KHRONOS GROUP

Press Pre-Briefing

GDC 2015

Neil Trevett | Khronos President
NVIDIA Vice President Mobile Ecosystem

All Materials Embargoed
Until Tuesday 3rd March, 12:01AM Pacific Time

Khronos Connects Software to Silicon

Open Consortium creating ROYALTY-FREE, OPEN STANDARD APIs for hardware acceleration

Defining the roadmap for low-level silicon interfaces needed on every platform

Graphics, compute, rich media, vision, sensor and camera processing

Rigorous specifications AND conformance tests for cross-vendor portability

Acceleration APIs
BY the Industry
FOR the Industry

Well over a *BILLION* people use Khronos APIs
Every Day…

Significant Khronos API Ecosystem Advances

- **Reveal of Vulkan API – next generation graphics API**
  - Low overhead, high-efficiency graphics and compute on GPUs
  - Formerly discussed as Next Generation OpenGL Initiative
  - Reveal and demos at GDC – no formal specification yet
- **SPIR-V – paradigm shift to support both graphics *and* compute constructs**
  - Now will be used by both Vulkan AND OpenCL 2.1
  - Provisional specification released
- **OpenCL 2.1 provisional specification released**
  - C++ Kernel language, SPIR-V in core
  - Provisional specification released

The Need for Vulkan

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

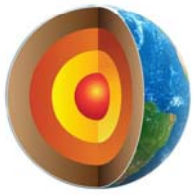
Vulkan Explicit GPU Control

Complex drivers lead to driver overhead and cross vendor unpredictability	Application	Application responsible for memory allocation and thread management to generate command buffers	Simpler drivers for low-overhead efficiency and cross vendor portability
Error management is always active	Traditional graphics drivers include significant context, memory and error management		Layered architecture so validation and debug layers can be unloaded when not needed
Driver processes full shading language source		Direct GPU Control	Run-time only has to ingest SPIR-V intermediate language
Separate APIs for desktop and mobile markets	GPU	GPU	Unified API for mobile, desktop, console and embedded platforms

