SIGGRAPH 2024
DENVER+ 28 JUL — 1 AUG

KHRONOS GROUP

Vulkan®

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

LUNARG

A Brief History of Vulkan

KHRONOS GROUP

August 2014 — March 2015 — February 2016

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)

LUNARG

A Brief History of Vulkan

First Vulkan POC

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

A Brief History of Vulkan

GDC

- Technical Previews
- Valve Source2 Engine
- Vulkan ILO Driver

A Brief History of Vulkan

KHRONOS GROUP

August 2014 — March 2015 — February 2016

Public Launch

LUNARG

Vulkan SDK – A Retrospective

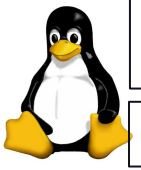
LUNARG

The Vulkan SDK (Today)

	Vulkan Loader	vkconfig	Validation Layer	
SPIR-V Optimizer	SPIR-V Tools	Crash Diagnostic Layer	vulkaninfo	Emulation Layers
shaderc	SPIR-V Validator	Profiles Toolset	GPUInfo	VOLK
DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKVIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

License Registry

Ubuntu Packages



Tarball

Windows

LUNARG

The Vulkan SDK (2016)

2016

Vulkan 1.0

Vulkan Loader

SPIR-V Tools

Validation Layer

vulkaninfo

Only SPIR-V assembler and disassembler

apidump

glslang

GLSL to SPIR-V

GLM

Tarball

Windows

LUNARG

The Vulkan SDK (2017)

Vulkan Loader

Validation Layer

SPIR-V Optimizer

SPIR-V Tools

vulkaninfo

shaderc

SPIR-V Validator

apidump

VKVIA

SPIR-V Cross

glslang

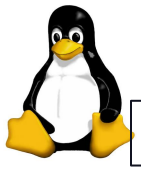
Vulkan-HPP

Screenshot

SDL

Monitor

GLM



Tarball

Windows

LUNARG

The Vulkan SDK (2018)

2016 2017 2018

Vulkan Loader

vkconfig

Validation Layer

First version of GPU-AV

First version of Best Practices, called the "Assistant Layer"

SPIR-V Optimizer

SPIR-V Tools

vulkaninfo

shaderc

SPIR-V Validator

apidump

VKVIA

Ubuntu Packages

Tarball

SPIR-V Cross

glslang

Vulkan-HPP

Screenshot

Windows

MoltenVK

SDL

Monitor

GLM

Vulkan 1.1

LUNARG

The Vulkan SDK (2019)

Vulkan Loader

vkconfig

Validation Layer

SPIR-V Optimizer

SPIR-V Tools

vulkaninfo

License Registry

shaderc

SPIR-V Validator

apidump

VKVIA

Ubuntu Packages

SPIR-V Cross

glslang

Vulkan-HPP

Screenshot

Tarball

MoltenVK

SDL

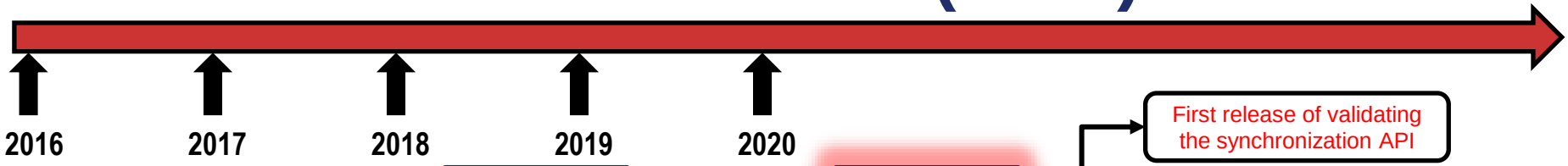
Monitor

GLM

Windows

LUNARG

The Vulkan SDK (2020)



Vulkan Loader

vkconfig

Validation Layer

First release of validating the synchronization API

SPIR-V Optimizer

SPIR-V Tools

vulkaninfo

Replaced vktrace / vkreplay API capture and replay

License Registry

shaderc

SPIR-V Validator

Total rewrite – setting the groundwork for the future

DXC

SPIR-V Reflect

apidump

GFX Reconstruct

VKVIA

Ubuntu Packages

SPIR-V Cross

glslang

Vulkan-HPP

Screenshot

Tarball

MoltenVK

SPIR-V Visualizer

SDL

Monitor

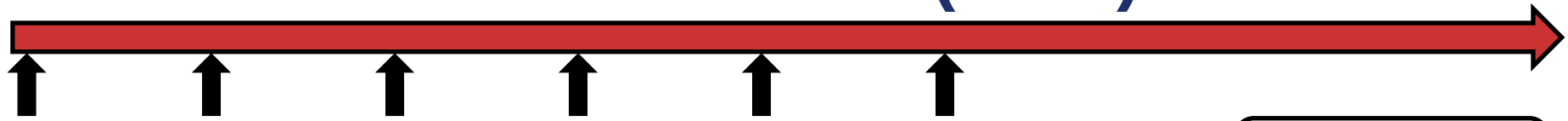
GLM

Windows

Vulkan 1.2

LUNARG

The Vulkan SDK (2021)



License Registry

Vulkan Loader

vkconfig

Validation Layer

Synchronization2 API Validation

SPIR-V Optimizer

SPIR-V Tools

vulkaninfo

Emulation Layers

Synchronization2 emulation

License Registry

shaderc

SPIR-V Validator

DXC

SPIR-V Reflect

apidump

GFX Reconstruct

VKVIA

Ubuntu Packages

SPIR-V Cross

glslang

Vulkan-HPP

Screenshot

Tarball

MoltenVK

SPIR-V Visualizer

SDL

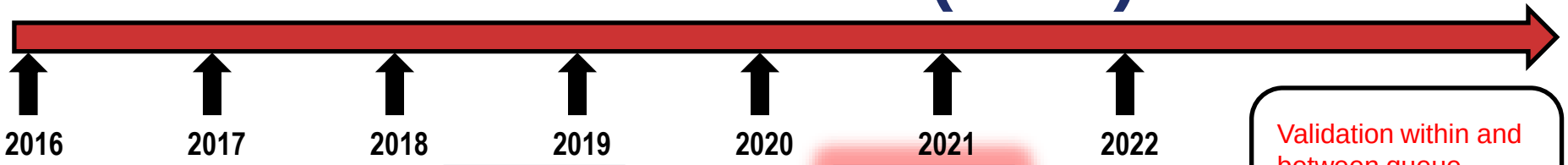
Monitor

GLM

Windows

LUNARG

The Vulkan SDK (2022)



