The Vulkan SDK

From the Vulkan API Launch to Today

Karen Ghavam
CEO and Engineering Director
LunarG, Inc

LUNARG

Today's Talk

- Creation of the Vulkan SDK
- Vulkan API and the Vulkan Developer Tools
- Creation of Present Day LunarG

LUNARG

Important Context

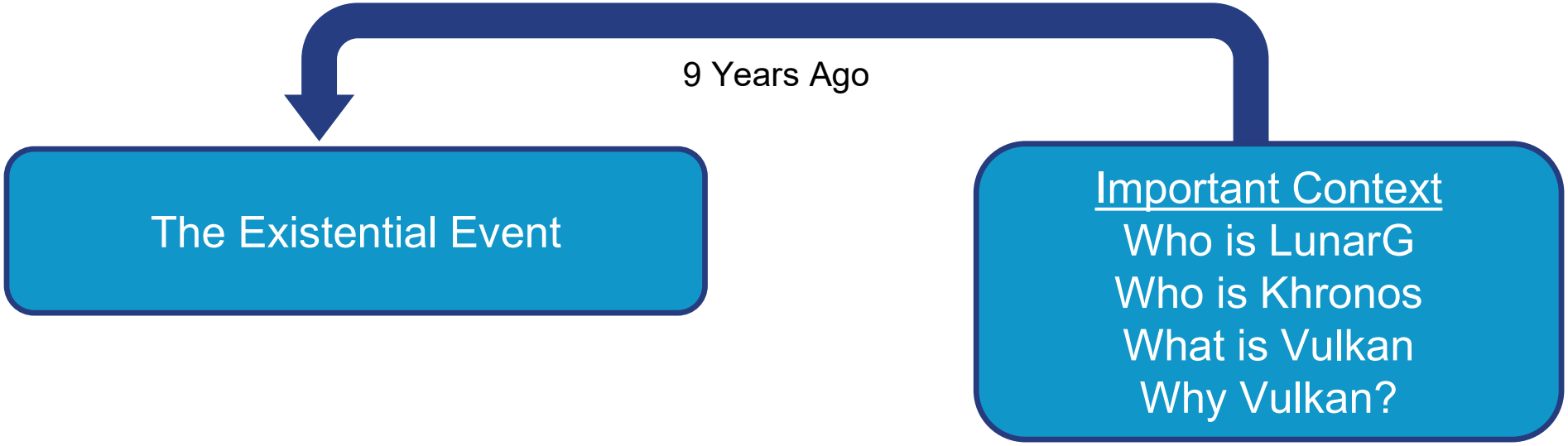
Who is LunarG

Who is Khronos

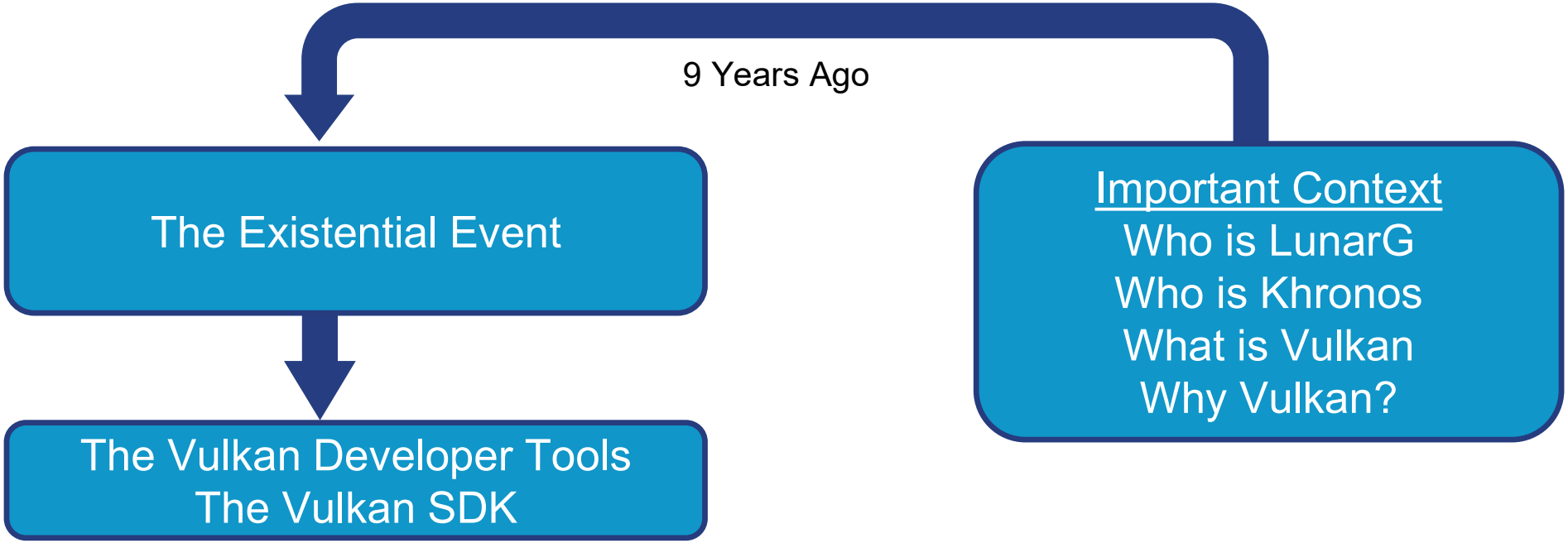
What is Vulkan

Why Vulkan?

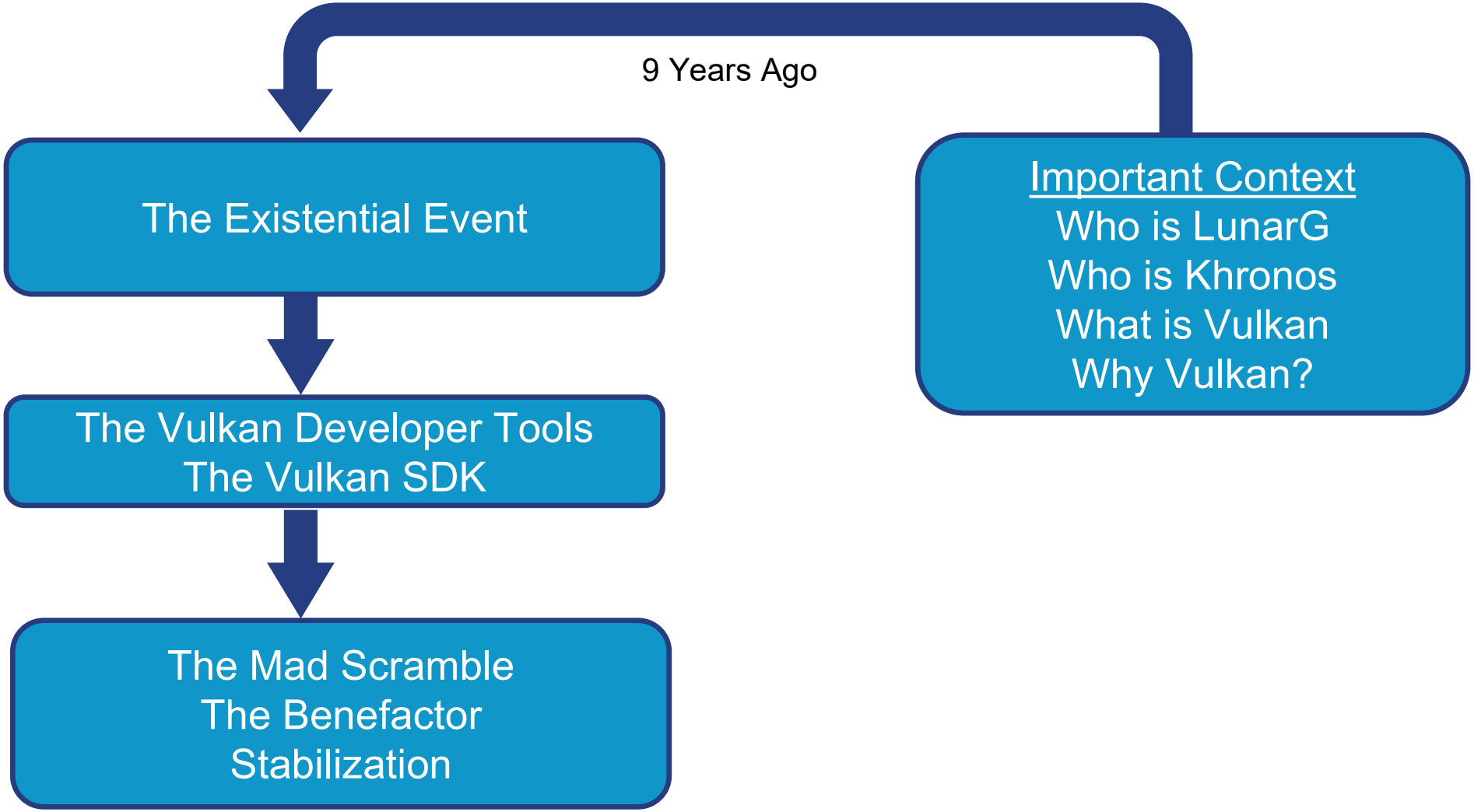
LUNARG


9 Years Ago
The Existential Event
Important Context
Who is LunarG
Who is Khronos
What is Vulkan
Why Vulkan?

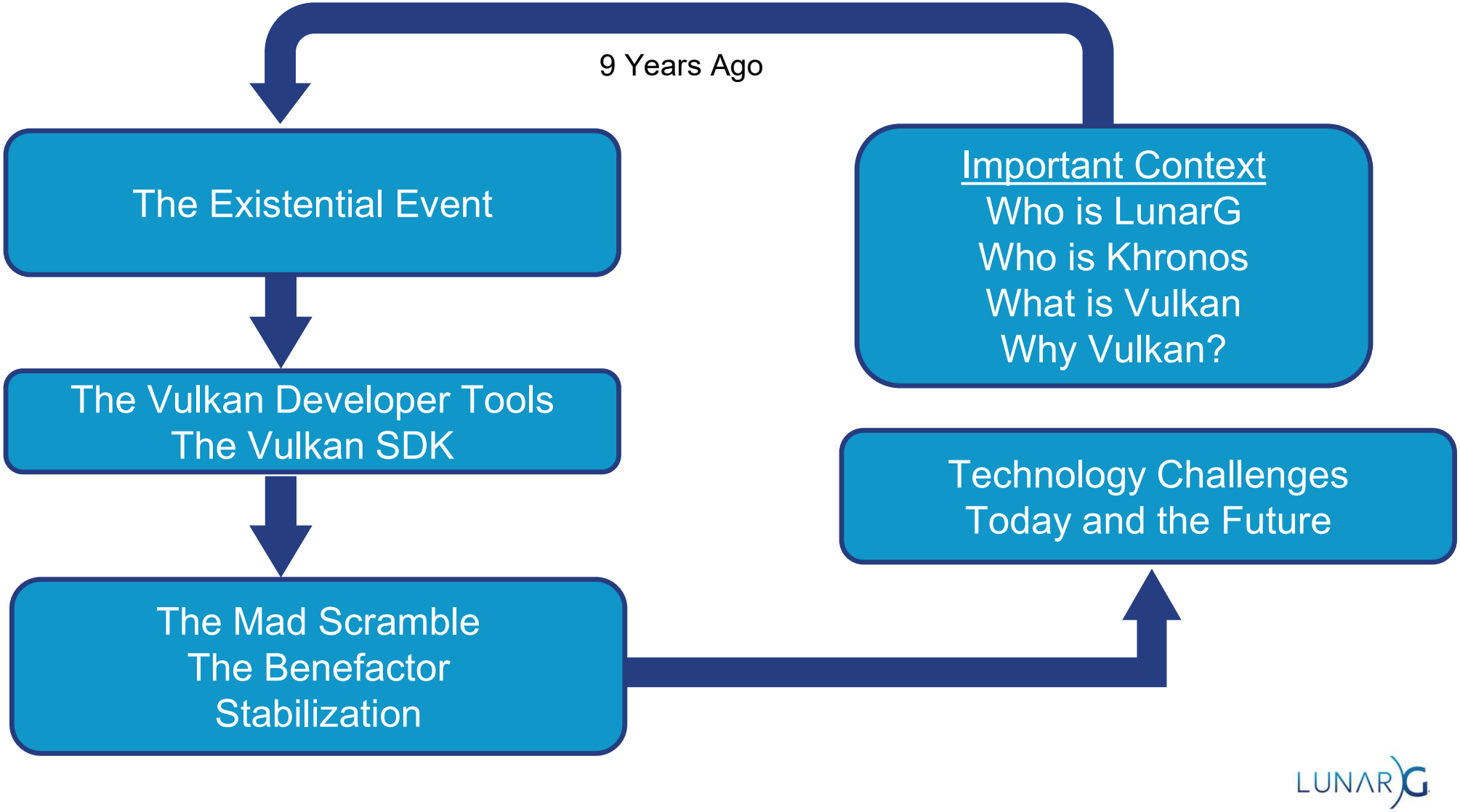
LUNARG


9 Years Ago
The Existential Event
The Vulkan Developer Tools
The Vulkan SDK
Important Context
Who is LunarG
Who is Khronos
What is Vulkan
Why Vulkan?