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

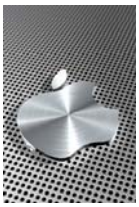
Cross Platform Challenge

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



One family of GPUs

One OS

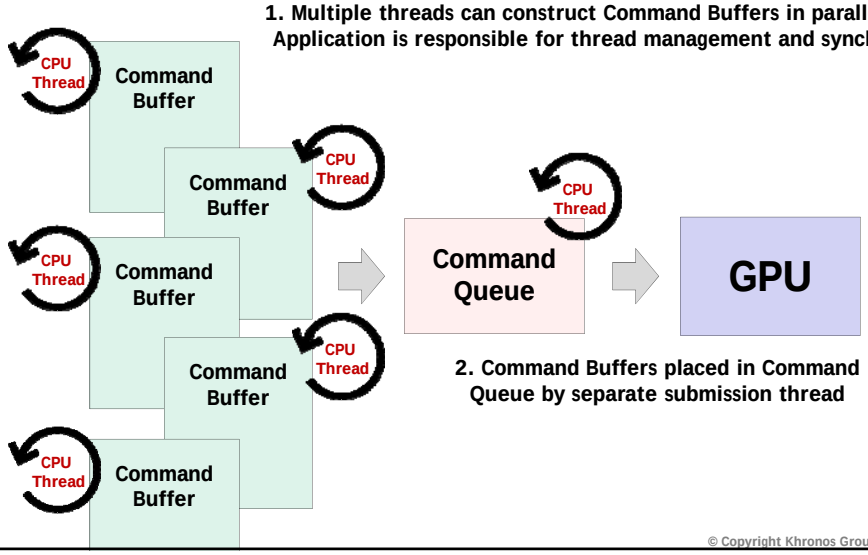


One GPU on one OS

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

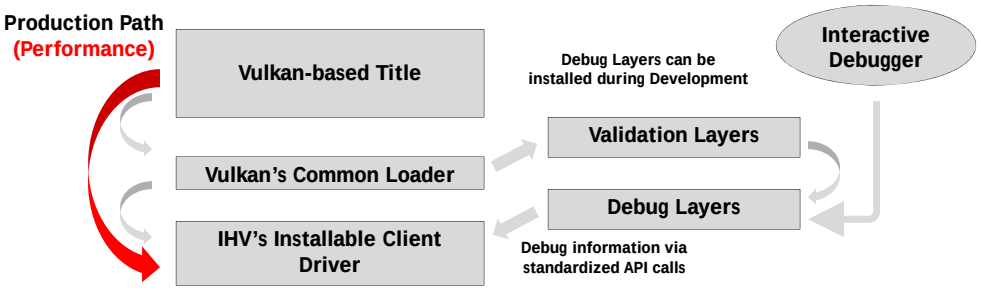
Cross-vendor Portability

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

Open source conformance test source components for community engagement

Vulkan Tools Architecture

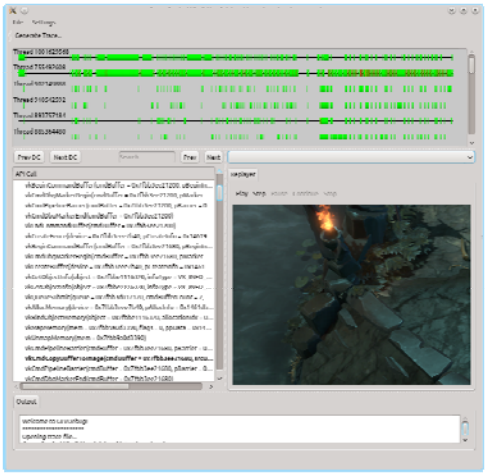
- **Layered design for cross-vendor tools innovation and flexibility**
  - IHVs plug into a common, extensible architecture for code validation, debugging and profiling during development without impacting production performance
- **Common Loader used to enable use of tools layers during debug**
  - Cross-vendor API calls provide debug data



Vulkan Tools Ecosystem

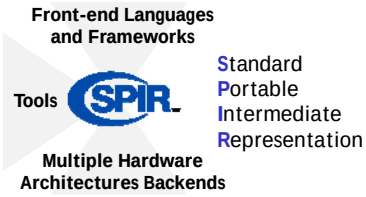
- **Extensible modular architecture encourages many fine-grained layers - new layers can be added easily**
- **Khronos encouraging open community of tools e.g. shader debugging**
- **Valve, LunarG, Codeplay and others are already driving the development of open source Vulkan tools**
- **Customized interactive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