Validation within and between queue submissions and across queues

License Registry

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

License Registry

Ubuntu Packages

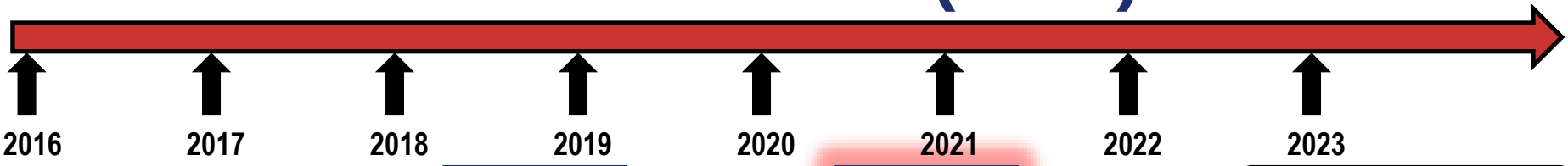
Tarball

Windows

Vulkan 1.3

LUNARG

The Vulkan SDK (2023)



License Registry

Vulkan Loader

vkconfig

Validation Layer

VK_EXT_shader_object validation

SPIR-V Optimizer

SPIR-V Tools

vulkaninfo

Emulation Layers

Shader Object Emulation Layer

License Registry

shaderc

SPIR-V Validator

Profiles Toolset

GPUInfo

VOLK

DXC

SPIR-V Reflect

apidump

GFX Reconstruct

VKVIA

SPIR-V Cross

glslang

Vulkan-HPP

Screenshot

VMA

MoltenVK

SPIR-V Visualizer

SDL

Monitor

GLM

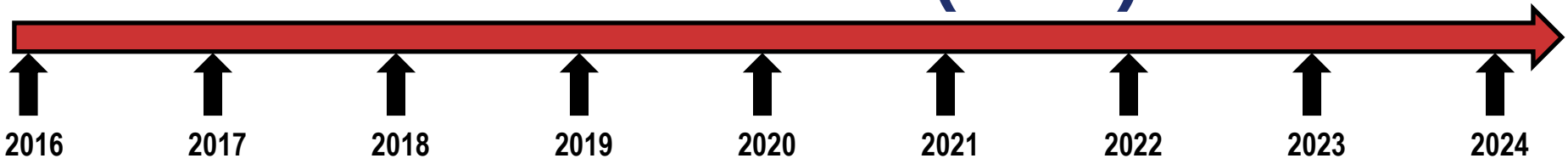
Ubuntu Packages

Tarball

Windows

LUNARG

The Vulkan SDK (2024)



	Vulkan Loader	vkconfig	Validation Layer	
SPIR-V Optimizer	SPIR-V Tools	Crash Diagnostic Layer	vulkaninfo	Emulation Layers
shaderc	SPIR-V Validator	Profiles Toolset	GPUInfo	VOLK
DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKVIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

License Registry

Ubuntu Packages

Tarball

Windows

Debug GPU hangs and crashes

NEW!