LUNARG


9 Years Ago
The Existential Event
The Vulkan Developer Tools
The Vulkan SDK
The Mad Scramble
The Benefactor
Stabilization
Important Context
Who is LunarG
Who is Khronos
What is Vulkan
Why Vulkan?

LUNARG


9 Years Ago
The Existential Event
The Vulkan Developer Tools
The Vulkan SDK
The Mad Scramble
The Benefactor
Stabilization
Important Context
Who is LunarG
Who is Khronos
What is Vulkan
Why Vulkan?
Technology Challenges
Today and the Future

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

What is Vulkan?

- Cross-platform, Cross-vendor Graphics and Compute API
  - PCs, consoles, mobile phones, embedded platforms
- Vulkan API Specification Created by the Khronos Group
  - Member driven consortium for the creation and maintenance of open standards
    - GPU vendors
    - Platform vendors
    - SoC Integrators
    - Tool developers
    - Game Engine developers
    - …
- Low level and explicit API
  - Specification in essence defines a GPU

LUNARG

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

First Vulkan Proof of Concept

- 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

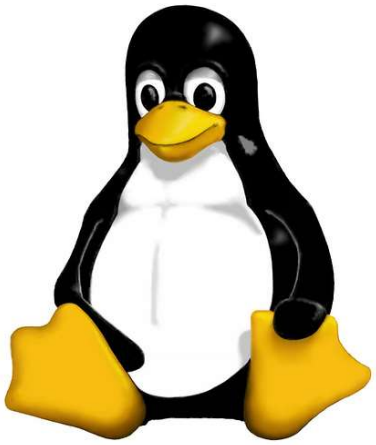
LUNARG

Why Vulkan?

LUNARG

Why Vulkan? Cross-platform support

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

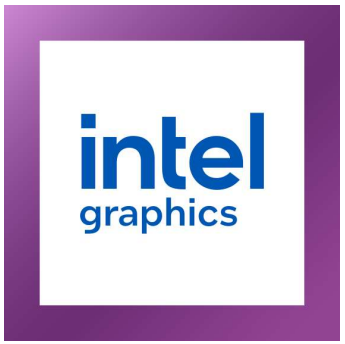


VIA

LUNARG

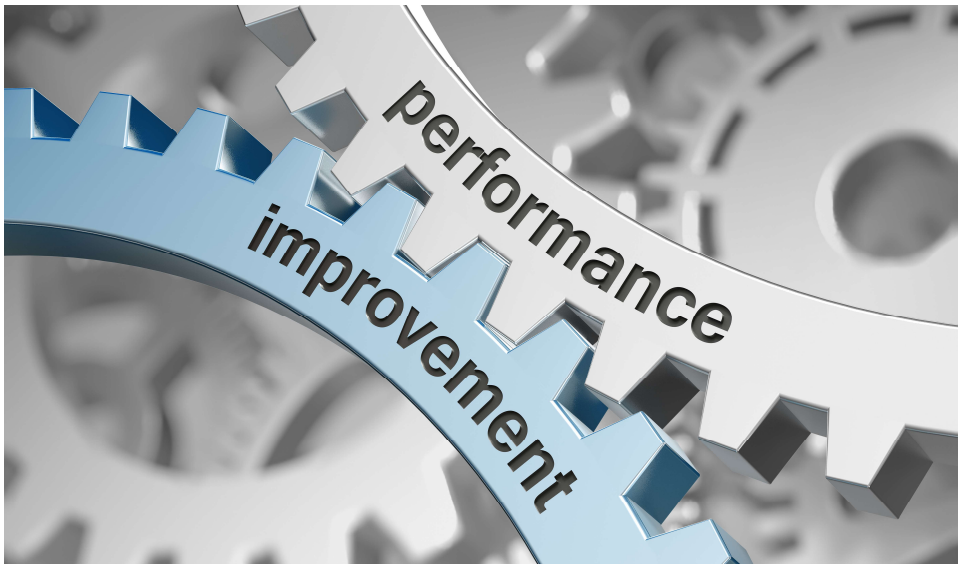
Why Vulkan? Improved Cross-vendor Compatibility

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



LUNARG

Why Vulkan? Improved Performance



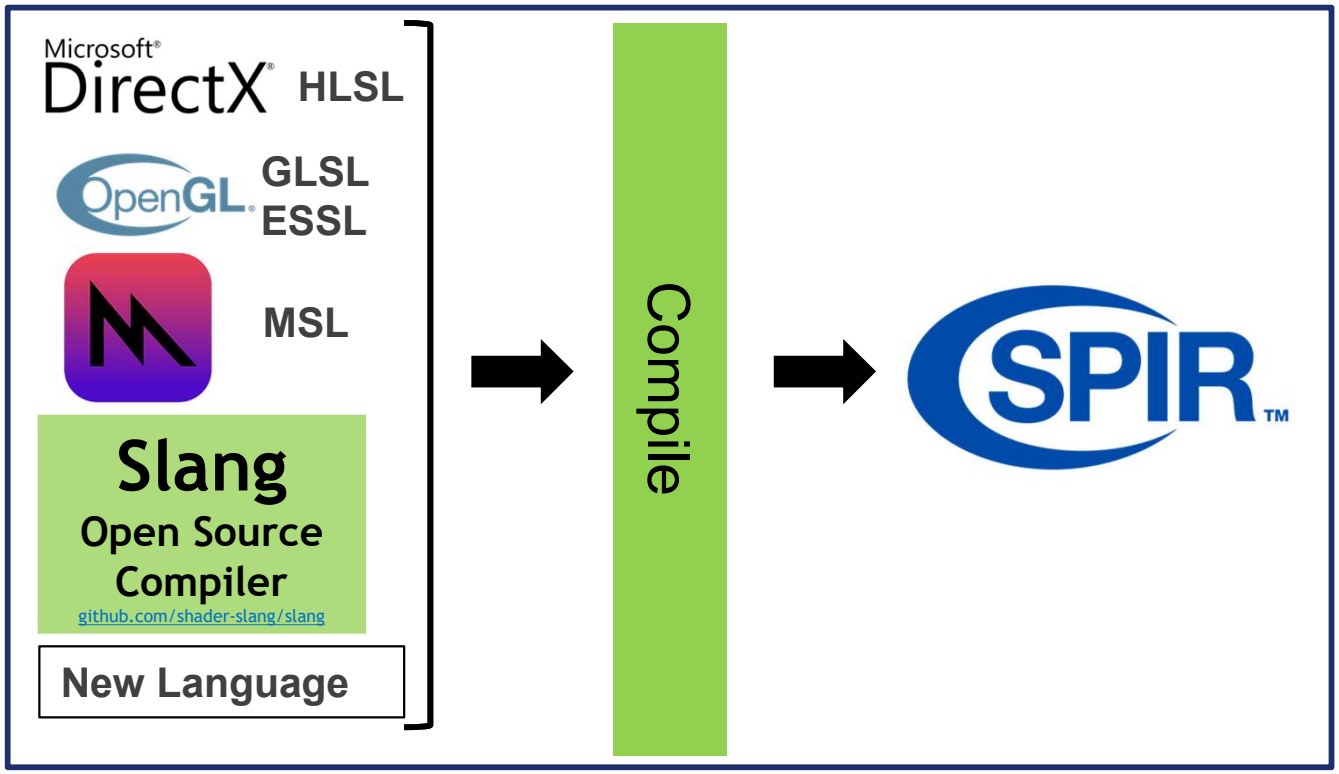
- Explicit application control over GPU and CPU workloads
- Multithreading-friendly API
- No more error checking in the Vulkan driver

LUNARG

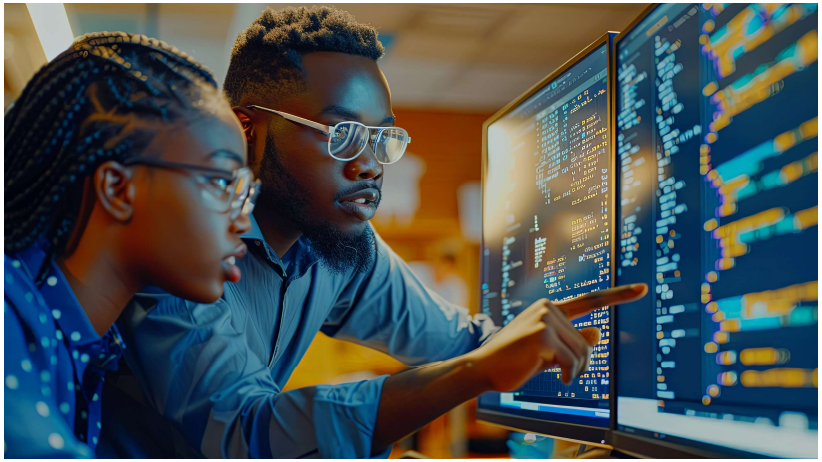
Why Vulkan? Shader Language Flexibility

Standardized Intermediate Language (SPIR-V)

- Eliminates front-end compilers from drivers
  - Reduce driver complexity
- Front-end language flexibility
  - Improve portability