Prototype Vulkan Debugger from Valve and LunarG



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

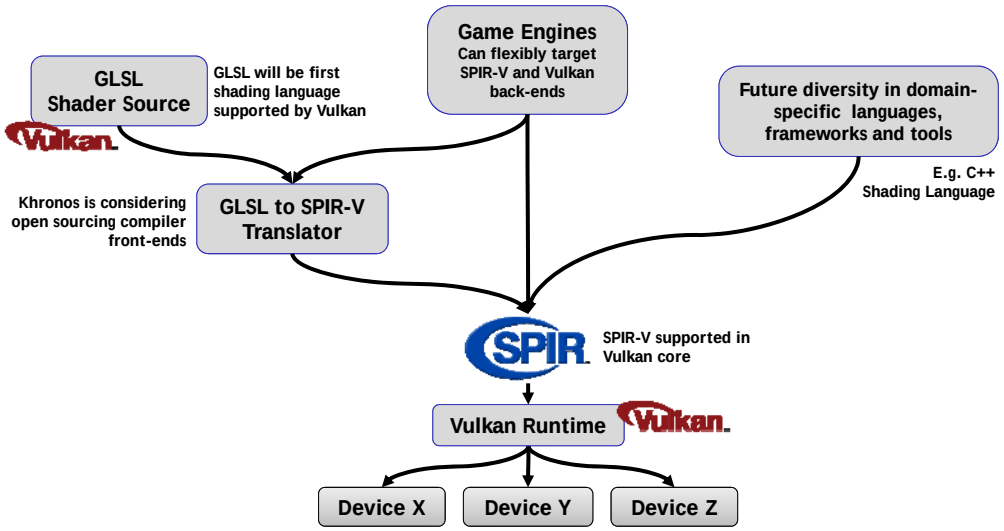


SPIR-V for Developers

- **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



Ground-up Explicit API Redesign

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

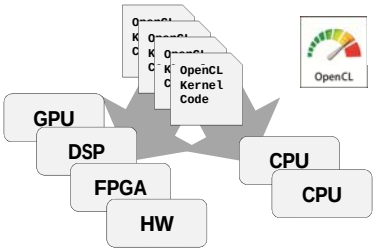
Vulkan Status

- **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

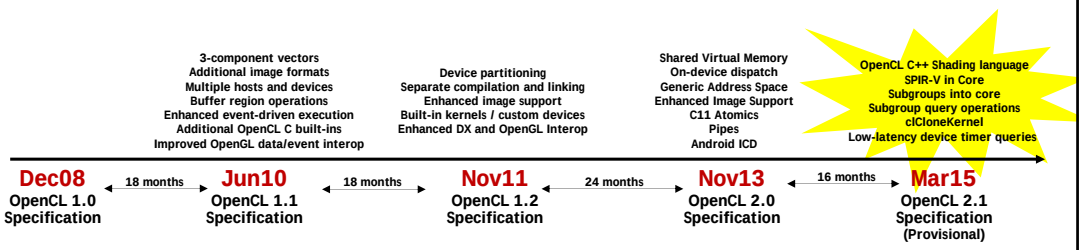
OpenCL – Portable Heterogeneous Computing

- **Portable Heterogeneous programming of diverse compute resources**
  - Targeting supercomputers -> embedded systems -> mobile devices
- **One code tree can be executed on CPUs, GPUs, DSPs, FPGA and hardware**
  - Dynamically interrogate system load and balance work across available processors
- **OpenCL = Two APIs and Kernel language**
  - C Platform Layer API to query, select and initialize compute devices
  - C Runtime API to build and execute kernels across multiple devices

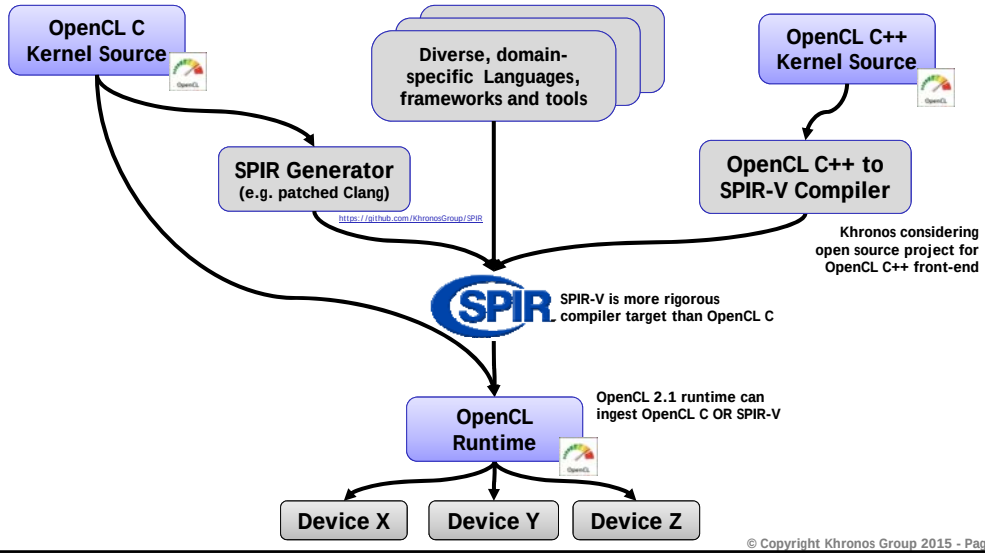


OpenCL 2.1 Provisional Released!

- **New OpenCL C++ kernel language based on a subset of C++14**
  - Significantly enhanced programmer productivity
  - OpenCL C still supported to preserve kernel code investment
- **Support for the new Khronos SPIR-V intermediate language in core**
  - SPIR-V now used by both OpenCL 2.1 and the new Vulkan graphics API
  - OpenCL 1.2 and 2.0 kernel languages also supported by SPIR-V for backwards compatibility