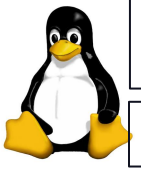
LUNARG

The Vulkan SDK

Windows arm

License Registry

Ubuntu Packages



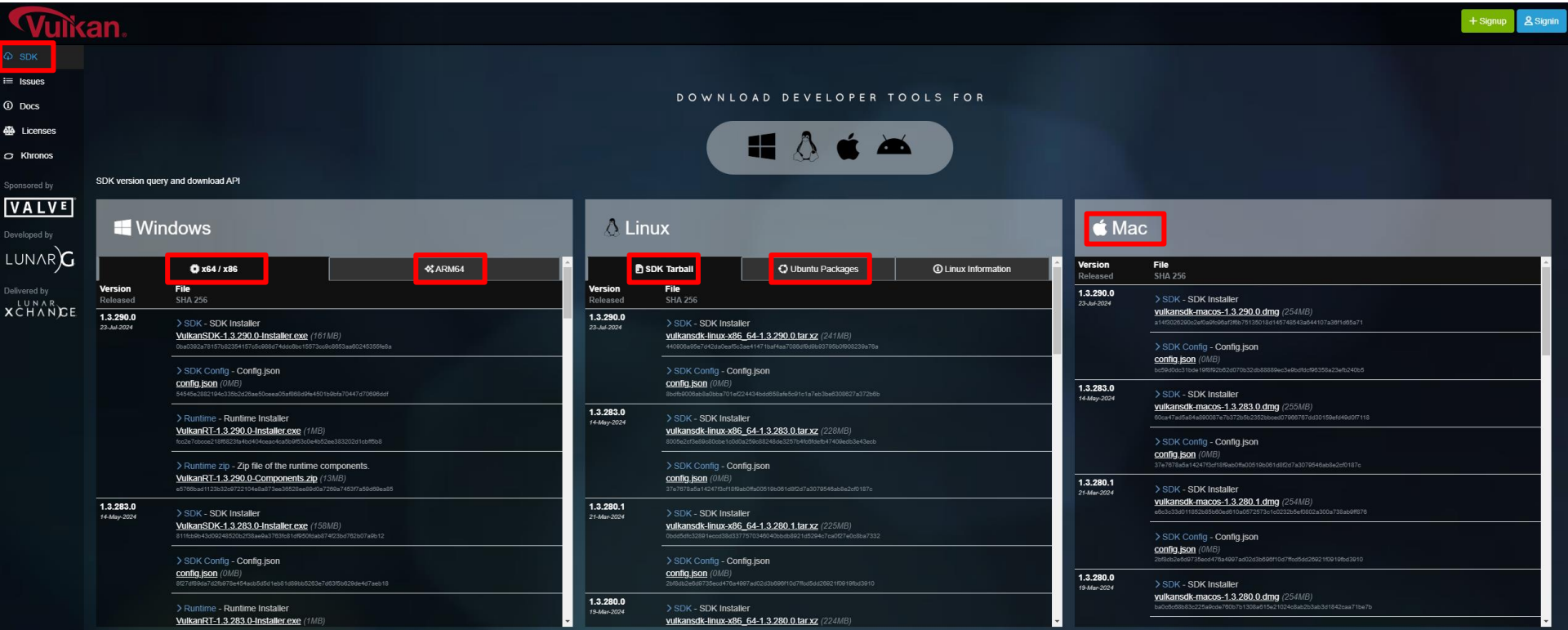
Tarball

Windows

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

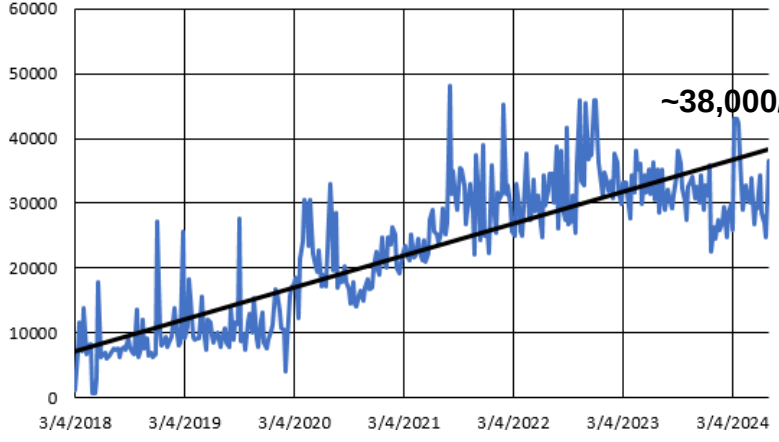
LUNARG

Vulkan SDK Download Page (vulkan.lunarg.com)



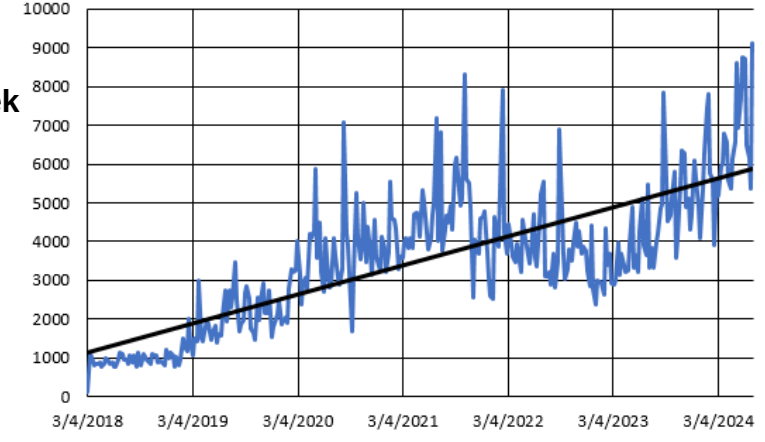
Vulkan SDK Downloads are Healthy

Windows SDK



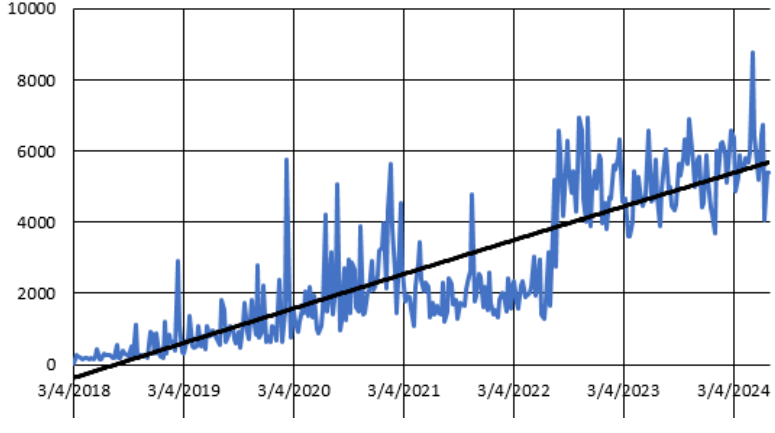
~38,000/week

Linux SDK



~6000/week

Mac SDK



~5800/week

Note: Numbers are for Linux “Tarball” only and don’t include Ubuntu packages also available from LunarG or other linux distros

LUNARG

How is this funded?

How is this funded?

LUNARG

How is this funded?

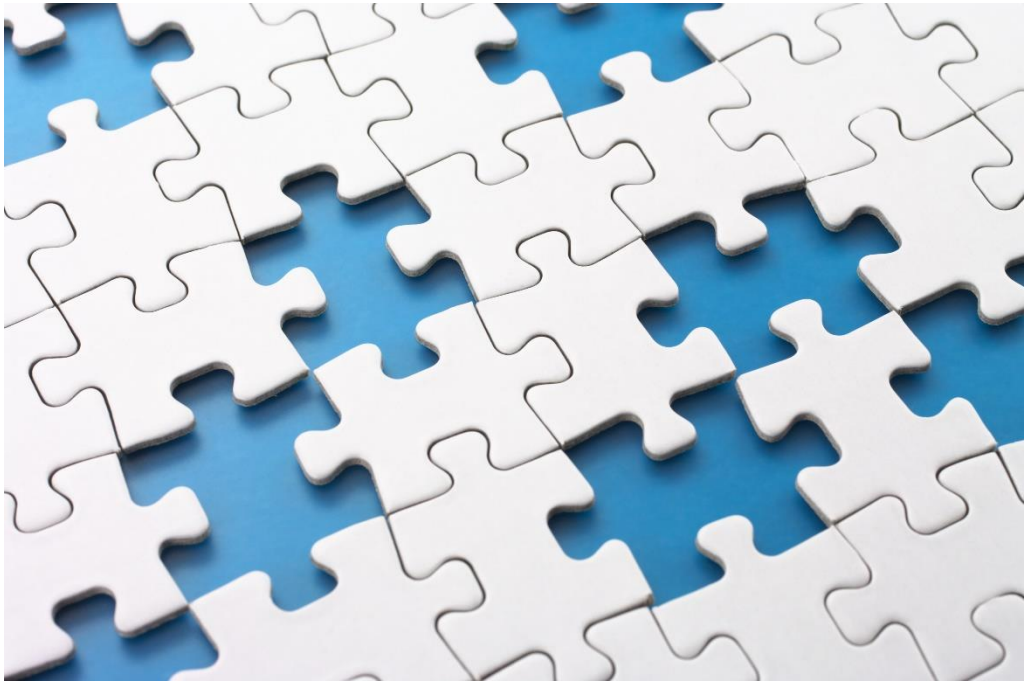
LUNARG

How is this funded?