Why Vulkan? Open Standard

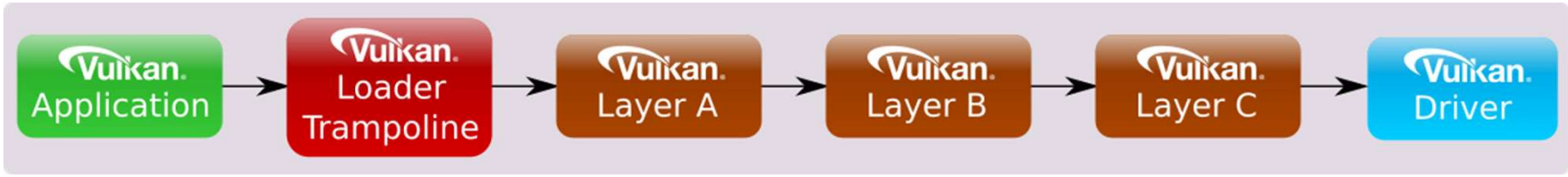


Strengthened ecosystem focus

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

LUNARG

Vulkan is a Layered Architecture

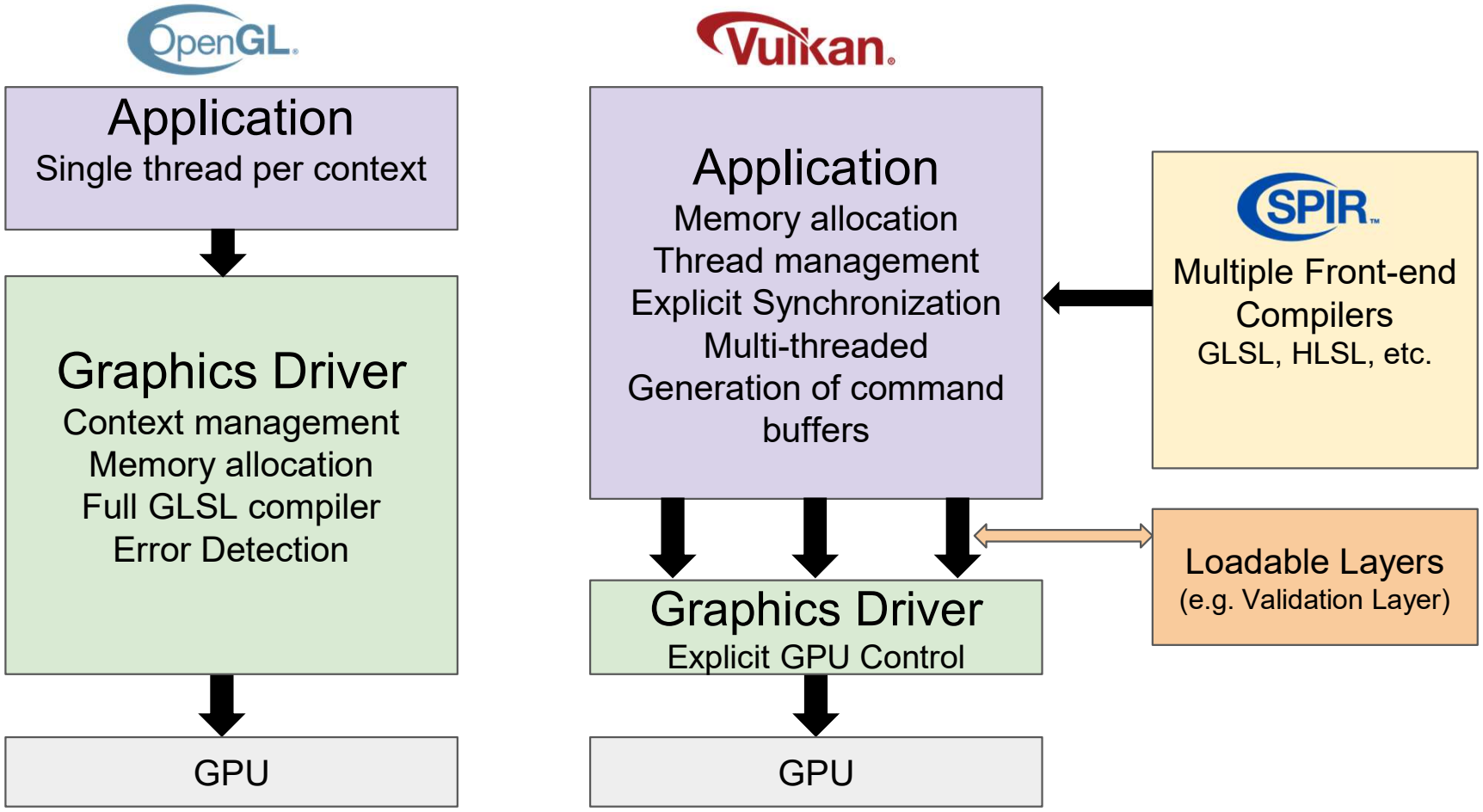


API calls work their way through the loader, layers, and driver in order

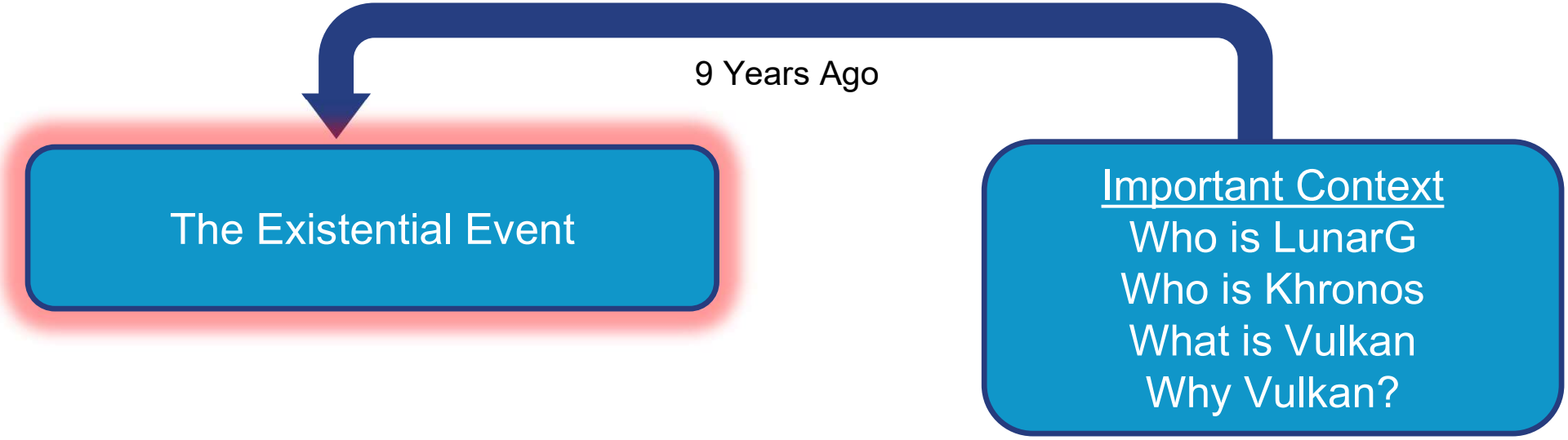
- Vulkan Loader
  - Library that finds and loads Drivers & Layers
- Vulkan Layer
  - “Plugin interface to the Vulkan API”
  - Intercepts Vulkan API calls made by applications
  - Enables mechanism for valuable cross-vendor debugging tools

LUNARG

OpenGL vs. Vulkan



LUNARG



The Forming of Present-day LunarG

LUNARG

When did "The Vulkan Journey" start for Karen?

- Labor Day Weekend, 2015.
  - "Off the Grid"
  - Upon return, it all started…
- "Do you want to coast to retirement, or go out with a bang?“
- October 12, 2015 – First day at LunarG

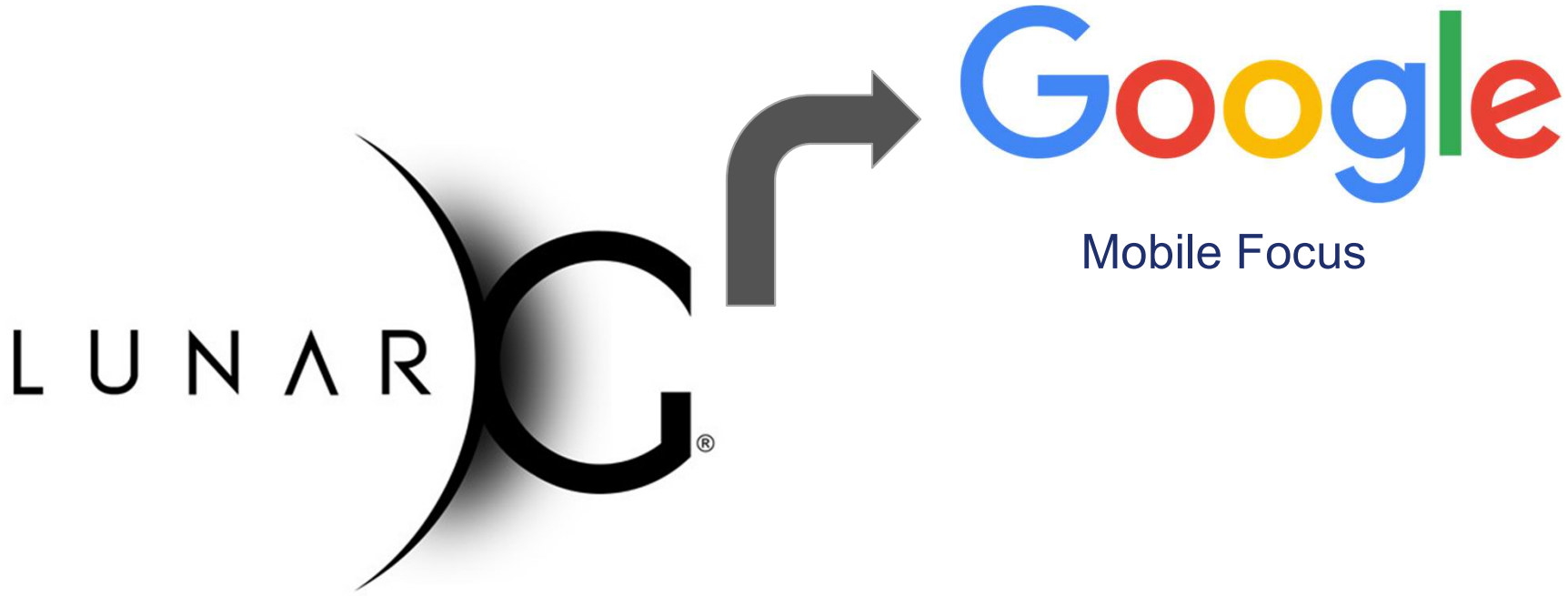
The Existential Event

- Turmoil 3 months before the public launch!

LUNARG

The Existential Event

- Turmoil 3 months before the public launch!



The Existential Event

- Turmoil 3 months before the public launch!

Google

Mobile Focus

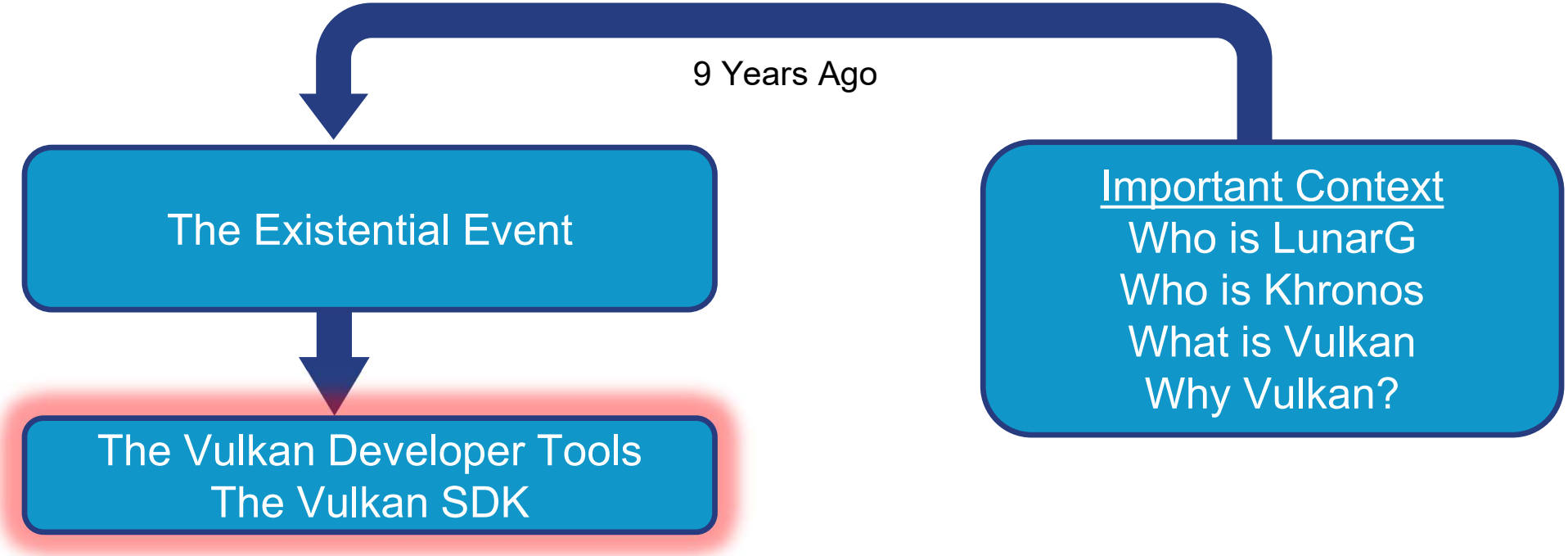
LUNARG

Desktop Focus

LUNARG

The Existential Event

Engineering Team cut in half!!



Developer Tools and the Vulkan SDK

LUNARG

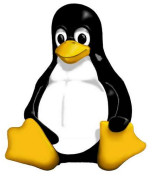
Why Developer Tools?

- While the Vulkan API specification is absolutely necessary…
  - It isn't sufficient for the success of the Vulkan API

- Application developers need debugging tools!

