Vulkan SDK
Where We Started
Where We are Going

Karen Ghavam
CEO and Engineering Director
LunarG, Inc
Today’s Talk

- A Historical Look – Vulkan API and the Vulkan SDK
- Developer Tools – Challenges, Successes, and the Future
A Brief History
A Brief History of Vulkan

SIGGRAPH in Vancouver

- Khronos call for participation in defining the "glNext" API
  - OpenGL, Direct3D were mature with minor feature updates
  - A need to scrape away the abstractions included in OpenGL and Direct3D
  - Mantle, Direct3D 12, Metal all demonstrated the needs of the future

- Features
  - High-efficiency access to graphics and compute on modern GPUs
  - Abstraction removal – explicit GPU and CPU control over workloads
  - Multithreading-friendly API with reduced overhead
  - Common shader programming intermediate language (SPIR-V)
A Brief History of Vulkan

August 2014 |
First Vulkan POC

March 2015 |
- Vulkan ILO Driver (Linux, Intel GPU)
- Valve Source2 Engine
- Key feedback for the Vulkan 1.0 Specification

February 2016
A Brief History of Vulkan

August 2014

March 2015

February 2016

GDC

- Technical Previews
- Valve Source2 Engine
- Vulkan ILO Driver
A Brief History of Vulkan

August 2014

March 2015

February 2016

Public Launch
Vulkan SDK – A Retrospective
The Vulkan SDK (2016)

Vulkan Loader
SPIR-V Tools
Vulkaninfo
Validation Layer

Only SPIR-V assembler and disassembler

GLSL to SPIR-V

apidump
glslang

GLM

Tarball
Windows
The Vulkan SDK (2017)

2016

2017

Vulkan Loader

SPIR-V Tools

GPU Open Source

SPIR-V Validator

Vulkan Cross HPP

GLSLang

APIDump

Vulkan-HPP

ScreenShot

SDL

Monitor

GLM

Validation Layer

vulkaninfo

SPIRV Cross

shaderc

Tarball

Windows
The Vulkan SDK (2018)

First version of GPU-AV
First version of Best Practices, called the “Assistant Layer”
The Vulkan SDK (2019)

- Vulkan Loader
- SPIR-V Optimizer
- shaderc
- SPIR-V Tools
- SPIR-V Validator
- Vulkan-HPP
- glslang
- MoltenVK
- vkconfig
- Validation Layer
- Vulkaninfo
- apidump
- VKVIA
- Screeshot
- Monitor
- GLM
- Ubuntu Packages
- Tarball
- License Registry
- Windows
The Vulkan SDK (2020)

- Vulkan Loader
- SPIR-V Tools
- Vulkaninfo
- vkconfig
- Validation Layer
- First release of validating the synchronization API
- Replaced vktrace / vkreplay API capture and replay

- SPIR-V Optimizer
- shaderc
- SPIR-V Validator
- SPIR-V Optimizer
- DXC
- SPIR-V Reflect
- SPIR-V Reflect
- glslang
- Vulkan-HPP
- GFX Reconstruct
- Total rewrite – setting the groundwork for the future

- MoltenVK
- SPIR-V Visualizer
- apidump
- Vulkan-HPP
- Screenshot
- Monitor
- GLM

- Ubuntu Packages
- Tarball
- Windows
- License Registry
- Apple
Validation within and between queue submissions and across queues.
The Vulkan SDK (2024)

- Vulkan Loader
- Vulkan-HPP
- Vulkan-rt
- Vulkan-PP
- Vulkan-SPIR

- SPIR-V Optimizer
- SPIR-V Tools
- SPIR-V Validator
- SPIR-V Reflect
- glslang

- DXC
- apidump
- GFX Reconstruct
- Screenshot

- shaderc
- Profiles Toolset
- Vulkaninfo
- Vulkaninfo
- Vulkaninfo
- Vulkaninfo

- Validate Layer
- Emulation Layers
- Debug GPU hangs and crashes

License
Registry
Ubuntu
Packages
Tarball
Windows
The Vulkan SDK

- Vulkan Loader
- vkconfig
- Validation Layer
- vulkaninfo
- Emulation Layers
- SPIR-V Optimizer
- SPIR-V Tools
- Profiles Toolset
- GPUInfo
- VOLK
- shadec
- SPIR-V Validator
- apidump
- VKVIA
- SPIR-V Reflect
- SPIR-V Cross
- glslang
- Vulkan-HPP
- Screenshot
- VMA
- SPIR-V Visualizer
- SDL
- Monitor
- GLM

License
Registry

Ubuntu Packages
Tarball

Windows
arm
## Vulkan SDK Download Page (vulkan.lunarg.com)