LUNARG

The First Vulkan SDK

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

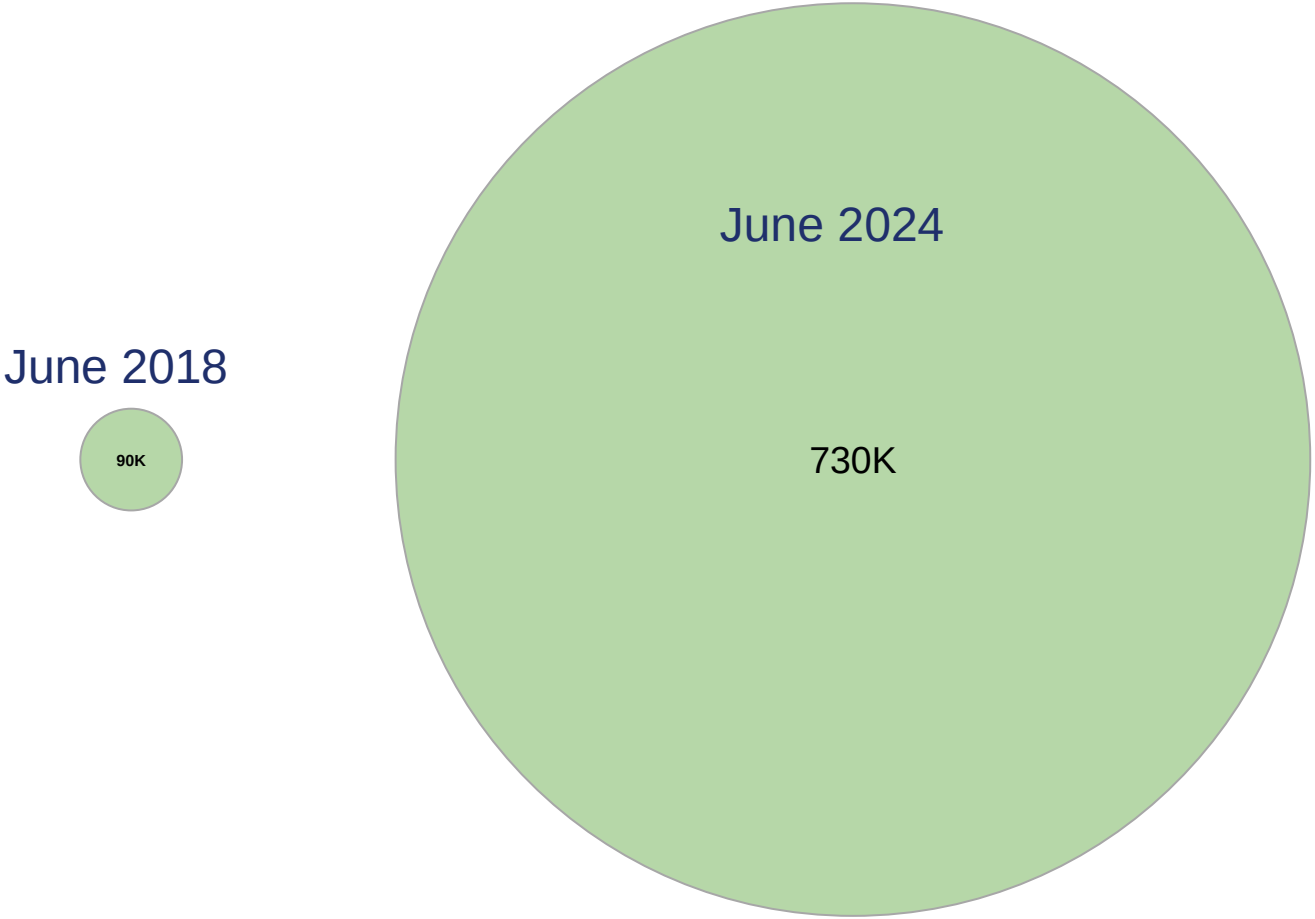


LUNARG

Validation Layer - Then and Now

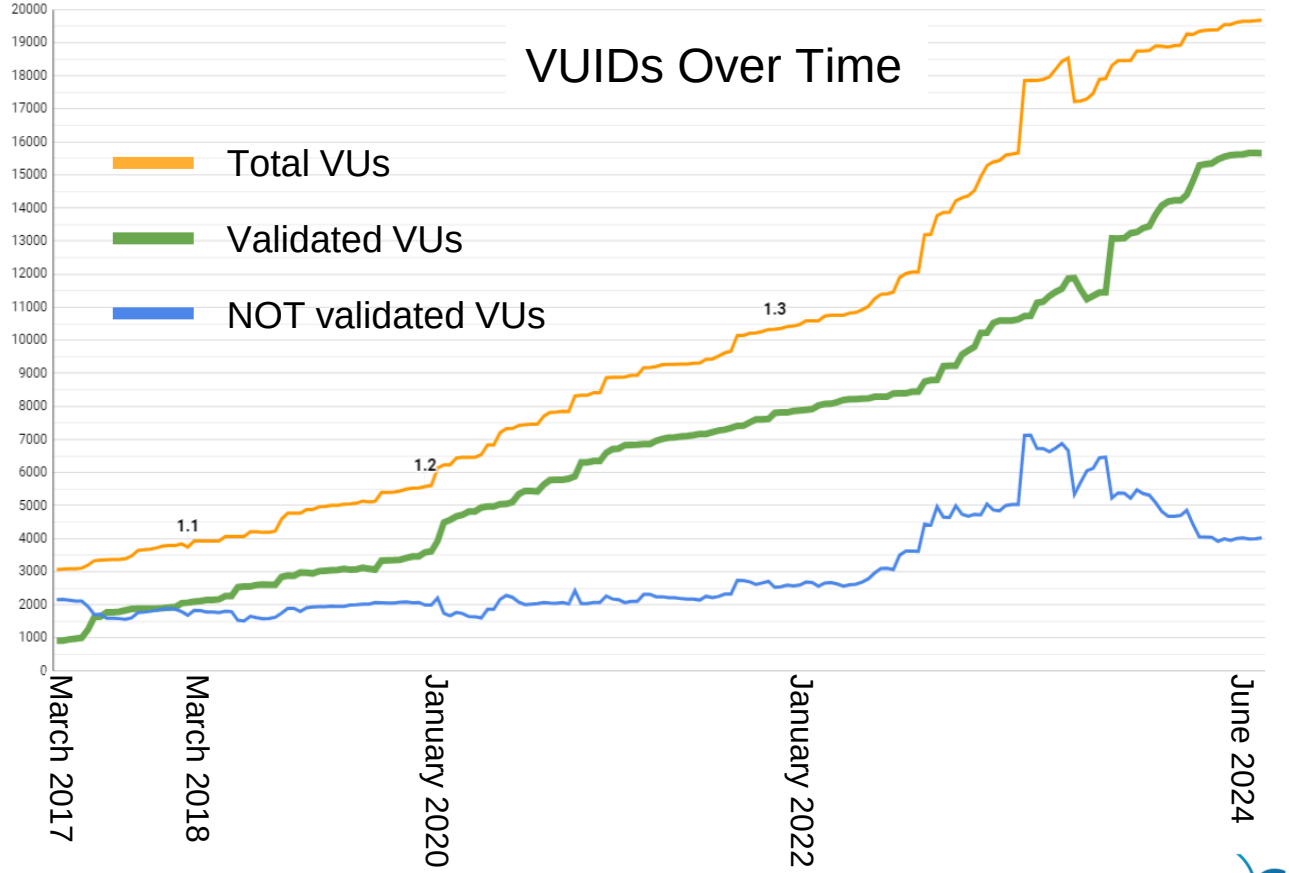
June 2018

Validation Layer - Then and Now



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”



LUNARG

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!

LUNARG