Open-source Vulkan Developer Tools

Included in the Vulkan SDK

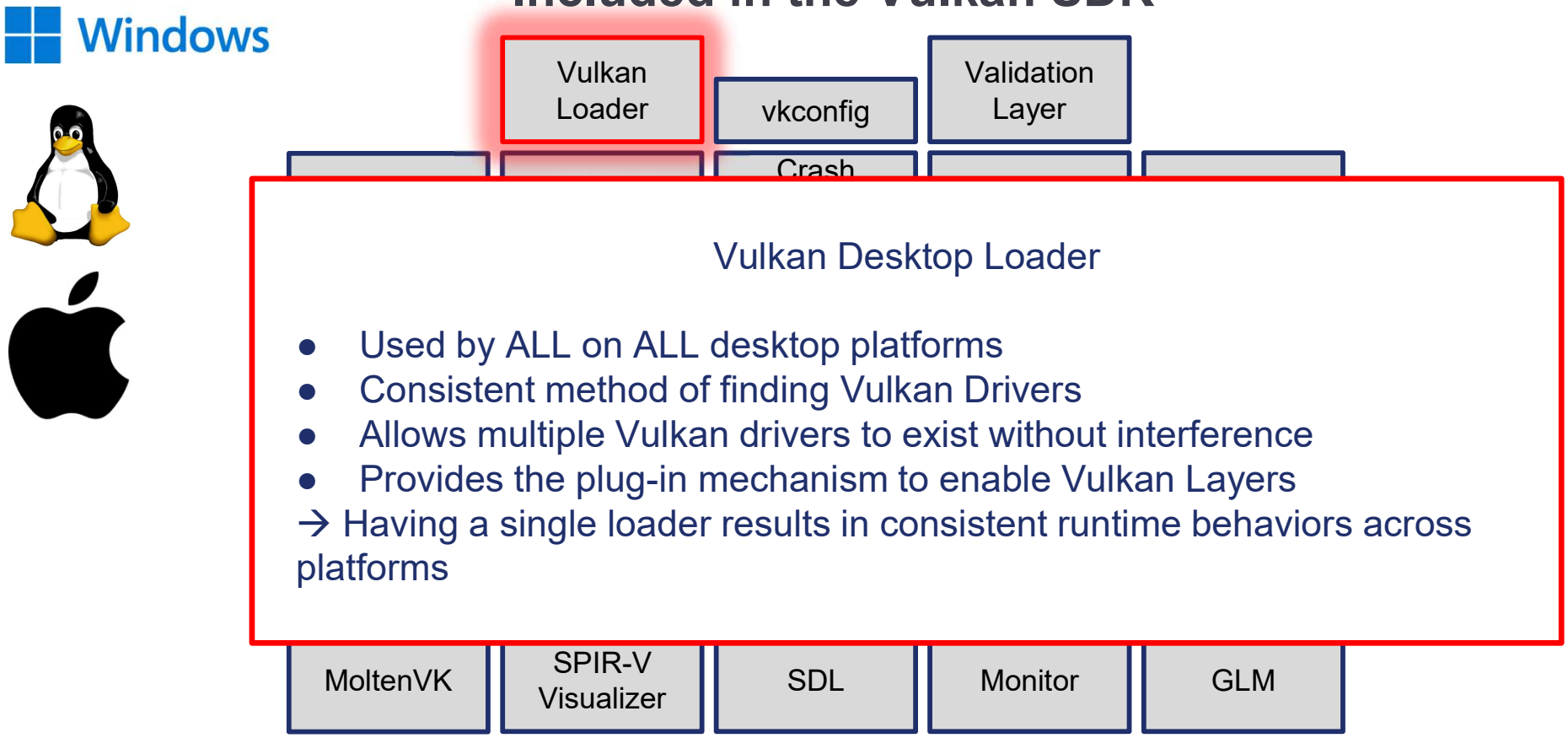


	Vulkan Loader	vkconfig	Validation Layer	
SPIR-V Optimizer	SPIR-V Tools	Crash Diagnostic Layer	vulkaninfo	Emulation Layers
shaderc	SPIR-V Validator	Profiles Toolset	GPUInfo	VOLK
DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKVIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL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

LUNARG

Open-source Vulkan Developer Tools

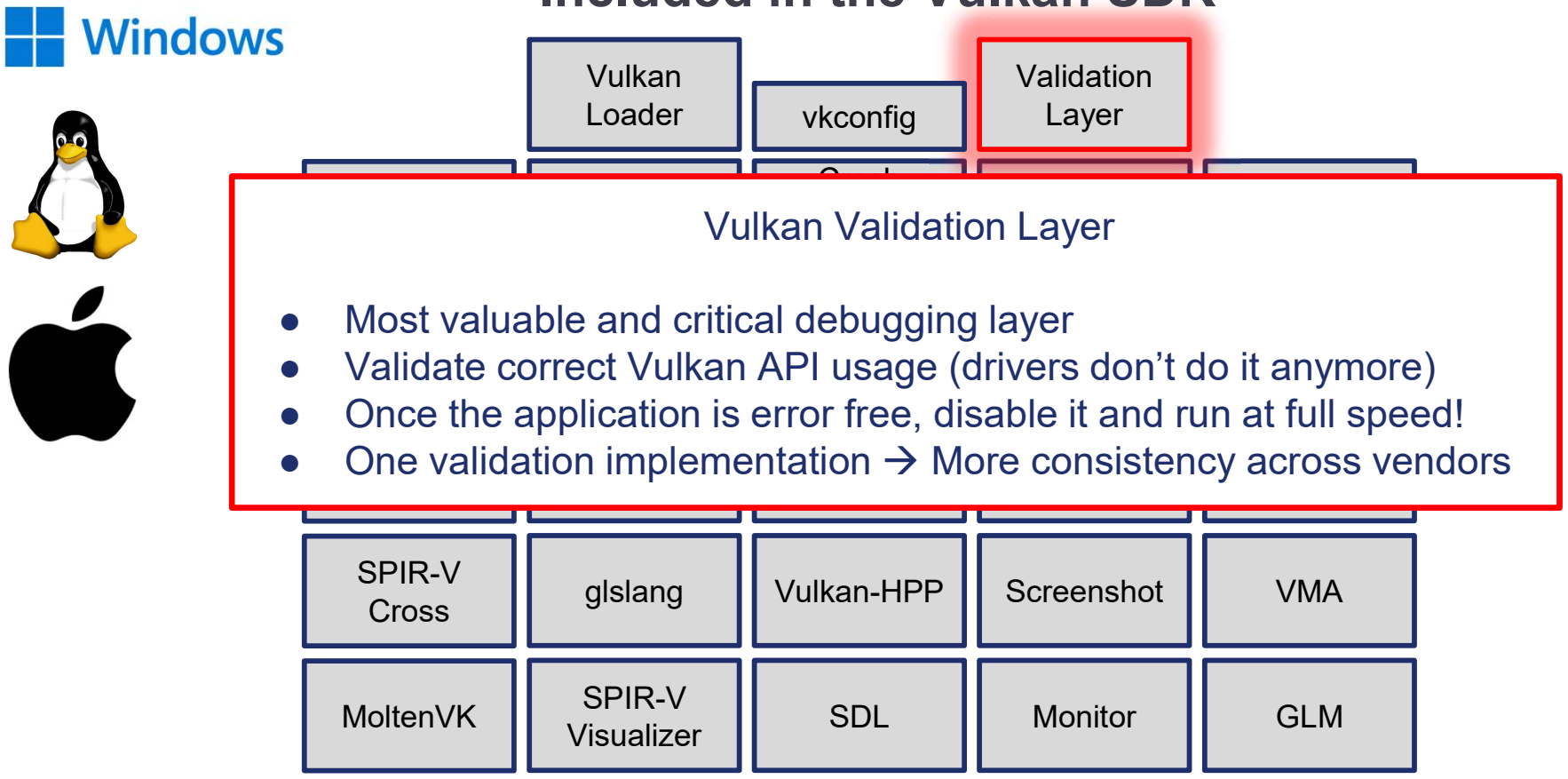
Included in the Vulkan SDK



LUNARG

Open-source Vulkan Developer Tools

Included in the Vulkan SDK

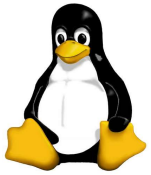


LUNARG

Open-source Vulkan Developer Tools

Included in the Vulkan SDK

Windows



	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
	SPIR-V	idump	GFX Reconstruct	VKVIA
		an-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

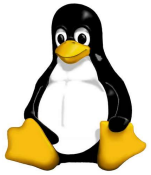
MoltenVK
- Bringing Vulkan to Apple

LUNARG

Open-source Vulkan Developer Tools

Included in the Vulkan SDK

Windows



GFXReconstruct - API Capture and Replay

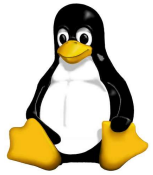
- Cross-platform (Windows, Linux, Android, macOS)
- Run Vulkan workloads during GPU development
- Debug Vulkan applications
- Regression testing using real application workloads
- Underlying engine for profiling and debugging tools

DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKVIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

LUNARG

Open-source Vulkan Developer Tools

Included in the Vulkan SDK



	Vulkan Loader	vkconfig	Validation Layer	
SPIR-V Optimizer	SPIR-V Tools	Crash Diagnostic Layer	vulkaninfo	Emulation Layers

Crash Diagnostic Layer

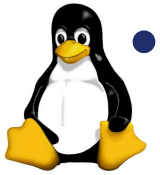
- Track down and identify the cause of GPU hangs and crashes
- Instruments command buffers with completion checkpoints
- Get a dump file
- Strong user demand. Debugging Device Lost errors very difficult!

MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

LUNARG

The Vulkan SDK

Windows



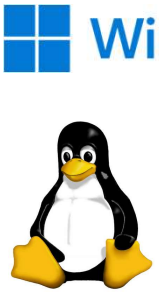
- Benefits
  - Pre-built
  - Curated
  - Integrated
  - System Installation
  - vkconfig ready for use
  - License Registry

	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

Delivered by LunarG in close coordination with the Khronos Vulkan working group

LUNARG

The Vulkan SDK



Windows On arm

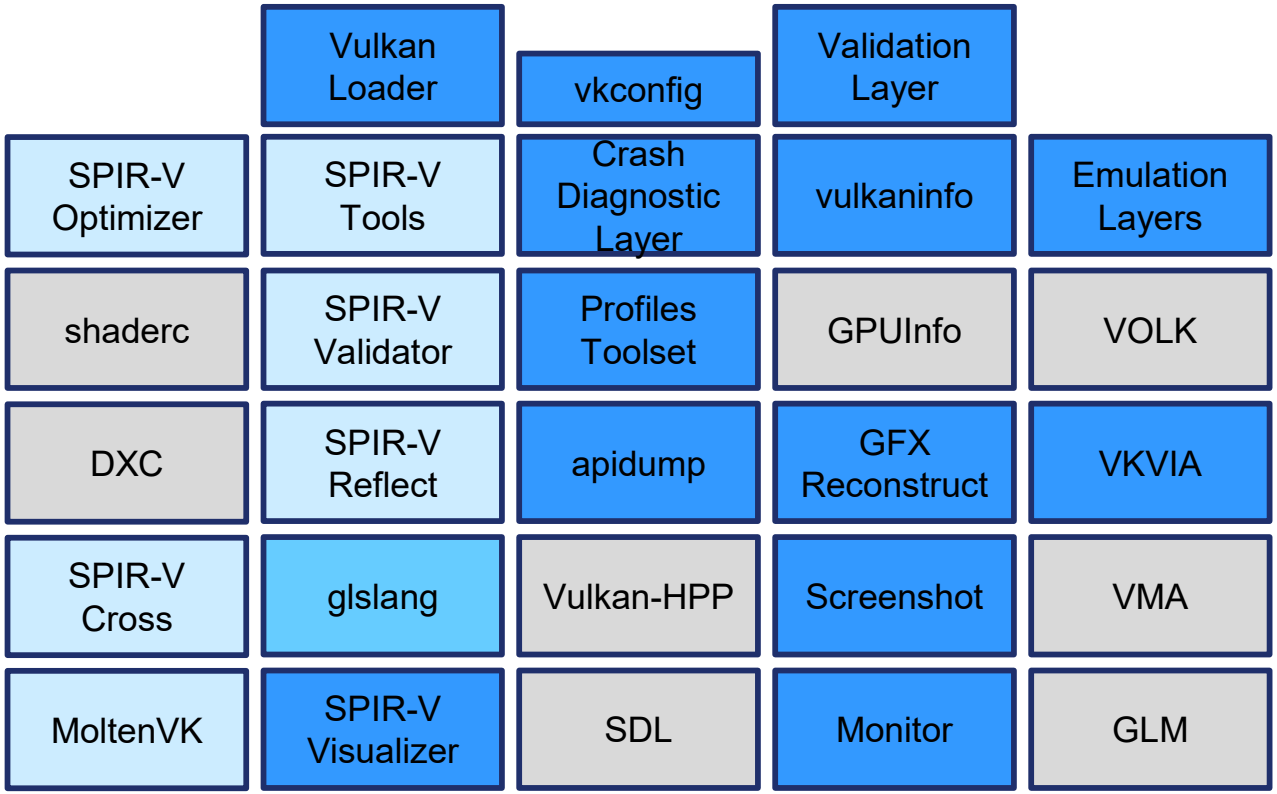
	Vulkan Loader	vkconfig	Validation Layer	
SPIR-V Optimizer	SPIR-V Tools	Crash Diagnostic Layer	vulkaninfo	Emulation Layers
shaderc	SPIR-V Validator	Profiles Toolset	GPUInfo	VOLK
DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKVIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL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

