Graphics and Compute Belong Together

GDC, March 2015
The Need for Vulkan

Ground-up design of a modern open standard API for driving high-efficiency graphics and compute on GPUs used across diverse devices.

In the twenty two years since OpenGL was invented - the architecture of GPUs and platforms has changed radically.

GPUs being used for graphics, compute and vision processing on a rapidly increasing diversity of platforms - increasing the need for cross-platform standards.
Vulkan Explicit GPU Control

Complex drivers lead to driver overhead and cross vendor unpredictability

- Error management is always active
- Driver processes full shading language source
- Separate APIs for desktop and mobile markets

Vulkan delivers the maximized performance and cross platform portability needed by sophisticated engines, middleware and apps

Traditional graphics drivers include significant context, memory and error management

Application responsible for memory allocation and thread management to generate command buffers

Direct GPU Control

Simpler drivers for low-overhead efficiency and cross vendor portability

Layered architecture so validation and debug layers can be unloaded when not needed

Run-time only has to ingest SPIR-V intermediate language

Unified API for mobile, desktop, console and embedded platforms
Cross Platform Challenge

- An explicit API that is also cross-platform needs careful design

One family of GPUs
One OS
One GPU on one OS

All Modern Platforms and GPUs
A challenge that needs...
Participation of key players
Proven IP Framework
Battle-tested cooperative model
The drive to not let the 3D industry fragment
Vulkan Multi-threading Efficiency

1. Multiple threads can construct Command Buffers in parallel. Application is responsible for thread management and synch.

2. Command Buffers placed in Command Queue by separate submission thread.
Vulkan - Enhancing Driver Reliability

Streamlined API is easier to implement and test

Cross-vendor Portability

SPIR-V intermediate language improves shader program portability and reduces driver complexity

Open source conformance test source components for community engagement
Vulkan Tools Architecture

- **Layered design for cross-vendor tools innovation and flexibility**
  - IHVs plug into a common, extensible architecture for code validation, debugging and profiling during development without impacting production performance

- **Common Loader used to enable use of tools layers during debug**
  - Cross-vendor API calls provide debug data

Production Path (Performance)

- **Vulkan-based Title**
- **Vulkan’s Common Loader**
- **IHV’s Installable Client Driver**
- **Validation Layers**
- **Debug Layers**
- **Interactive Debugger**

Debug Layers can be installed during Development

Debug information via standardized API calls
**Vulkan Tools Ecosystem**

- Extensible modular architecture encourages many fine-grained layers - new layers can be added easily
- Khronos encouraging open community of tools e.g. shader debugging
- Valve, LunarG, Codeplay and others are already driving the development of open source Vulkan tools
- Customized interactive debugging and validation layers will be available together with first drivers
SPIR-V Unleashes Language Innovation

- First open standard cross-API intermediate language for parallel compute and graphics
  - Can natively represent Vulkan and OpenCL source languages
  - Including full flow control, graphics and parallel constructs not in LLVM

- Fully specified Khronos-defined standard
  - Khronos is working on creating SPIR-V <-> LLVM conversion tools

- Splitting the Compiler Chain enables parallel software/hardware innovation
  - Front-ends for languages can access multiple production quality backends
  - Back-ends using multicore, GPU, vector, VLIW or other technologies can reuse production quality language frontends and abstractions
  - Tooling - encourages innovation in advanced program analysis and optimization of programs in SPIR form
SPIR-V for Developers

- Developers can use same front-end compiler across multiple platforms
  - Eliminating major source of cross-vendor portability
- Reduces runtime shader compilation time
  - Driver only has to process SPIR-V not full source language
- Don’t have to ship shader source code
  - Provides a measure of IP protection
- SPIR-V is core in OpenCL 2.1 AND Vulkan
  - Exposes machine model for OpenCL 1.2, 2.0, 2.1 and Vulkan
  - Supports OpenCL 1.2, 2.0, 2.1 kernel languages
  - Supports GLSL shader language (under development)

SIGNIFICANT OPPORTUNITY TO LEVERAGE AND CONVERGE LANGUAGES FOR GRAPHICS AND COMPUTE
Vulkan Language Ecosystem

GLSL Shader Source

GLSL will be first shading language supported by Vulkan

GLSL to SPIR-V Translator

Game Engines
Can flexibly target SPIR-V and Vulkan back-ends

Future diversity in domain-specific languages, frameworks and tools
E.g. C++ Shading Language

SPIR-V supported in Vulkan core

Vulkan Runtime

Device X
Device Y
Device Z

Khronos is considering open sourcing compiler front-ends

GLSL will be first shading language supported by Vulkan

Game Engines
Can flexibly target SPIR-V and Vulkan back-ends

Future diversity in domain-specific languages, frameworks and tools
E.g. C++ Shading Language

SPIR-V supported in Vulkan core

Vulkan Runtime

Device X
Device Y
Device Z

Khronos is considering open sourcing compiler front-ends
# Ground-up Explicit API Redesign

<table>
<thead>
<tr>
<th>ground-up Explicit API Redesign</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Original architecture</th>
<th>Matches architecture of modern platforms including mobile platforms with unified memory, tiled rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originally architected for graphics workstations with direct renderers and split memory</td>
<td>Matches architecture of modern platforms including mobile platforms with unified memory, tiled rendering</td>
</tr>
<tr>
<td>Driver does lots of work: state validation, dependency tracking, error checking. Limits and randomizes performance</td>
<td>Explicit API – the application has direct, predictable control over the operation of the GPU</td>
</tr>
<tr>
<td>Threading model doesn’t enable generation of graphics commands in parallel to command execution</td>
<td>Multi-core friendly with multiple command buffers that can be created in parallel</td>
</tr>
<tr>
<td>Syntax evolved over twenty years – complex API choices can obscure optimal performance path</td>
<td>Removing legacy requirements simplifies API design, reduces specification size and enables clear usage guidance</td>
</tr>
<tr>
<td>Shader language compiler built into driver. Only GLSL supported. Have to ship shader source</td>
<td>SPIR-V as compiler target simplifies driver and enables front-end language flexibility and reliability</td>
</tr>
<tr>
<td>Despite conformance testing developers must often handle implementation variability between vendors</td>
<td>Simpler API, common language front-ends, more rigorous testing increase cross vendor functional/performance portability</td>
</tr>
</tbody>
</table>
Vulkan Status

- **Rapid progress since June 2014**
  - Significant proposals and IP contributions received from members

- **Participants come from all segments of the graphics industry**
  - Including an unprecedented level of participation from game engine ISVs

- **Initial specs and implementations expected this year**
  - Will work on any platform that supports OpenGL ES 3.1 and up

*Working Group Participants*
Khronos Open Standards for Graphics and Compute

A comprehensive family of APIs to address the full spectrum of developer requirements

1990’s
Workhorse cross-platform API for professional 3D apps and gaming

2000’s
Ubiquitous API for mobile gaming and general purpose graphics

2008
Heterogeneous parallel computation

Portable intermediate representation for graphics and parallel compute

2014
High-efficiency GPU graphics and compute API for performance critical apps

2015

All APIs will be evolved and maintained to meet industry needs. Rich mix of open technologies for future innovation