The Validation Layer - Today

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

LUNARG

The Validation Layer

We aren't done yet!
Vulkan API continues to evolve!

Opportunities Presented by the Technology

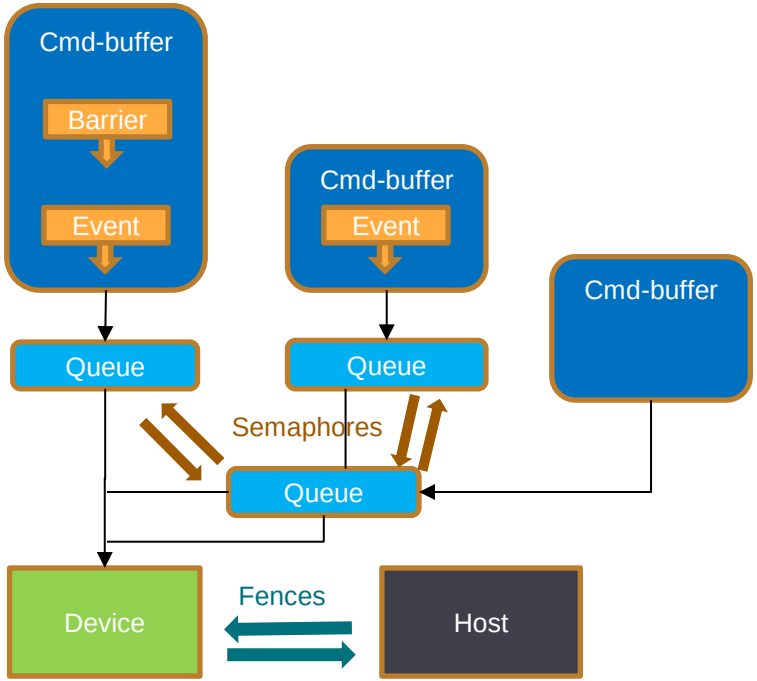
LUNARG

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



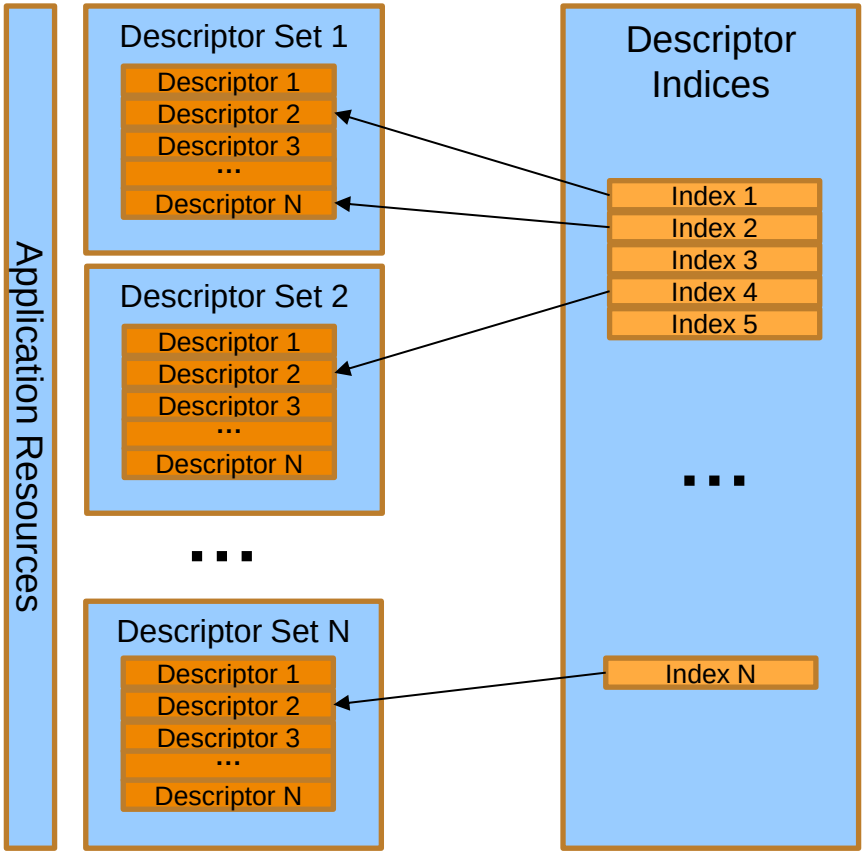
Validation Layer Improvement Opportunity:

