Easy GL and Vulkan tests with shader_runner and Amber

Arcady Goldmints-Orlov, Igalia

| agoldmints@igalia.com | www.igalia.com

Motivation

Why should you care?

- We test graphics drivers and the compiler with piglit and the Khronos CTS.
- These are big, complicated programs full of C code.
- But they contain within them some neat scripting tools: shader_runner in piglit and Amber included with the CTS.
- These tools are useful both for developing new tests and for writing code for debugging.

Programming is hard, let’s go scripting!

- Rather than writing a whole program in C or C++, just use an existing framework.
- For scripting OpenGL tests, shader_runner is part of piglit.
- For scripting Vulkan, Amber is used in the Vulkan CTS but is actually a separate project and can be used standalone.

Running shader_runner

- When you build piglit, it can be found in bin/shader_runner
- Run it with `bin/shader_runner my_test.shader_test`
- The default mode is to draw into a window and keep that window when the test script completes.
- Useful options:
  - -auto – exit when the test is complete and print a pass or fail status
  - -fbo – do the drawing off-screen.

shader_runner syntax


```
[require]
GL >= 4.0
GLSL >= 4.10

[vertex shader passthrough]

[fragment shader]
#version 410
precision mediump float;

layout(location = 0) out vec4 color;

void main()
{
   float res = dFdx(gl_FragCoord.x);
   color = vec4(res/4, 0.502, 0, 1);
}

[test]
clear color 0.0 0.0 0.0 0.0
clear
draw rect -1 -1 2 2
probe all rgba 0.25 0.502 0.0 1.0
```

Other shader_runner commands

- shader_runner has a plethora of other useful commands, supporting most of OpenGL:

  compute – run a compute shader

  probe ssbo – check values in an SSBO

  ssbo subdata – set values in an SSBO

  probe rect rgba – verify the values in a rectangle in the output framebuffer

- Unfortunately these are not well documented, you have to read shader_runner.c to learn all the options.

- Lots of example *.shader_test files for inspiration.

Amber

- Created by Google, inspired by shader_runner. Can be found at https://github.com/google/amber

- Also integrated into the Vulkan CTS

- If you have the VK-GL-CTS, it can be found in external/amber

- Supports multiple syntaxes, its own AmberScript as well as a syntax similar to shader_runner

Amberscript Example


```
SHADER vertex vert_shader PASSTHROUGH
SHADER fragment frag_shader GLSL
#version 430
layout(location = 0) out vec4 color_out;
void main() {
  color_out = vec4(1.0, 0.0, 0.0, 1.0);
}
END

BUFFER framebuffer FORMAT B8G8R8A8_UNORM

PIPELINE graphics my_pipeline
  ATTACH vert_shader
  ATTACH frag_shader
  BIND BUFFER framebuffer AS color LOCATION 0
END

RUN my_pipeline DRAW_RECT POS 0 0 SIZE 250 250
EXPECT framebuffer IDX 0 0 SIZE 250 250 EQ_RGBA 255 0 0 255
```

More AmberScript

- `PIPELINE compute`
- `BUFFER foo DATA_TYPE float SIZE 42 FILL 0.0`
- `EXPECT buffer1 RMSE buffer2 TOLERANCE 1.0`
- AmberScript is documented in the amber source

Advanced Amber tips

- Amber supports Vulkan validation layers
- But you can disable them with -d
- Amber lets you dump any buffer(s) to a png file with the -I and -i options. Handy for debugging, especially when using multiple render targets.
- Shaders can be in SPIR-V as well as GLSL, the latter get compiled with glslangValidator
- You can have multi-step tests, drawing to two buffers and comparing them with a shader for example.

Conclusion

- I have found these tools useful to write quick tests, especially when working on the compiler.
- I hope you will also find these tools useful if you didn’t know about them already.
- I hope you are inspired to improve them even further.

Questions?

Thank you. Any questions?