Delivered by LunarG in close coordination with the Khronos Vulkan working group

LUNARG

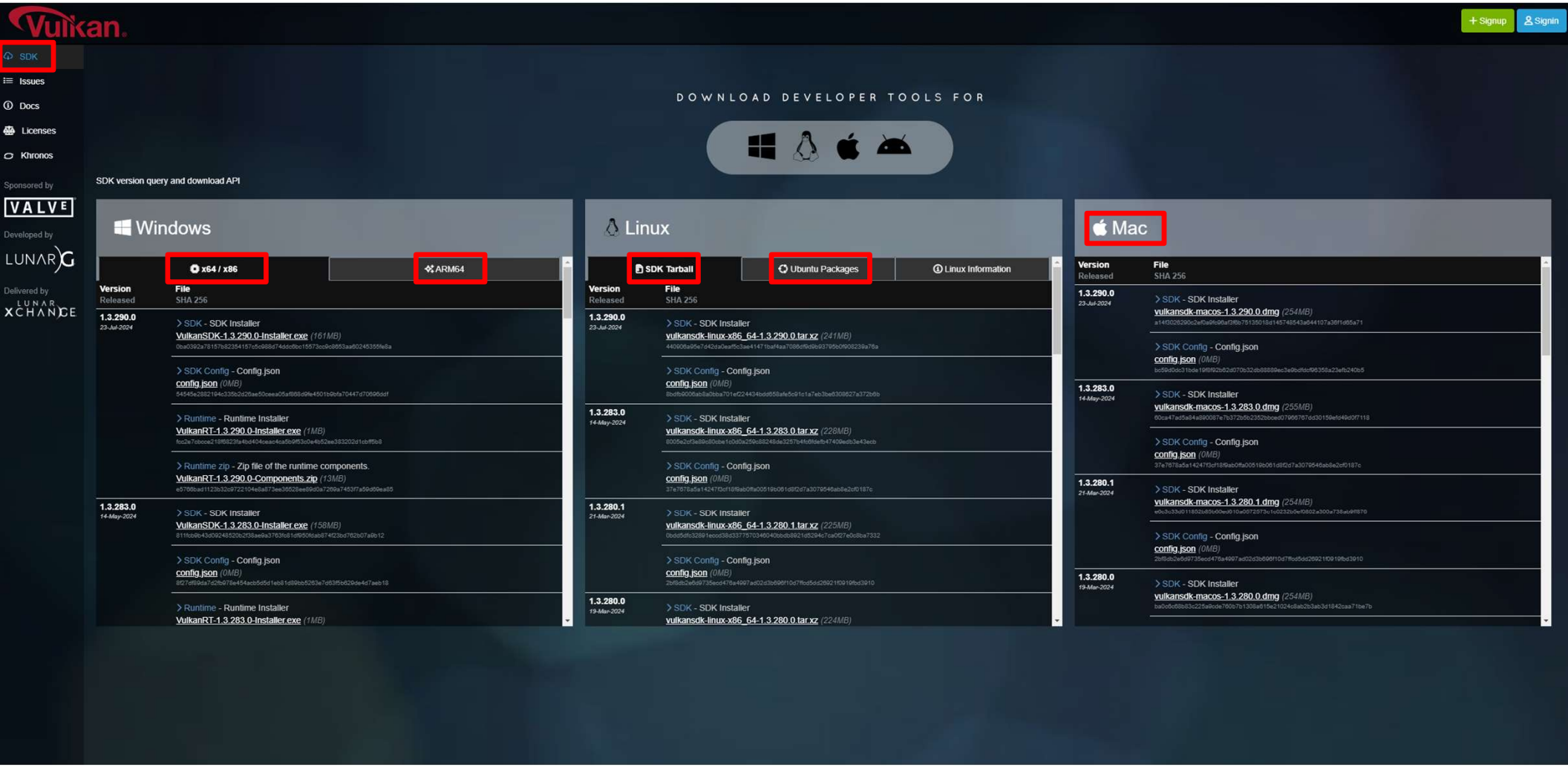
The Vulkan SDK

- LunarG Ownership
  - Initial creation
  - Ongoing enhancement and maintenance
- LunarG - contributor
- LunarG - Maintainer

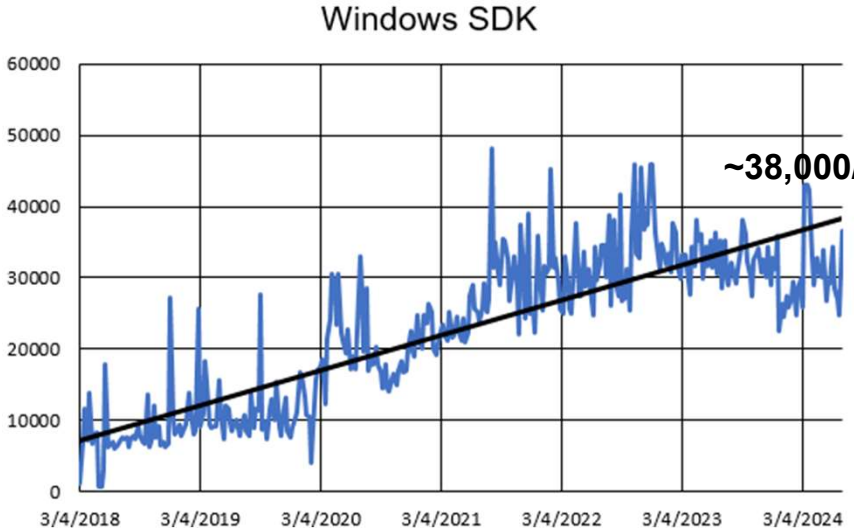


LUNARG

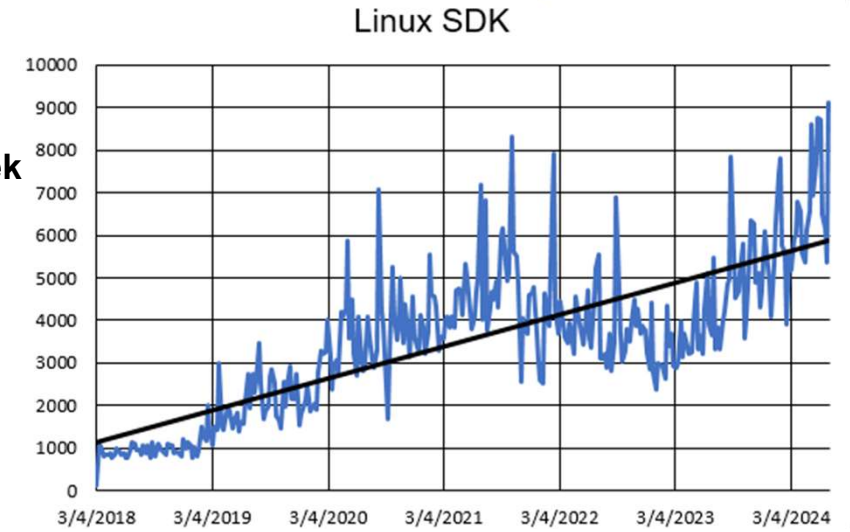
Vulkan SDK Download Page (vulkan.lunarg.com)



Vulkan SDK Downloads are Healthy

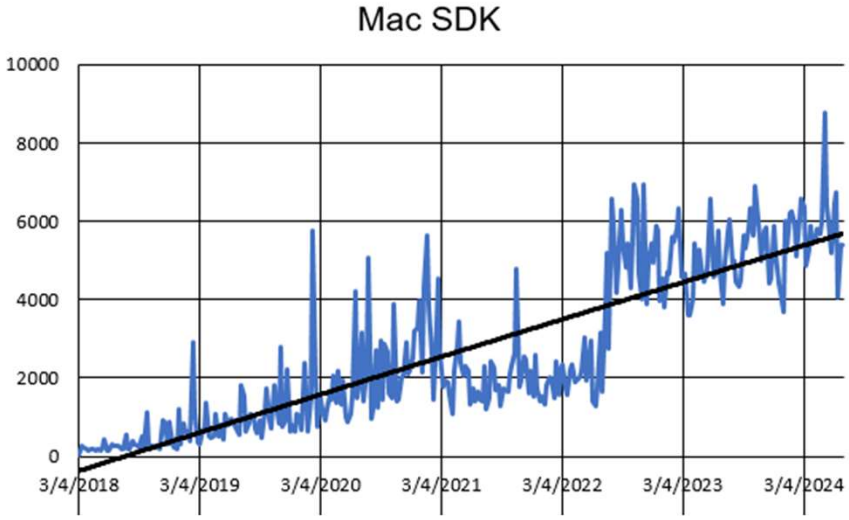


~38,000/week



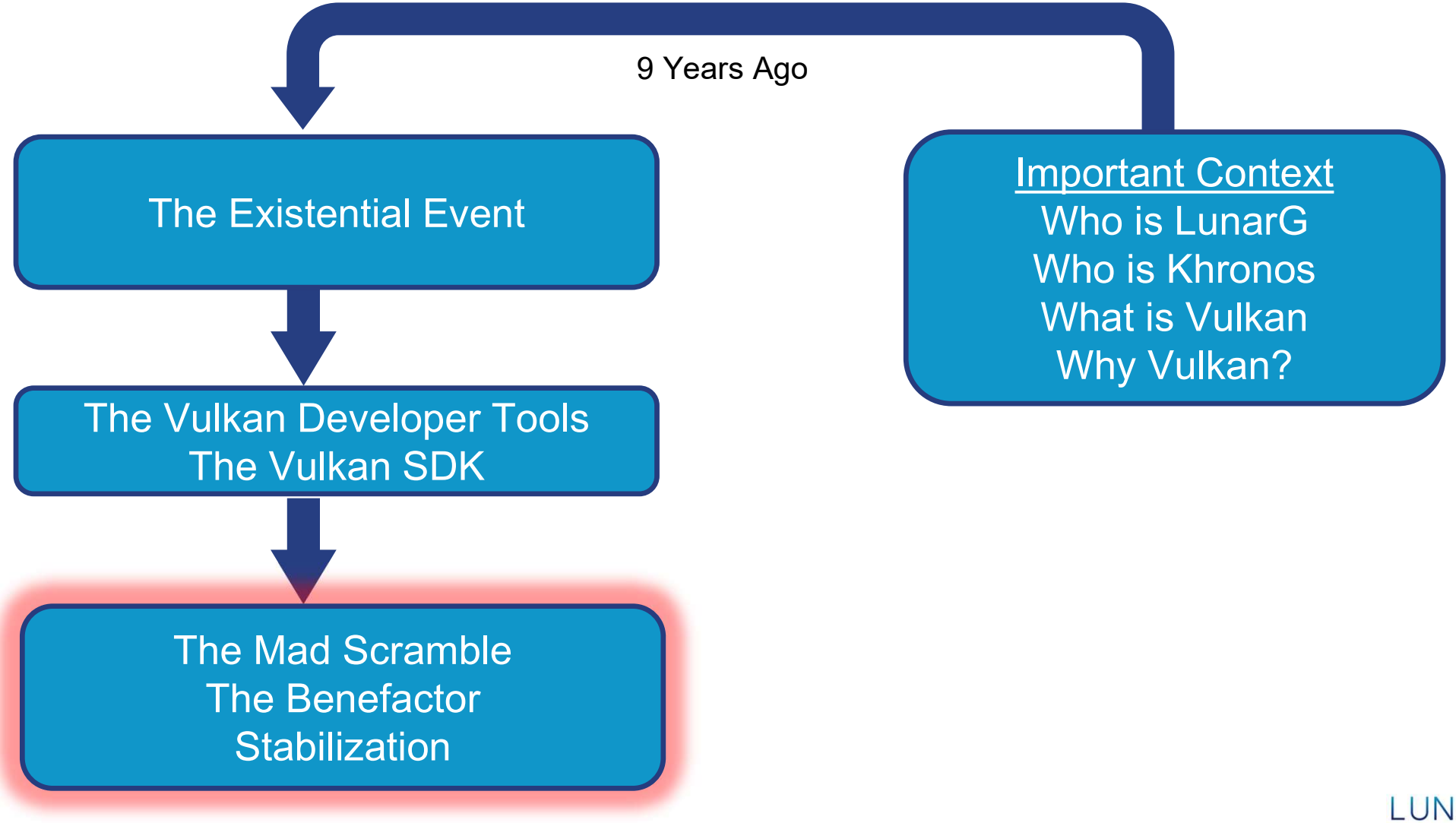
~6000/week

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



~5800/week

LUNARG


9 Years Ago
The Existential Event
The Vulkan Developer Tools
The Vulkan SDK
The Mad Scramble
The Benefactor
Stabilization
Important Context
Who is LunarG
Who is Khronos
What is Vulkan
Why Vulkan?

LUNARG

The Mad Scramble

- Half the engineering team
- Deliver an SDK in 3 months
  - Vulkan Loader
  - Vulkan Validation Layer
- Launch the SDK download site

The Mad Scramble

The Good News

- The engineers knew what they were doing

The Bad News

- The challenge in front of us!

LUNARG

The Vulkan API Launch - February 16, 2016

- Coordinated Launch
  - Khronos Vulkan 1.0 API specification
  - First Vulkan SDK

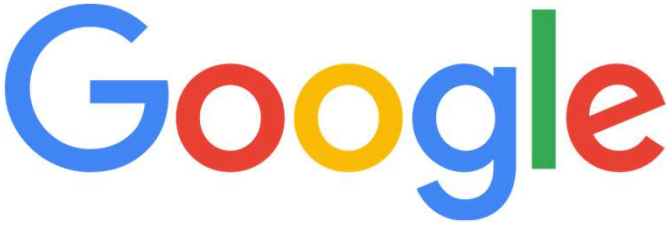
LUNARG

How is this funded?

How is this funded?

