SwiftShader
Reference Implementation and Fallback
Alexis Hétu / July 31, 2019
Introduction

- Graphics driver for the CPU
- Accelerated by
  - Multi-core
  - SIMD vectors
  - Specialized instructions
Google all-in on Vulkan

- High performance is key
- Mandatory for Android Q phones, except in extreme low-end
- Stadia uses Vulkan on Linux
- Chrome is adding Vulkan support
- The Skia graphics library has a Vulkan backend
- Dawn (WebGPU) also has a Vulkan backend
Making 3D Universally Accessible

Android

Web

WebGL

OpenGL ES

ANGLE

WebGPU

Dawn

Web

Vulkan

SwiftShader

Metal

DX12

Vendor Driver
SwiftShader’s Transition to Vulkan-only

Application

OpenGL ES

GLSL
↓ASM

Renderer

Shader

ANGLE

glslang

Vulkan

Device

Pipeline / SPIR-V

Reactor

JIT

LLVM

Subzero

CPU
Reactor

- High level C-like language for code generation of low level CPU operations
- Produces code for a JIT compiler rather than executing that code
- Example:

```c
if(condition) {
    Int a = computeSomething(); // “a” is a Reactor integer object
    If(a == Int(0)) {
        a += Int(10); // The “+=” operator generates the necessary instruction(s)
    } // “If()” is evaluated at runtime and results in a dynamic branch inside the JITed code
} // Regular “if()” selects whether instructions are generated, no branch in resulting JITed code
```
SwiftShader Vulkan driver

- Passes 100% of dEQP-VK conformance test suite
- Vulkan 1.1 mandatory features only
- x86 and ARM, 32 bit and 64 bit
- WSI for desktop and mobile
Future directions

● PERFORMANCE!

● Tight integration with ANGLE
  ○ Short term, Vulkan features required by ANGLE for OpenGL ES 2.0 and 3.0 translation
  ○ Longer term, adding features required for OpenGL ES 3.1 and 3.2

● Integration into Chromium
  ○ Chromium Vulkan backend tests on build bots
  ○ ANGLE on SwiftShader Vulkan as WebGL fallback
  ○ SwiftShader Vulkan as Dawn (WebGPU) fallback

● Android
  ○ Tests on SwiftShader Vulkan
Highlights of using SwiftShader Vulkan

- Rapid prototyping of a new Vulkan feature
- Building with sanitizers (like ASAN or TSAN) to let fuzzers find cracks in the implementation/spec and improve test coverage
- Platform-independent shader debugging
- Finding bugs in applications making assumptions about available features
- macOS support
- Ability to test dEQP in under 15 minutes on a single (powerful) workstation
Conclusion

Conformant, consistent, hardware-independent results on Windows, Linux, macOS, Android, Fuchsia on x86, ARM, and more