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

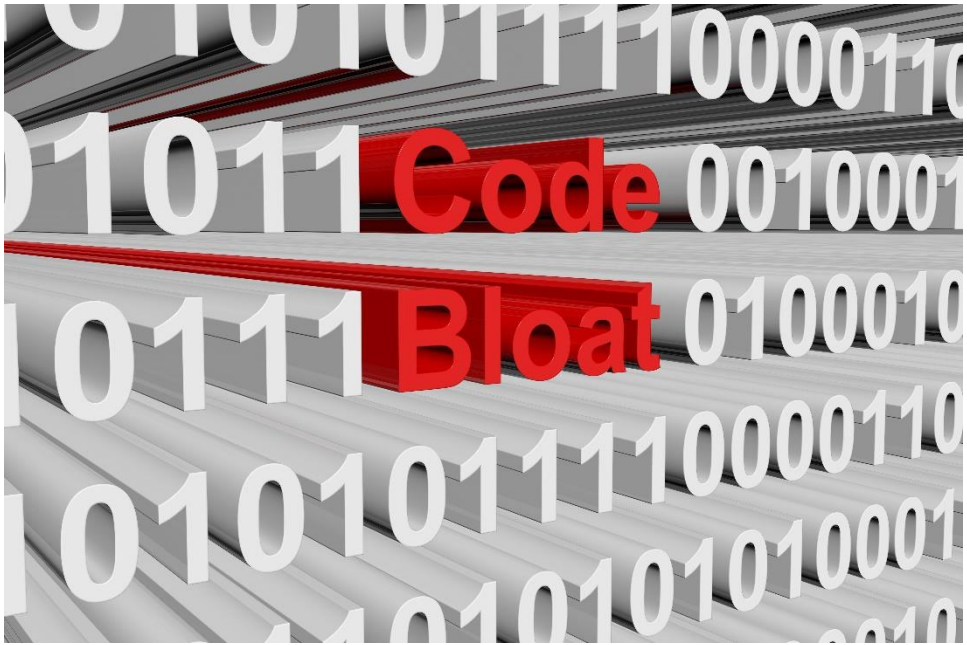
LUNARG

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

LUNARG

Validation Layer – Latency in Error Reporting

DELIVERY DELAY

- 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

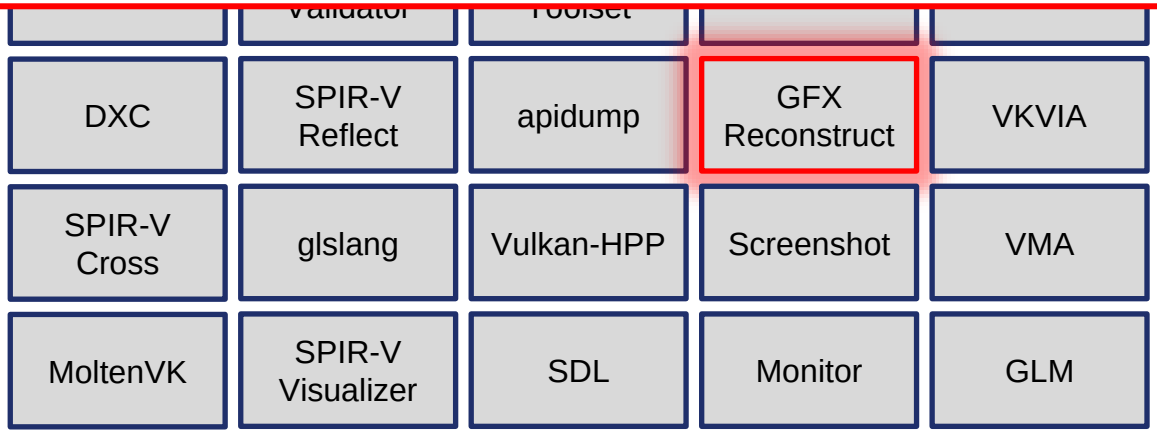
LUNARG

Open-source Vulkan Developer Tools

Included in the Vulkan SDK

GFXReconstruct - API Capture and Replay

- Cross-platform (Windows, Linux, Android, macOS)
- Run Vulkan workloads during GPU development
- Debug Vulkan applications
- Regression testing using real application workloads
- Underlying engine for profiling and debugging tools