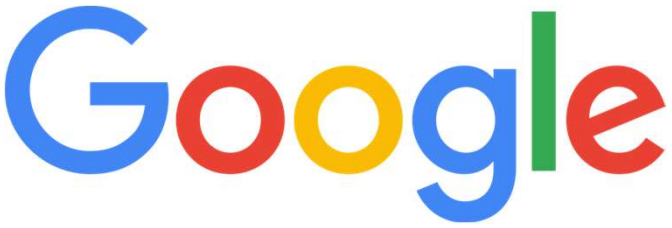
LUNARG

How is this funded?



LUNARG

How is this funded?



LUNARG

Why Did We Succeed in the Beginning?

A Meaningful Purpose

- Our work matters, and will have a positive and broad industry impact

Why Did We Succeed in the Beginning?

<u>A Meaningful Purpose</u>

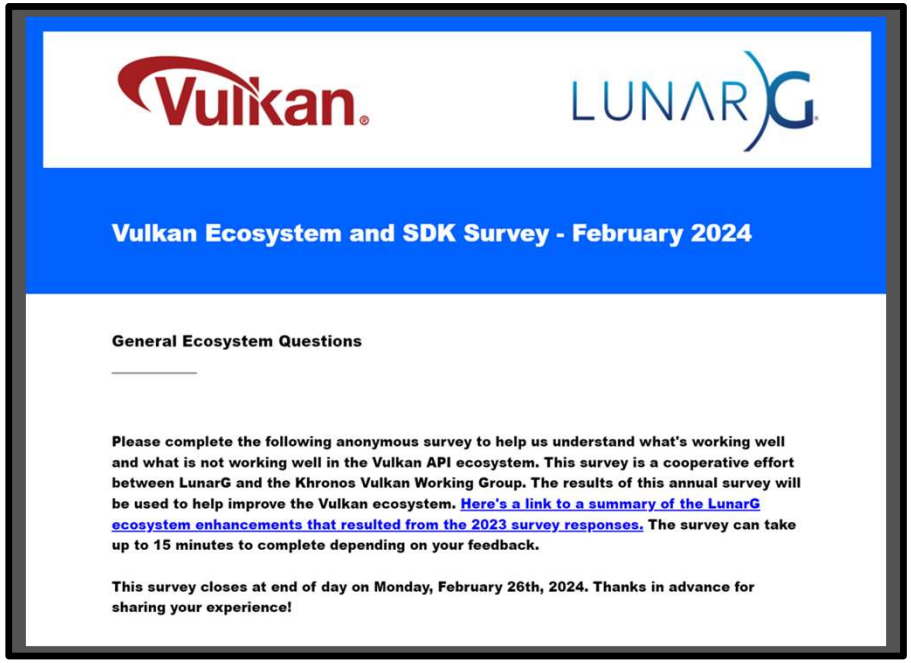
- Our work matters, and will have a positive and broad industry impact
- The Generosity of the Vulkan Ecosystem Benefactor

VALVE

LUNARG

Stabilization

- Strong Vulkan API adoption as a low-level standard
- More companies actively participating in building the ecosystem
  - Enabling benefits for ALL
- Listening to the Vulkan application developers
  - Yearly LunarG Developer Survey
  - Accountability to the developers



LUNARG

Stabilization

- A team of 3D graphics SW experts excited about the vision
  - Talented, skilled, enthusiastic
  - Naturally attracts the right people for the job

LUNARG

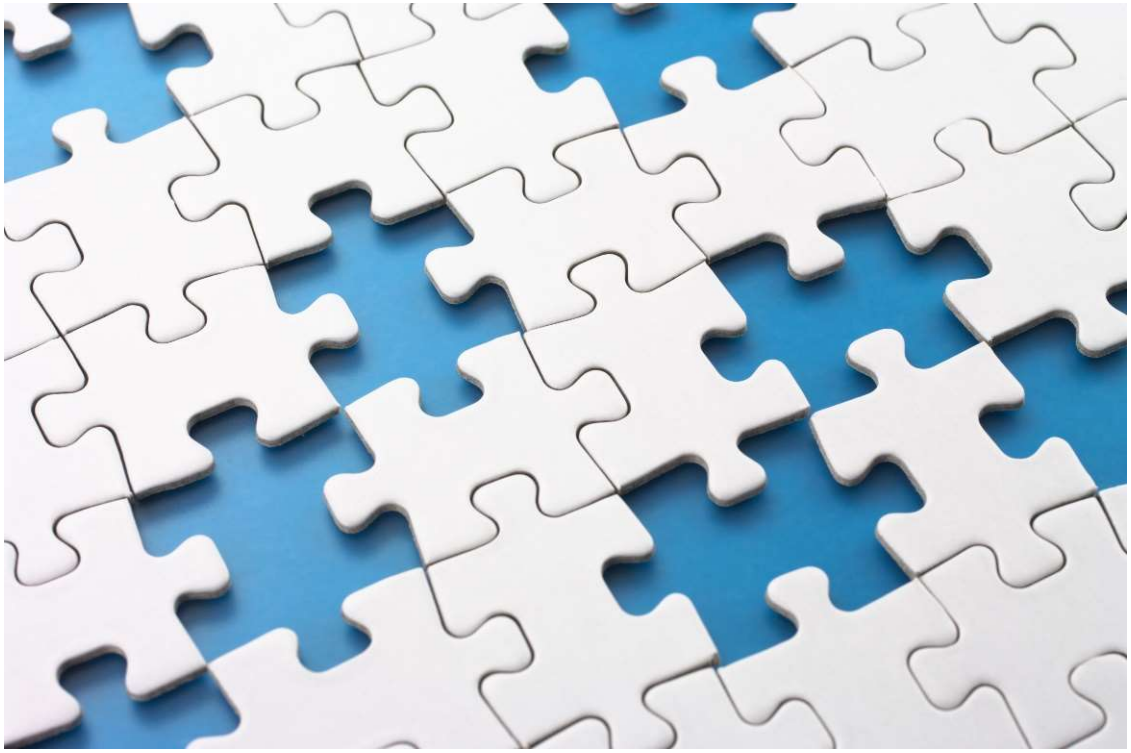
Stabilization

- A team of 3D graphics SW experts excited about the vision
  - Talented, skilled, enthusiastic
  - Naturally attracts the right people for the job

- And the LunarG purpose continues!

The First Vulkan SDK

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



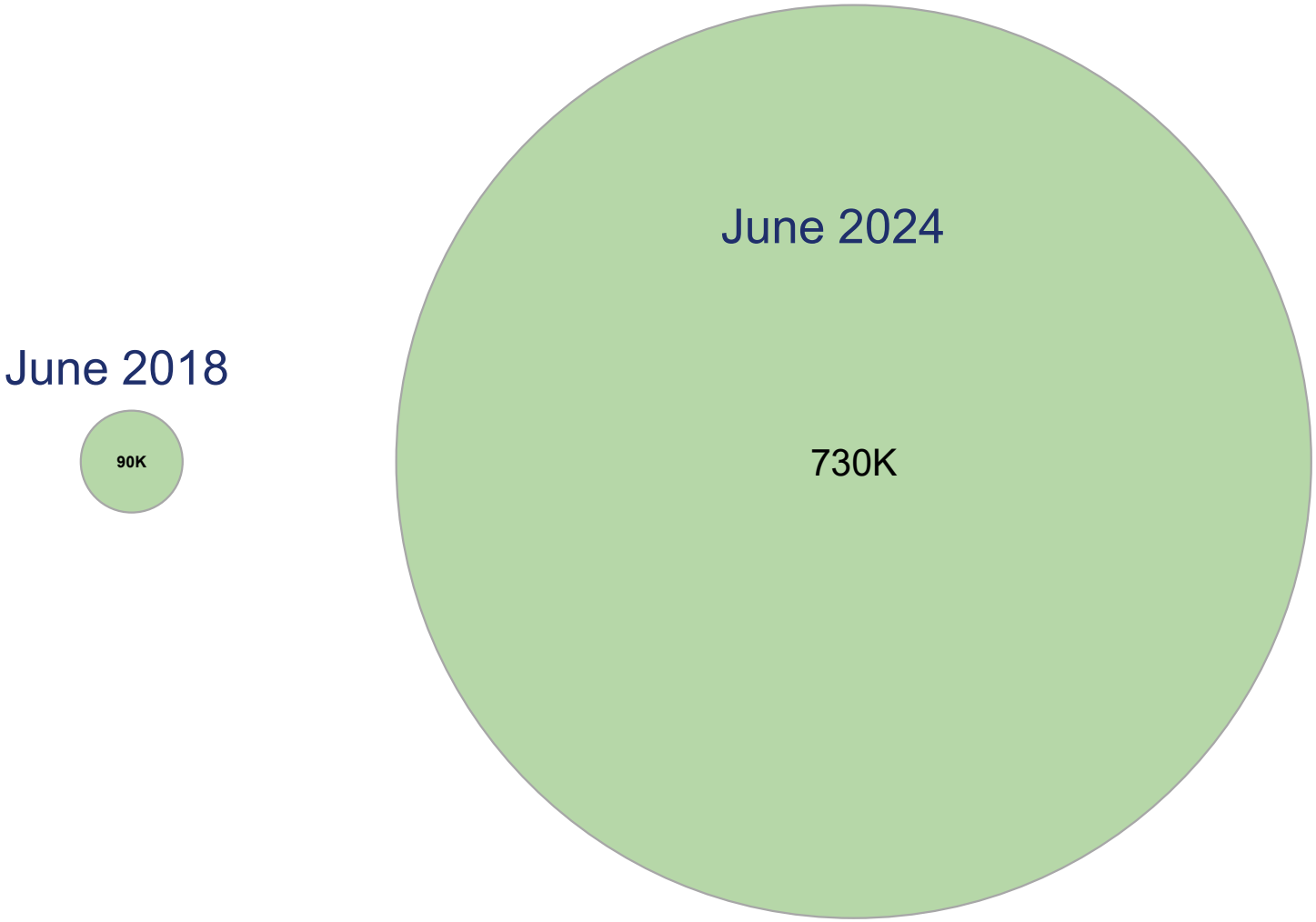
LUNARG

Validation Layer - Then and Now

June 2018

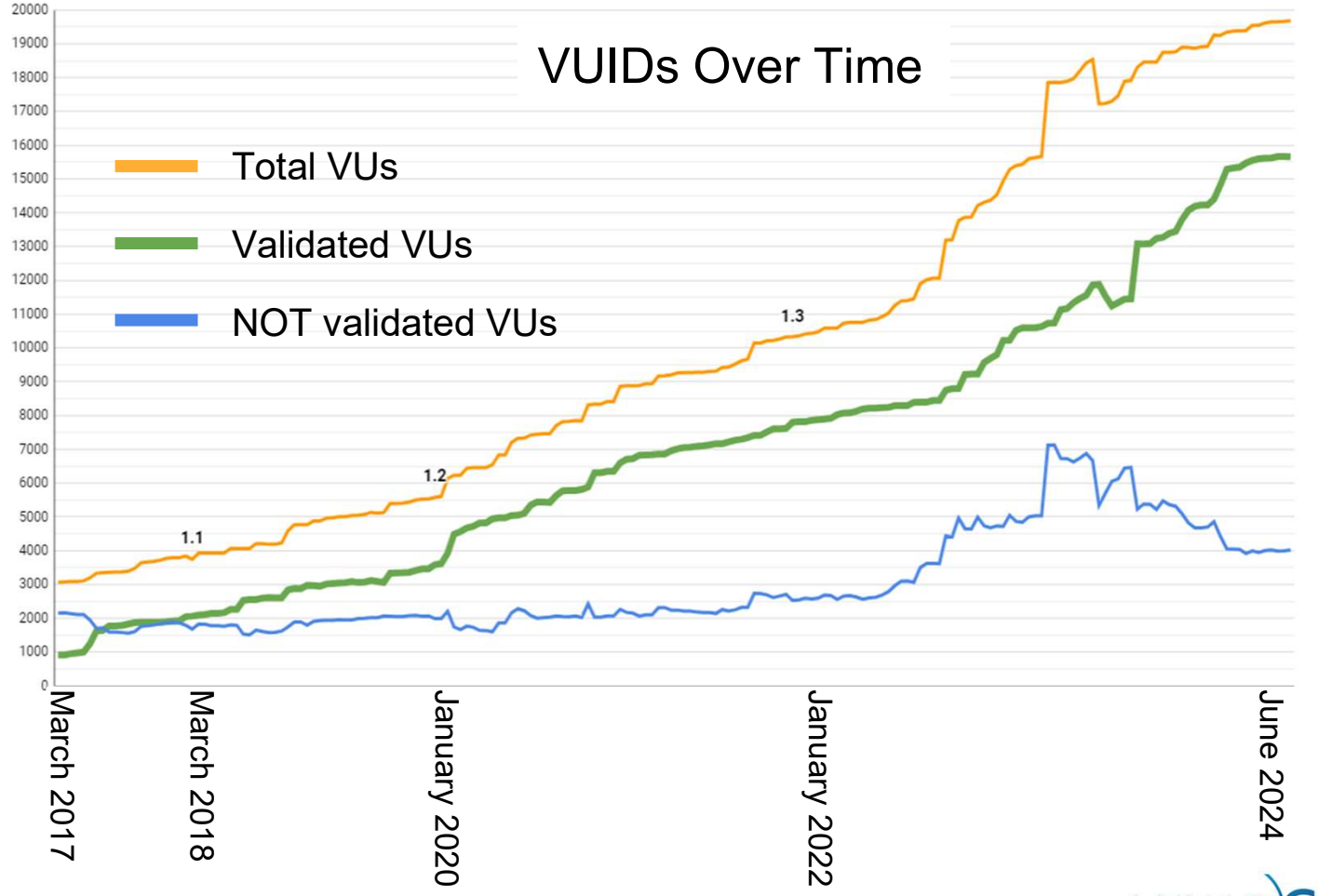
LUNARG

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

LUNARG

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
    - 2019 - Hazard detection within a single buffer
    - 2022 - Hazard detection within and between queue submissions and across queues
    - These two versions enable baseline functionality and does not cover all Vulkan extensions. More to do!

