit location/binding etc
- Also automatic filtering of tests that are not compatible with ARB_gl_spirv
- We got a lot of tests this way. Allowed a test-based implementation since basically day 0.

Piglit (conversion numbers)

- From the 2174 hand-written tests:
  - 926 skipped, 24 fail
  - 1224 tests successfully generated
- From the 32386 generated tests:
  - 1775 skipped
  - 30611 tests successfully generated
- We got 31835 tests with SPIR-V shaders just borrowing tests from other specs

Piglit (pass rate)

- With our development branch:
  - [34561] skip: 5928, pass:28508, fail:114, crash: 11
- Most crashes come from lack of multi-dimensional ubo/ssbo
- Most failures come from missing more validation checks
  - Several of them are likely to be skipped

Current status

What we already have on master?

- ~80 Mesa patches, and ~30 patches are already on their respective masters
- Thanks to all reviewers, specially Timothy Arceri and Jason Ekstrand
- Those cover
  - Uniforms
  - Atomic counters
  - Transform feedback and geometry streams
  - 64-bit and 64-bit vertex attribute
  - etc

What’s happening now?

- Just this week we send a 26 patches series with the ubo/ssbo support, and some extras
  - Minus multi-dimensional arrays
- Those extras are mostly in order to get all the ARB_gl_spirv CTS tests passing

What’s next? (mandatory)

- Improve interface query support
  - A lot of work done on testing
- Multidimensional ubo/ssbo
- Validation
- Enable extension

What’s next? (long term)

- Shader cache support
  - Right now we disable on the ARB_gl_spirv path
- Bring names if available
- Test with real applications

Personal thoughts

Two worlds

- The best of two worlds
  - SPIR-V on OpenGL!
  - But you can still use as much OpenGL API as possible!
-

Two worlds

- The best of two worlds
  - SPIR-V on OpenGL!
  - But you can still use as much OpenGL API as possible!
- The worst of two worlds:
  - OpenGL driver need to do more linking with SPIR-V that the Vulkan driver
  - Just to provide a subset of the OpenGL introspection

Shouldn’t it better to chose one?

- Depends on the main use case
- Transitional helper while people port to Vulkan? Or a way to help to reduce the cost of having two renderers?
- On the former, “OpenGL way of things” make more sense
- On the latter, “Vulkan way of things” make more sense

GLSL confusion

- Two different specs defining the GLSL valid for SPIR-V generation
- Too many feature switching/tweaking
- Although it is most a frontend problem, in the end the driver needs to support a set of features
- Sometimes not so clear what is needed to be supported
- Likely a issue for applications too

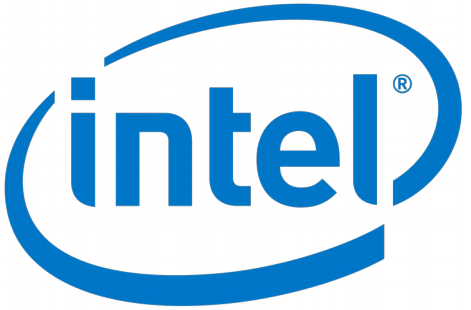
Closing

Who is working on this?

- Started by Nicolai Hänhle

- Continued by Igalia

- Supported by Intel



Questions?


igalia