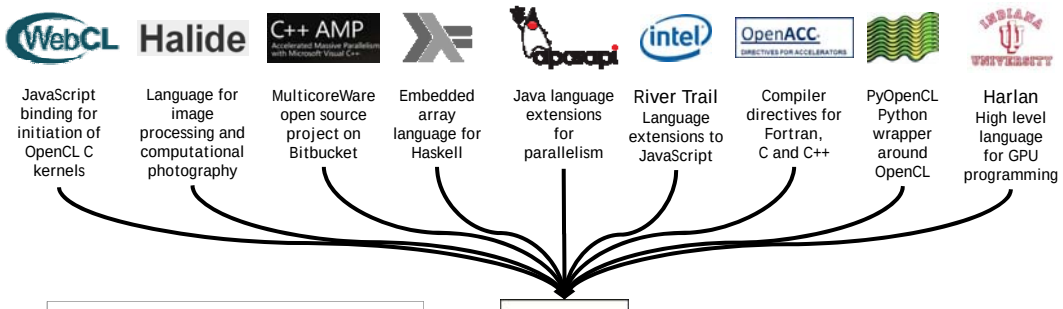


New OpenCL 2.1 Compiler Ecosystem



OpenCL as Parallel Language Backend

Approaching 200 languages, framework an projects using OpenCL as a back-end



Support for SPIR-V in OpenCL core will accelerate this trend as it is a more rigorously defined compiler target than OpenCL C

OpenCL

OpenCL provides vendor optimized, cross-platform, cross-vendor access to heterogeneous compute resources

OpenCL C++

- **The OpenCL C++ kernel language is a static subset of C++14**
  - Frees developers from low-level coding details without sacrificing performance
- **C++14 featured removed from OpenCL C++ for parallel programming**
  - Throwing and catching exceptions (throw, catch)
  - Allocation and release of memory (new, delete)
  - Virtual functions and abstract classes (virtual)
  - Function pointers, Recursion and goto
- **Classes, lambda functions, templates, operator overloading etc..**
  - Fast and elegant sharable code - reusable device libraries and containers
  - Compile-time polymorphism via template meta-programming for highly adaptive software that delivers tuned performance across diverse platforms

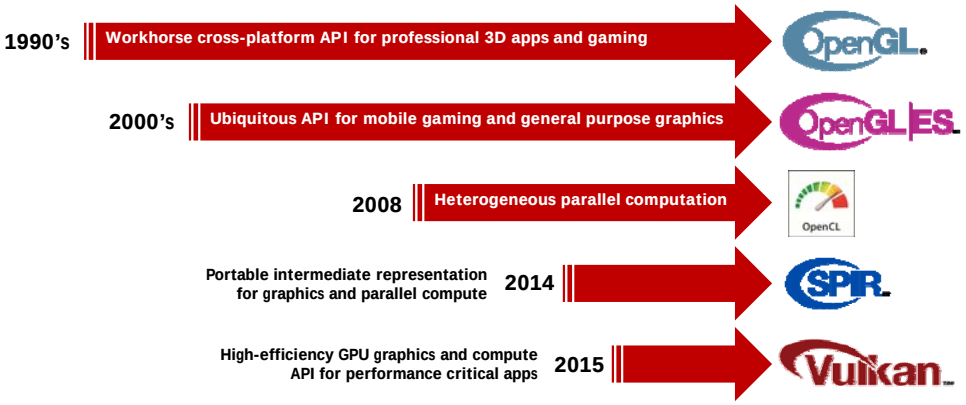
OpenCL

OpenCL 2.1 API Enhancements

- **Subgroup functionality moved into core with additional subgroup query operations**
  - Expose hardware threads/warps/wavefronts and their cross-lane operations
  - Host queries for forward progress extension, and workgroup->subgroup mapping
- **clCloneKernel enables copying of kernel objects and state**
  - Safe implementation of copy constructors in wrapper classes
  - Used to pass kernel to second host thread, or for C++ wrappers for kernel objects
- **Low-latency device timer queries**
  - Support alignment of profiling data between device and host code
- **clCreateProgramWithIL**
  - Enables ingestion of SPIR-V code by the runtime
- **Priority and throttle hint extensions for queues**
  - Specify execution priority on a per-queue basis
- **Zero-size enqueue**
  - Zero-sized dispatches are valid from the host

Khronos Open Standards for Graphics and Compute

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



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

Call to Action

- **Special Khronos sessions at GDC for more details:**
  - Details in press releases and here:
  - https://www.khronos.org/news/events/gdc-2015
- **Khronos seeking feedback on Vulkan, SPIR and OpenCL 2.1**
  - Links provided on Khronos forums
  - https://www.khronos.org/opencl/opencl_feedback_forum
  - https://www.khronos.org/spir_v_feedback_forum
  - https://www.khronos.org/vulkan/vulkan_feedback_forum
- **Any company or organization is welcome to join Khronos for a voice and a vote in any of these standards**
  - www.khronos.org
  - ntrevett@nvidia.com

KHRONOS GROUP