LUNARG

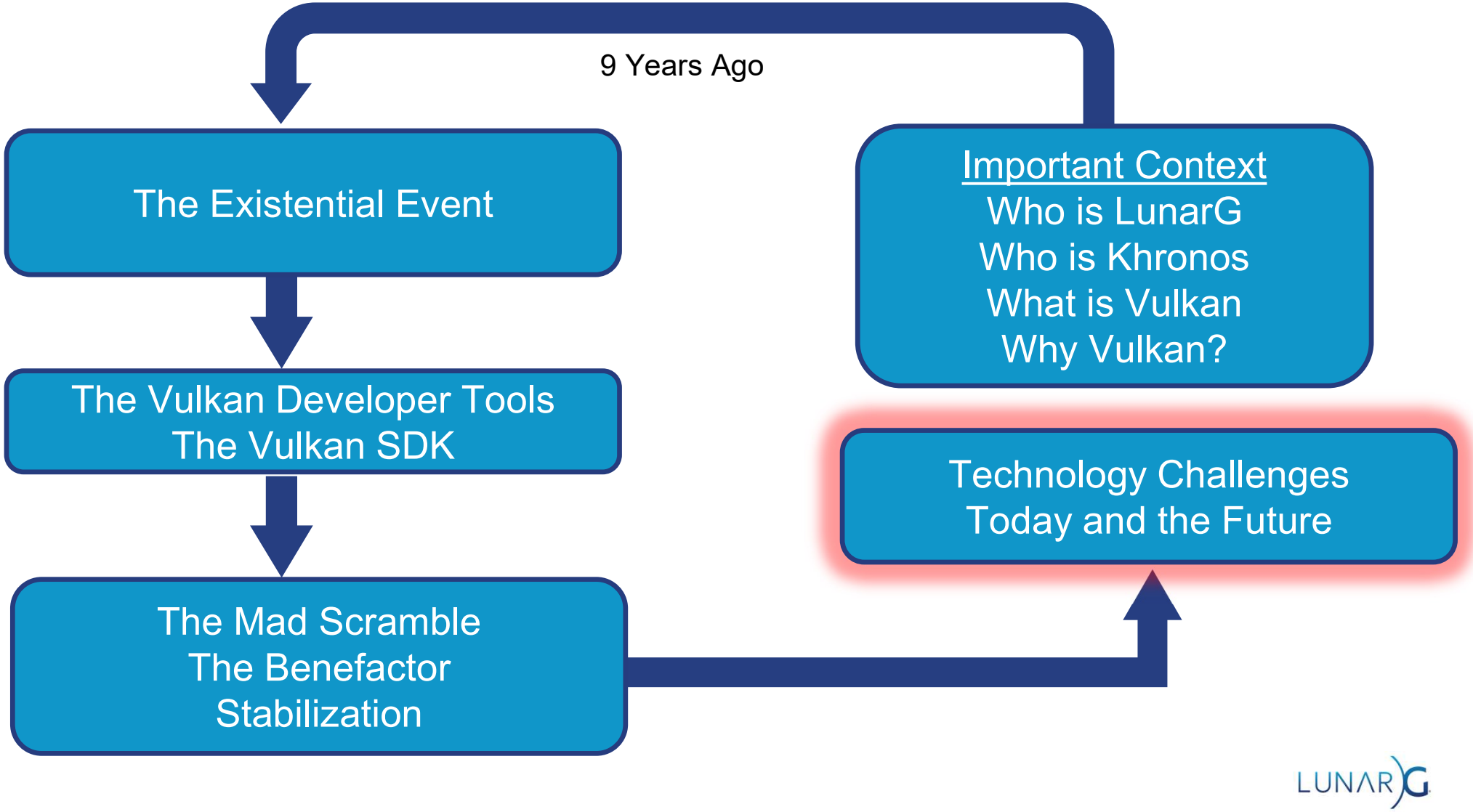
The Validation Layer - Today

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

LUNARG

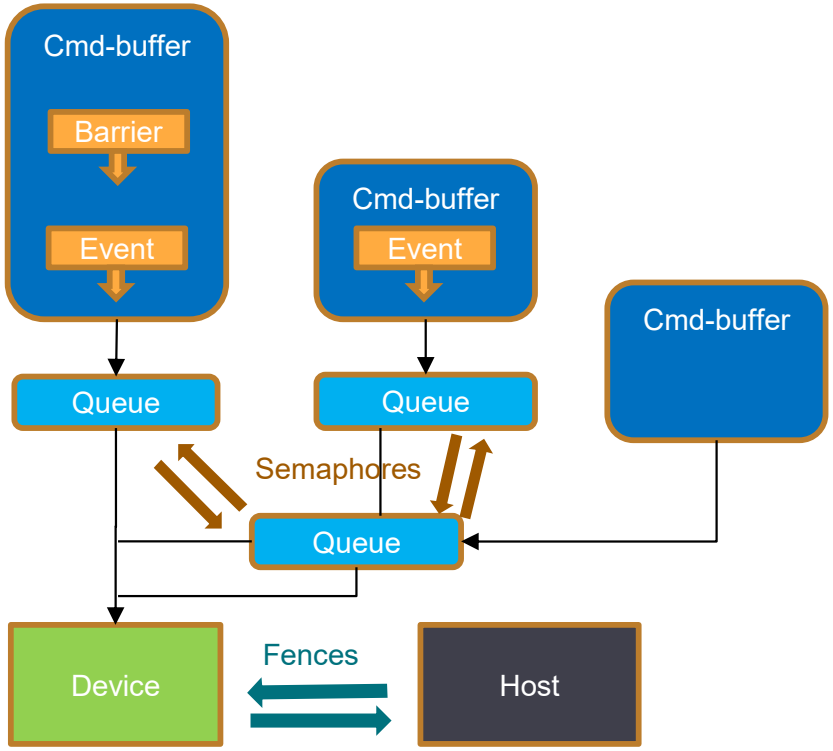
The Validation Layer

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


9 Years Ago
The Existential Event
The Vulkan Developer Tools
The Vulkan SDK
The Mad Scramble
The Benefactor
Stabilization
Important Context
Who is LunarG
Who is Khronos
What is Vulkan
Why Vulkan?
Technology Challenges
Today and the Future

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:

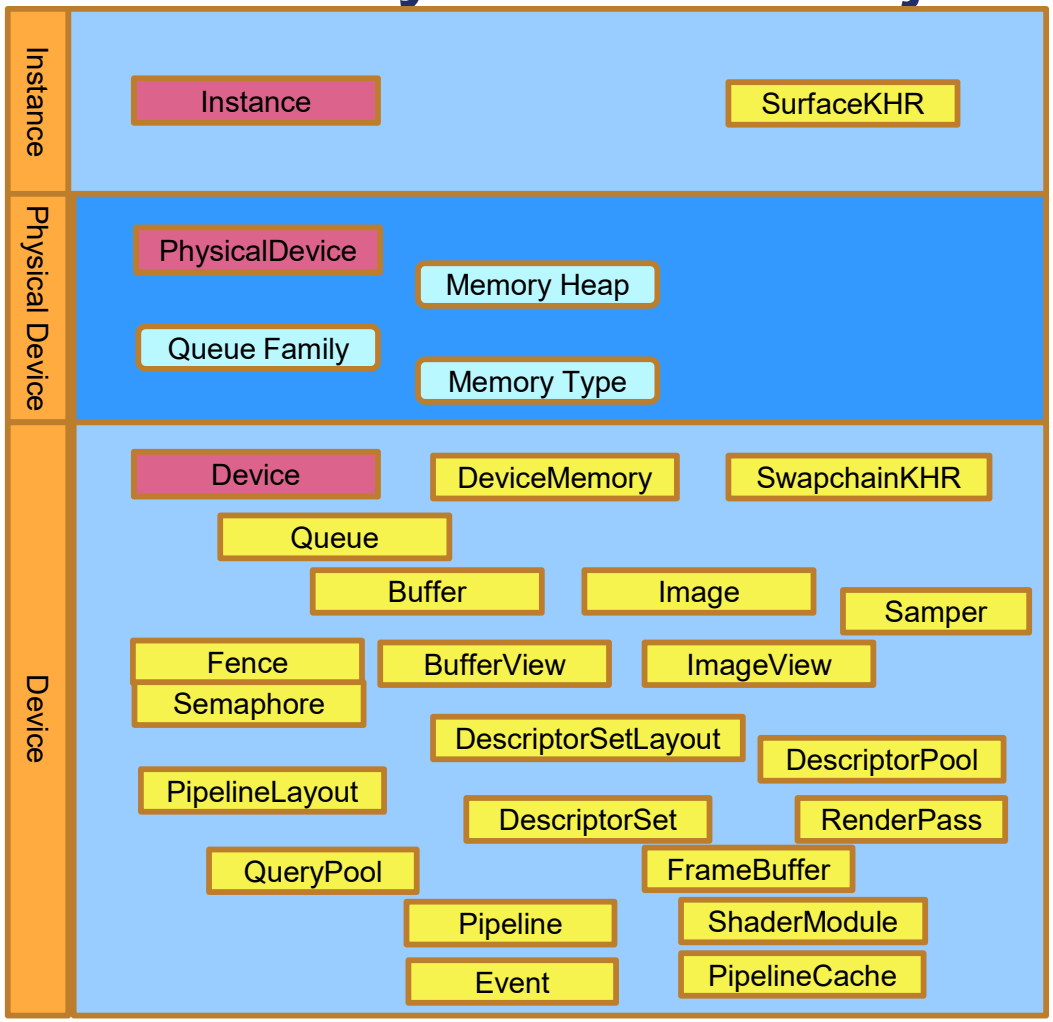
- High Performance Overhead due to required volume of state tracking
- Ongoing improvement opportunity: Performance tuning