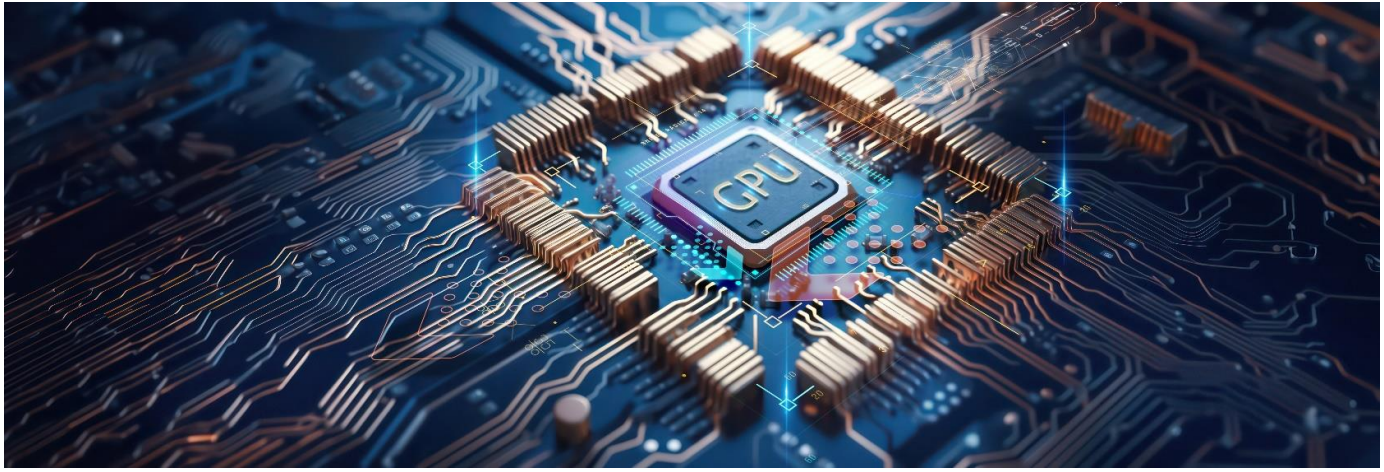
LUNARG

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

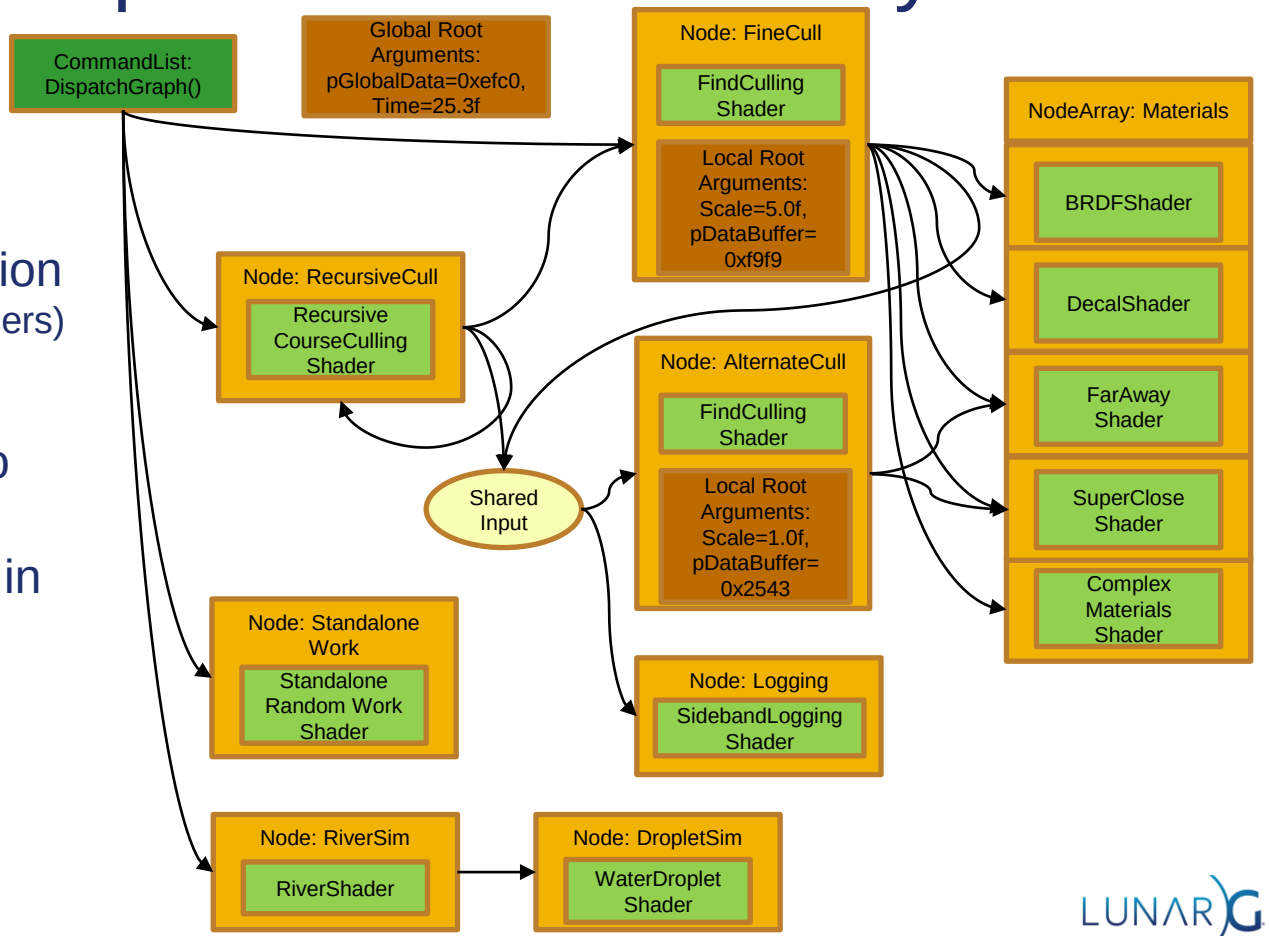
- GPUs - no longer "Graphics Processing Units"
  - Efficient processing of large blocks of data simultaneously
  - Compute - AI and ML
- Less Graphics API usage on the CPU
  - Rendering complexity still increasing via GPU driven rendering
- Many workloads moving to the GPU
  - Maximize utilization of GPU features
  - Reduce CPU interaction



LUNARG

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



Picture from MS D3D12 Work Graphs – DirectX Developer Blog. March 11, 2024

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!

LUNARG

LUNARG®

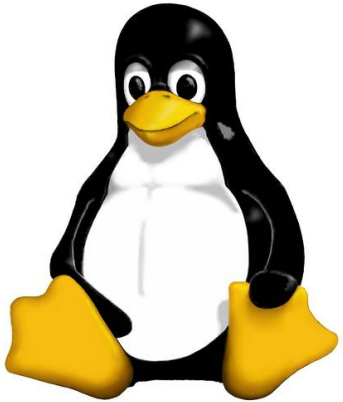

LUNARG®

Why Vulkan?

LUNARG

Why Vulkan? Cross-platform support

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

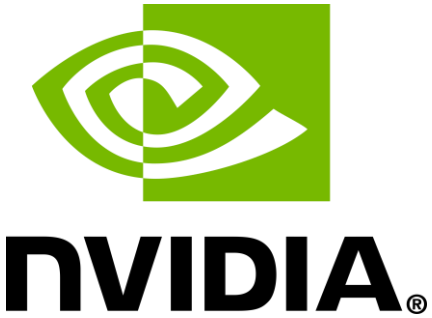


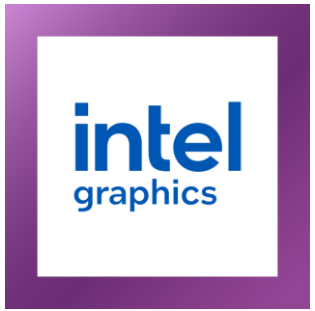
VIA

LUNARG

Why Vulkan? Improved Cross-vendor Compatibility

One API usage validator used by all (Vulkan-ValidationLayer)