**Windows**

<table>
<thead>
<tr>
<th>Version</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.0.0</td>
<td><a href="https://vulkan.lunarg.com/">Download</a></td>
</tr>
</tbody>
</table>

**Linux**

<table>
<thead>
<tr>
<th>Version</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.0.0</td>
<td><a href="https://vulkan.lunarg.com/">Download</a></td>
</tr>
</tbody>
</table>

**Mac**

<table>
<thead>
<tr>
<th>Version</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.0.0</td>
<td><a href="https://vulkan.lunarg.com/">Download</a></td>
</tr>
</tbody>
</table>
Vulkan SDK Downloads are Healthy

Windows SDK: ~38,000/week
Mac SDK: ~5800/week
Linux SDK: ~6000/week

Note: Numbers are for Linux “Tarball” only and don’t include Ubuntu packages also available from LunarG or other Linux distros.
How is this funded?
How is this funded?

VALUE
How is this funded?
How is this funded?

- Valve
- Google
- Samsung
- Qualcomm
- Arm
- AMD
- Meta
The First Vulkan SDK

- An INCOMPLETE Validation Layer implementation
- The first Vulkan Loader implementation
- Windows and Linux only
Validation Layer - Then and Now

June 2018
Validation Layer - Then and Now

- June 2018: 90K
- June 2024: 730K
Validation Layer and VUIDs

- VUID - Valid Usage ID
  - Assigned to each API usage
  - How that part of the API must be used
- Validation Layer is validating the VUIDs
  - “Error Checking”
The Validation Layer - Today

- Healthy open-source project with robust functionality
  - GPU-assisted validation - to support the bindless attributes of the Vulkan API
The Validation Layer - Today

- Healthy open-source project with robust functionality
  - GPU-assisted validation - to support the bindless attributes of the Vulkan API
  - Synchronization Validation - detection of race conditions in otherwise correct Vulkan programs
    - 2019 - Hazard detection within a single buffer
      - 18 man months of effort!
    - 2022 - Hazard detection within and between queue submissions and across queues
      - 24 man months of engineering effort!
  - These two versions enable baseline functionality and does not cover all Vulkan extensions. More to do!
The Validation Layer - Today

- CI Test Farm
  - SW testing
    - Mock ICD
  - GPU HW
    - Nvidia
    - AMD
    - Intel
    - Android
- Windows, Linux, Android, macOS
The Validation Layer

We aren't done yet!
Vulkan API continues to evolve!
Opportunities Presented by the Technology
Validation Layer - Vulkan Synchronization

Semaphores
Main cross-queue synchronization mechanism

Events and Barriers
Synchronization of commands submitted to a single queue

Fences
Synchronize work between the device and the host

Validation Layer Improvement Opportunity:

- High Performance Overhead due to required volume of state tracking
- Ongoing improvement opportunity: Performance tuning
Validation Layer - Descriptor Indexing Validation

- Descriptors invoked from shaders
  - Only used descriptors required to be valid
  - Might only use “10” out of millions
- Initial validation implementation
  - Slowed app from 100+ FPS to a fractional value!
  - All descriptors were being validated, regardless if used!
- Performance Improvement!
  - Using instrumented shaders on the GPU
    - Detect which descriptors are actually used
  - Only validate used descriptors
Validation Layer – GPU-AV Performance

- GPU-AV requires instrumenting shaders
- Shaders become bloated; impacting performance
  - Pipeline compile times
  - Runtime shader execution
Validation Layer – Latency in Error Reporting

- Errors detected well after the Vulkan API call that caused them (aka at vkQueueSubmit time)
- Difficult to provide meaningful error messages
- Opportunity to improve error messages:
  - Storing information for later use without unbearable performance impacts
**Open-source Vulkan Developer Tools**

*Included in the Vulkan SDK*

<table>
<thead>
<tr>
<th>Toolset</th>
<th>GFXReconstruct - API Capture and Replay</th>
<th>Vulkan Loader Validation Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXC</td>
<td>Cross-platform (Windows, Linux, Android, macOS)</td>
<td>vkconfig</td>
</tr>
<tr>
<td>SPIR-V Cross</td>
<td>Run Vulkan workloads during GPU development</td>
<td>GFX Reconstruct</td>
</tr>
<tr>
<td>MoltenVK</td>
<td>Debug Vulkan applications</td>
<td>VKVIA</td>
</tr>
</tbody>
</table>
|                  | Regression testing using real application workloads |)
|                  | Underlying engine for profiling and debugging tools |)