LUNARG

Validation Layer – 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

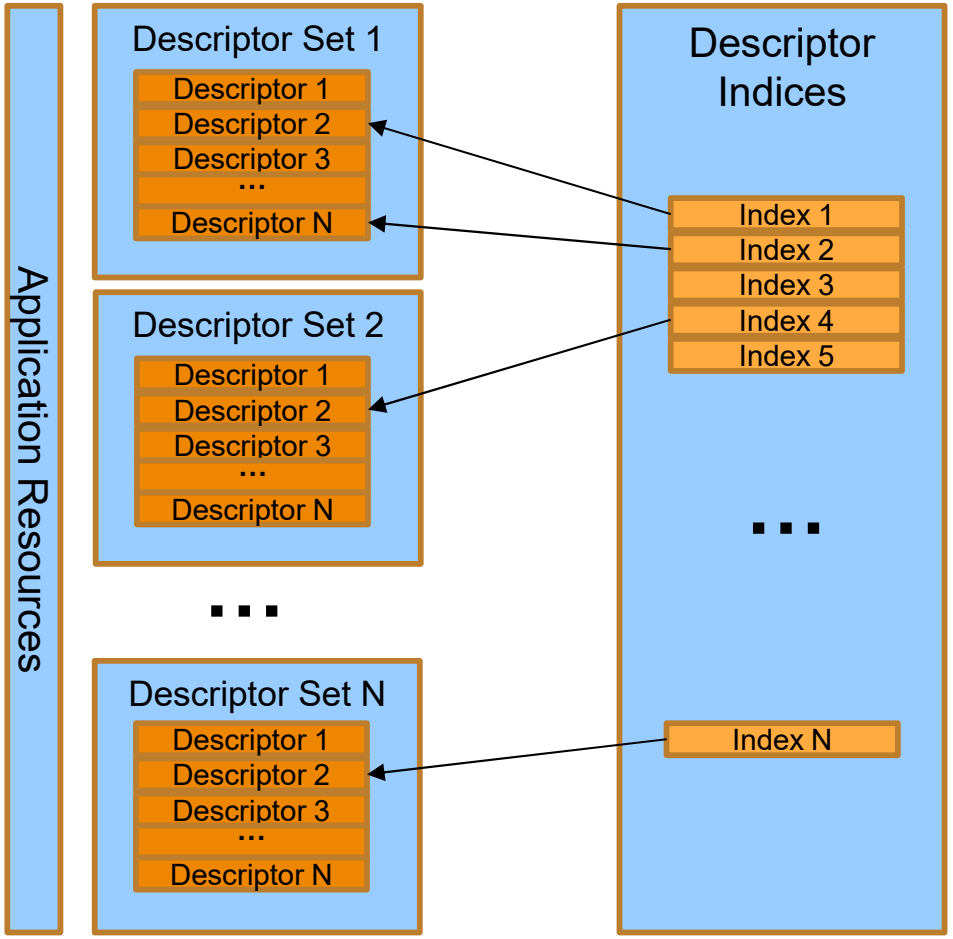
→ Complexity in the validation layer



LUNARG

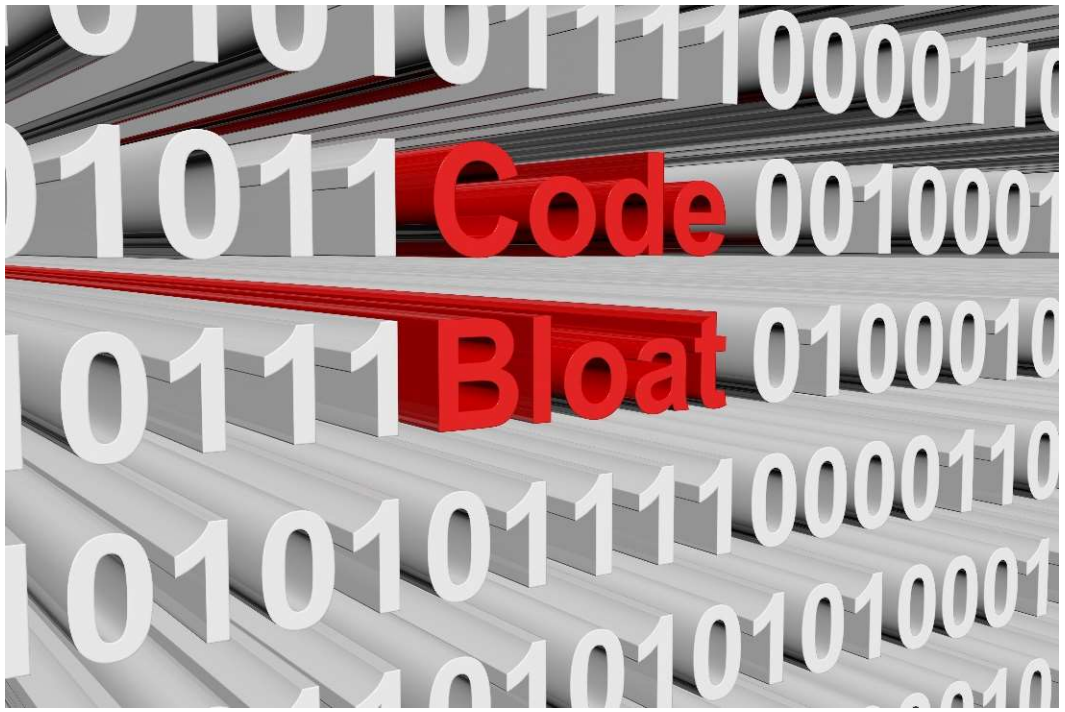
Validation Layer - Descriptor Indexing Validation

- 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



LUNARG

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

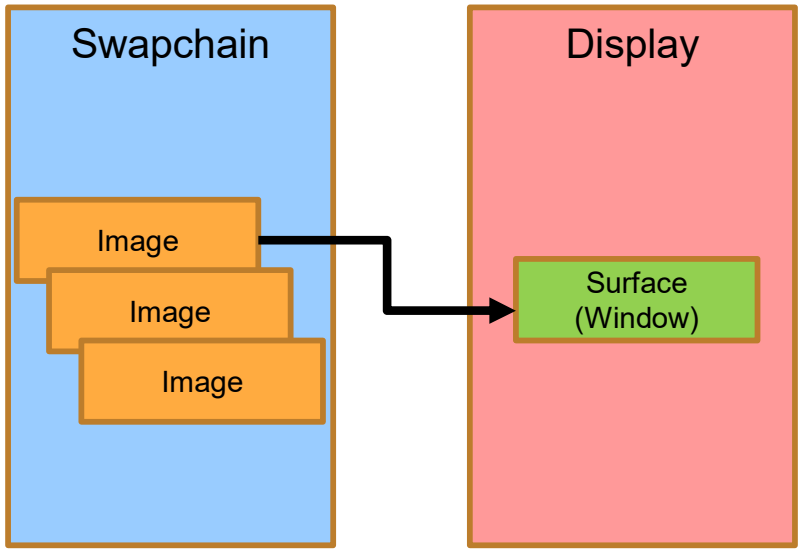
DELIVERY DELAY

- 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

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



LUNARG

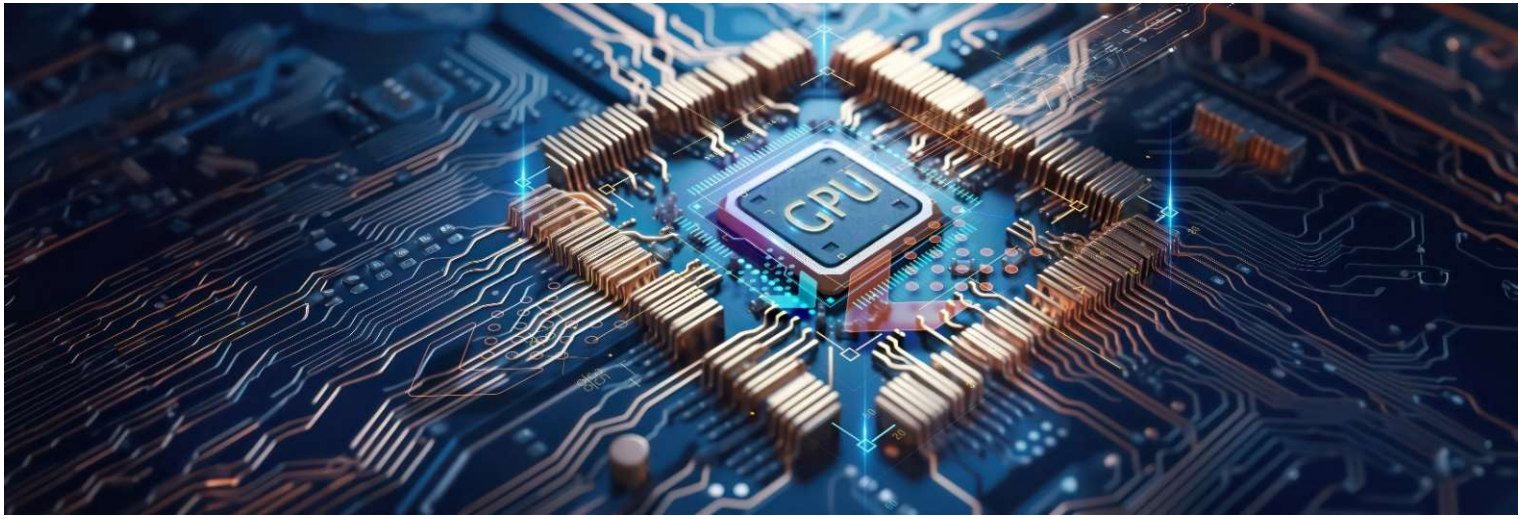
GFXReconstruct - API Explicitness

- 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?

LUNARG

The GPU-centric Universe

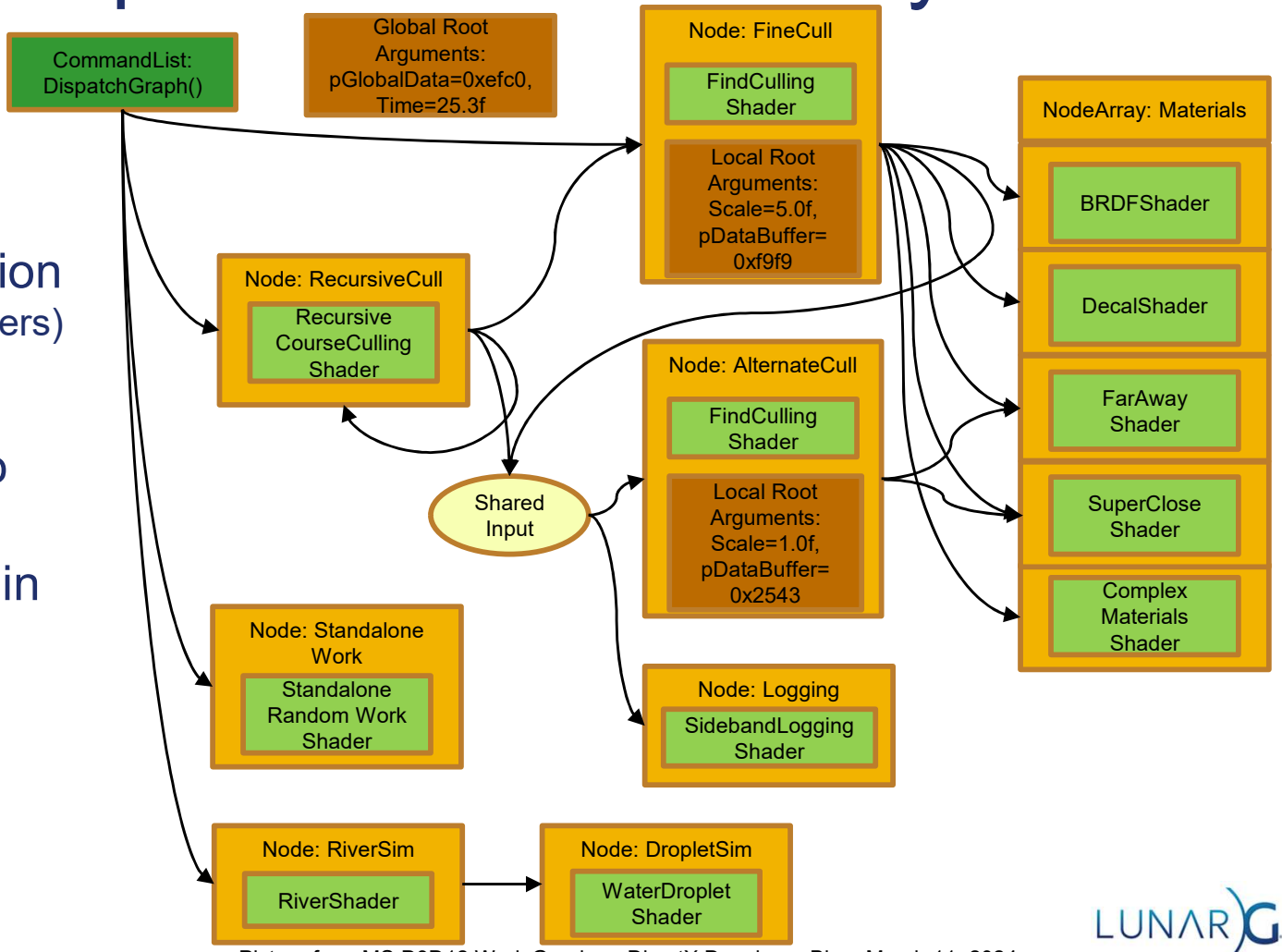
- GPUs - no longer "Graphics Processing Units"
  - Efficient processing of large blocks of data simultaneously
  - Compute - AI and ML
- Less Graphics API usage on the CPU
  - Rendering complexity still increasing via GPU driven rendering
- Many workloads moving to the GPU
  - Maximize utilization of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

LUNARG

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

LUNARG

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

LUNARG

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

From the launch of Vulkan to Today…

- There is ONE Industry-standard Vulkan SDK
  - Wide adoption
  - Strong satisfaction
  - Open and free for all developers
  - Cross-platform: Windows, Linux, Android, 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

Karen’s Reflection on the LunarG Journey

- The Power of being “Purpose Driven”
  - The ability to overcome adverse conditions to achieve amazing results!
- A Gift to the Vulkan Ecosystem
  - Useful
  - Impactful
  - Lasting and can be carried forward

LUNARG

LUNARG®

LUNARG