LUNARG

Why Vulkan? Improved Performance

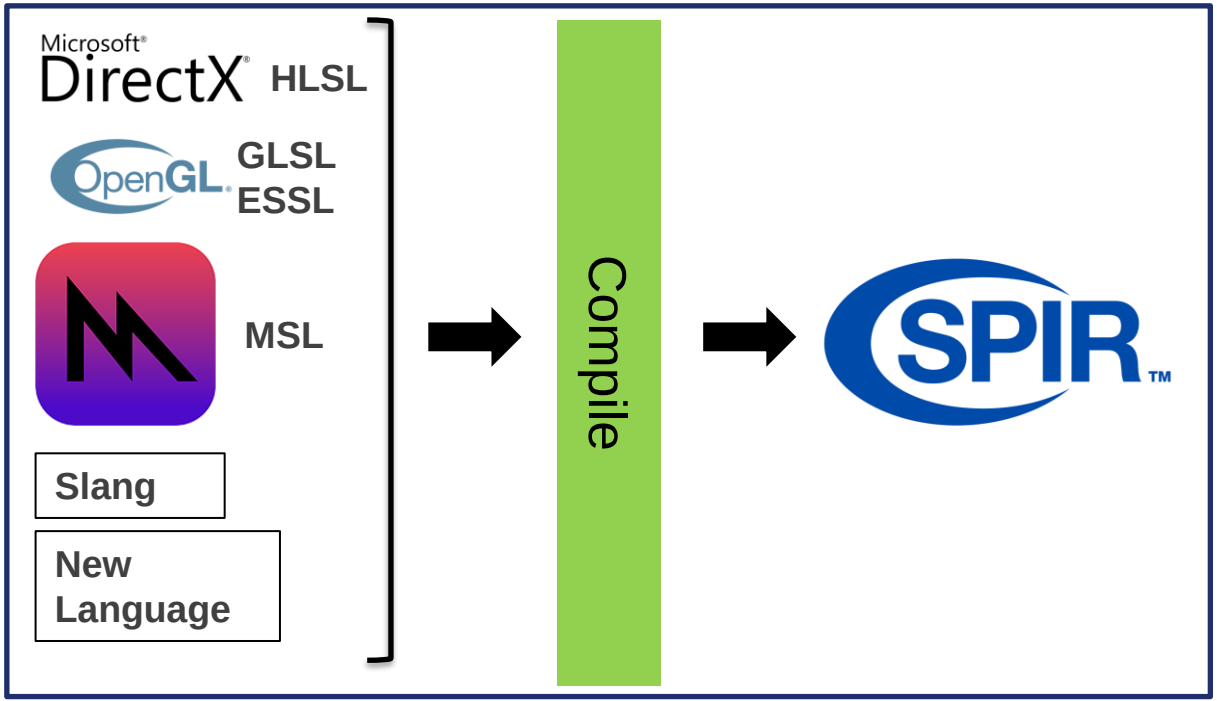
- Explicit application control over GPU and CPU workloads
- Multithreading-friendly API
- No more error checking in the Vulkan driver

LUNARG

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

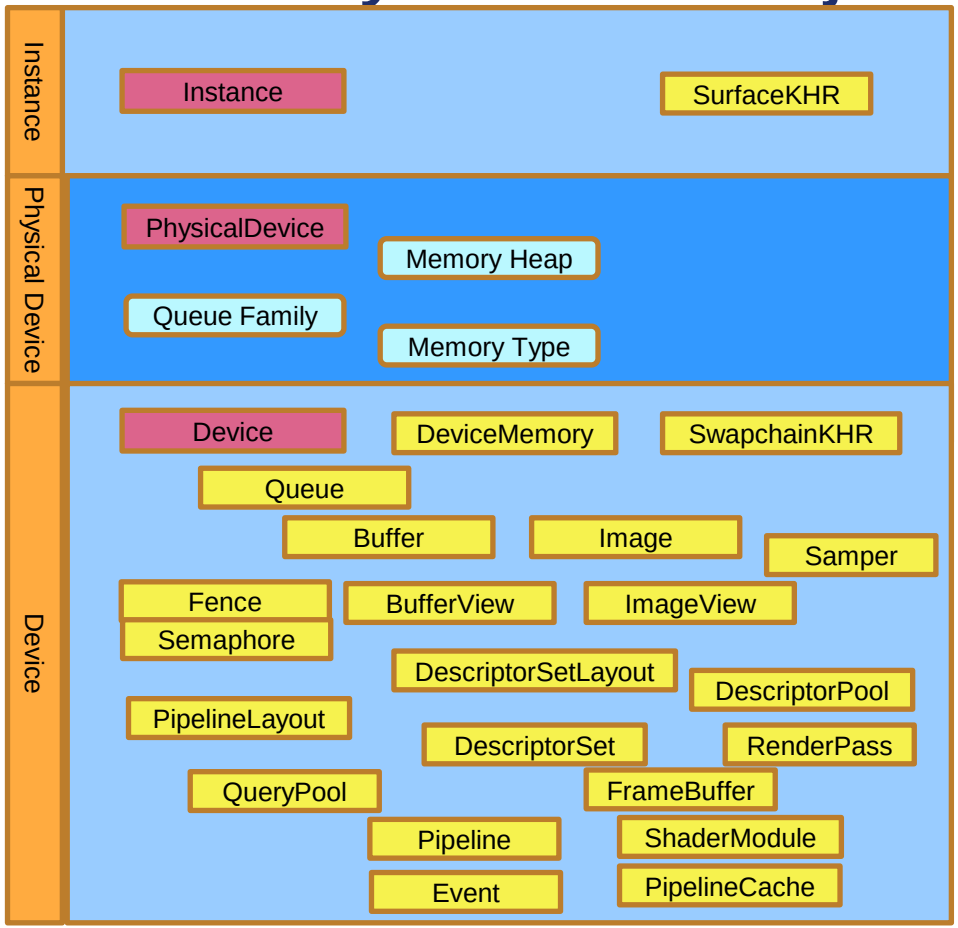
- Embrace and engage with the ISVs
- Open conformance test suite - more rigor
- More control put in the developer's hands

LUNARG

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

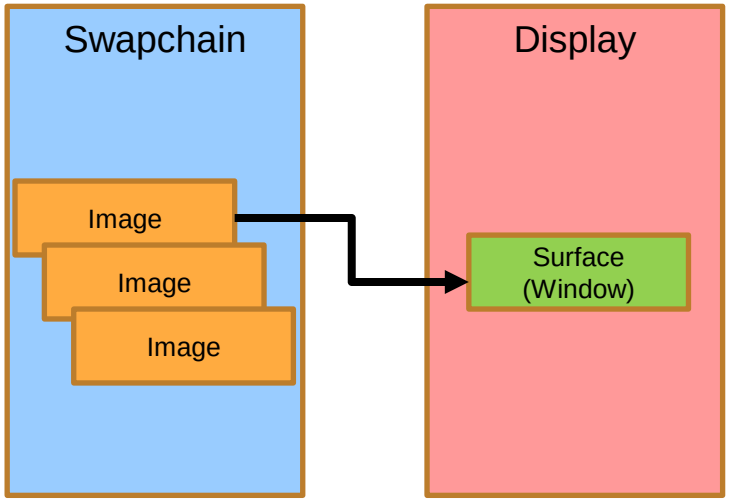
→ Complexity in the validation layer



LUNARG

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