- DXC
- SPIR-V
- MoltenVK
- Validator
- Toolset
- apidump
- GFX Reconstruct
- VKVIA
- glslang
- Vulkan-HPP
- Screenshot
- VMA
- SPIR-V Reflect
- SDL
- Monitor
- GLM
- SPIR-V Visualizer
- GFXReconstruct
- Vulkan-HP
GFXReconstruct - API Explicitness

- Portability Challenge
  - Vulkan API is explicit
  - Hence captures from one GPU can’t be replayed on another GPU

- Conflicting Use Cases
  - Exact API calls needed for analysis
  - Use existing captures on newer/different GPUs

- Opportunity: How to enable some portability of captures
  - Collect additional data?
  - Translation layer?
The GPU-centric Universe

● GPUs - no longer "Graphics Processing Units"
  ○ Efficient processing of large blocks of data simultaneously
  ○ Compute - AI and ML

● Less Graphics API usage on the CPU
  ○ Rendering complexity still increasing via GPU driven rendering

● Many workloads moving to the GPU
  ○ Maximize utilization of GPU features
  ○ Reduce CPU interaction
D3D12 Work Graphs – GPU Autonomy

- **GPU Autonomy**
  - GPU Feeds itself
- **Dynamic Work Expansion**
  - Shader threads (producers) requesting work to run (consumers)
- **Removes round trips to CPU**
- **Currently not available in Vulkan**
GFXReconstruct - GPU Autonomy

- Information no longer known at a function device call from the CPU side
- Addresses baked into capture content
  - Needs to be a different address during replay
GPU-Centric Universe : Developer Tools
Implications

● Debugging on a CPU vs GPU
  ○ CPUs provide the Instruction Set Architecture (ISA) and ability to step thru code
  ○ GPUs can be a black box and intrinsically different
    ○ Imagine stepping through 1 of a million items in a massive parallelism environment!

● Cross-GPU open-source tools are useful today
  ○ Evolve the tools for the GPU-centric universe
  ○ Cooperation needed from many parties
    ○ IHVs
    ○ Specification definitions
    ○ Tool writers
An Example API “hook”

- Vulkan “bufferDeviceAddressCaptureReplay”
  - Enable in driver during capture
  - Query memory location upon allocation
  - Can use that same memory allocation during replay
  - Current limitation: Not guaranteed to work from one vendor to another
From the launch of Vulkan to Today…

- There is ONE Industry-standard Vulkan desktop SDK
  - Wide adoption
  - Strong satisfaction
  - Open and free for all developers
  - Cross-platform SDK: Windows-x64/x86, Windows on arm, Linux, Apple platforms

- Valuable developer tools
  - Robust in features and reliability
  - Providing real value to Vulkan application developers
LunarG Purpose Continues!
Evolve the tools for a GPU-centric universe!
Why Vulkan?
Why Vulkan? Cross-platform support

Same API for Mobile, desktop, (and Apple platforms)

Windows 10

Windows 11
Why Vulkan? Improved Cross-vendor Compatibility

One API usage validator used by all (Vulkan-ValidationLayer)
Why Vulkan? Improved Performance

- Explicit application control over GPU and CPU workloads
- Multithreading-friendly API
- No more error checking in the Vulkan driver
Why Vulkan? Shader Language Flexibility

Standardized Intermediate Language (SPIR-V)

- Eliminates front-end compilers from drivers
  - Reduce driver complexity
- Front-end language flexibility
  - Improve portability
Why Vulkan? Open Standard

Strengthened ecosystem focus

- Embrace and engage with the ISVs
- Open conformance test suite - more rigor
- More control put in the developer's hands
Validation Layer – So Many Vulkan Objects!

- The Sheer Number of Vulkan Objects – complexity
- Different functions and usages
  - Rules for how can they be used
  - Rules for order of creation

→ Complexity in the validation layer
GFXReconstruct - Vulkan Swapchain

- Different swapchain modes present and return images in different order
  - From run to run
- No swapchain presentation mode guarantees return order!
- GFXReconstruct Opportunity: How can we display the correct image during replay?
  - Solution: Implemented a virtual swapchain
  - Same number of images in replay as in capture
  - Use the indices in the same order from capture to replay
Who is LunarG?

• Independent, privately owned software consultancy
• Passionate about 3D graphics & compute technology
• Industry leading, 3D-graphics software experts with decades of experience
  • Vulkan, OpenXR, OpenGL, Direct3D, Metal, …
  • Developer tools, drivers, performance tuning…
• Developers of proprietary and open source drivers, tools, & software solutions
• Founded in 2009 – Headquarters in Fort Collins, CO
• Delivers the Vulkan SDK