Vulkan in Open-Source

A discussion of the new Vulkan graphics API and its impact on open-source software

Jason Ekstrand

intel Software

Intel OpenSource TECHNOLOGY CENTER

Who am I?

- Linux user since around 2000
- Started playing with 3-D graphics/modeling with Blender in 2004
- First started experimenting with OpenGL around 2007
- Got involved in the Wayland project in December of 2012
- Hired by Intel in June of 2014 to work on the i965 driver in Mesa
  - Best known for NIR, the new optimizing middle-layer in our shader compiler

I am going to try and answer 3 questions:

- What is the Vulkan API?
- Why do we need it?
- What will we (open-source community) do once it gets here?

What is the Vulkan API?

Vulkan is a new 3-D rendering and compute api from Khronos, the same cross-industry group that maintains OpenGL

- Redesigned from the ground-up; It is *not* OpenGL++
- Designed for modern GPUs and software
- Will run on currently shipping hardware

Why do we need a new 3-D API?

- OpenGL 1.0 was released by SGI in January of 1992
  - Based on the proprietary IRIS GL API
- Brian Paul released mesa in August of 1993
- Computers have advanced a lot in 24 years:
  - GPUs are more powerful and flexible
  - Memory has gotten cheaper
  - Multi-core CPUs are common
- OpenGL has done amazingly well over the last 24 years!

Why do we need a new 3-D API?

Not everything in OpenGL has stood the test of time:

- The OpenGL is API is a state machine
- OpenGL state is tied to a single on-screen context
- OpenGL hides *everything* the GPU is doing

This all made sense in 1992!

Why do we need a new 3-D API?

Vulkan takes a different approach:

- Vulkan is an object-based API with no global state
  - All state concepts are localized to a command buffer
- WSI is an extension of Vulkan, not the other way round.
- Vulkan far more explicit about what the GPU is doing
  - Texture formats, memory management, and syncing are client-controlled
  - Enough is hidden to maintain cross-platform compatibility
- Vulkan drivers do no error checking!

Why should you care about Vulkan?

Because it’s cool!

Why should you care about Vulkan?

Because the industry is finally starting to care about open-source*:

- Many of the tools will be open-source:
  - glslang (a GLSL -> SPIR-V compiler) already lives on github
  - loaders, validation layers, and API tracers will be released with the spec
- An open-source conformance suite!
- Open-source drivers:
  - There will be an open-source driver for Intel hardware
  - AMD plans to open-source their Vulkan driver eventually [XDC, 2015]
  - We (the community) can write more!

*All of the above information is publically available on the Khronos website and/or in talks given at open-source conferences

Why should you care about Vulkan?

It provides opportunities for open-source

- It's a great API to target for apps and drivers
- Easier to integrate into toolkits
  - No more thread-local context to manage
  - Sane object model with better multithreading
- Easier to package and distribute
- May bring more apps to open platforms
  - Most games are written for DirectX and only later ported to OpenGL
  - If developers target Vulkan directly, Linux isn't a hard jump

Why should you care about Vulkan?

It brings challenges for open-source

- How do we mix OpenGL and Vulkan?
  - OpenGL will be around for a while
  - OpenGL and Vulkan will have to live in the same toolkit
- Developers need to learn about Vulkan
- Infrastructure needs to be built

Why should you care about Vulkan?

It provides opportunities for education

- Vulkan has a sharper learning curve than OpenGL
- Vulkan's object model matches hardware better

What happens next?

- The spec and initial drivers get released
- Distros package the loader and drivers
- Toolkits grow Vulkan support
- Apps and middleware libraries get written
- More free-software drivers get written

The good news: Work has already begun!

Questions?