Vulkan SDK Benefits and Enhancements Over the Past Year

Karen Ghavam
LunarG, Inc.

LUNARG

Vulkan SDK Benefits and Enhancements Over the Past Year

BIRDS OF A FEATHER

Speaker: Karen Ghavam, LunarG Inc.

CEO and Engineering Director

Slides are available here:

https://www.lunarg.com/news-insights/white-papers/vulkan-sdk-enhancements-over-the-past-year/

LUNARG

What is Vulkan?

Vulkan is a next-generation graphics and compute API that provides high efficiency and cross-platform access to modern GPUs used in PCs, consoles, mobile devices, and embedded platforms.

LUNARG

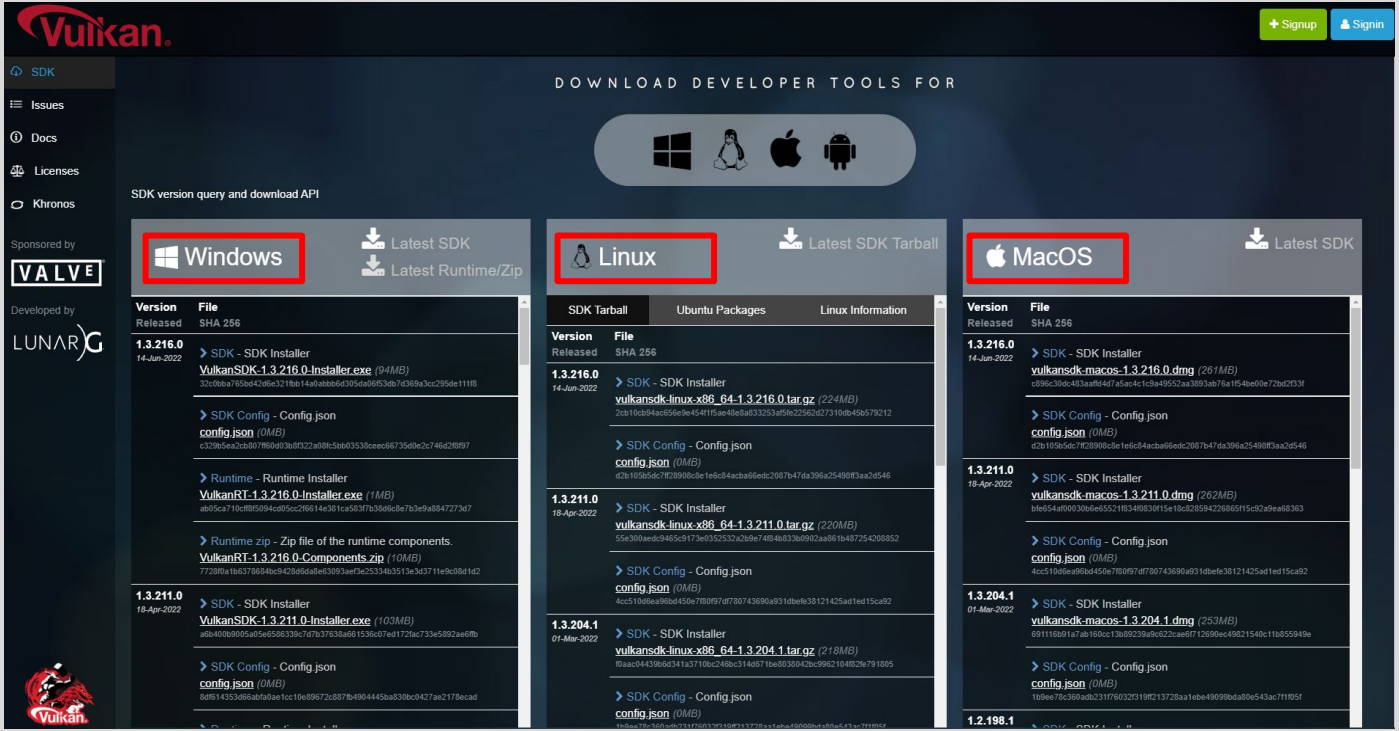
The Vulkan SDK (vulkan.lunarg.com)

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

LUNARG

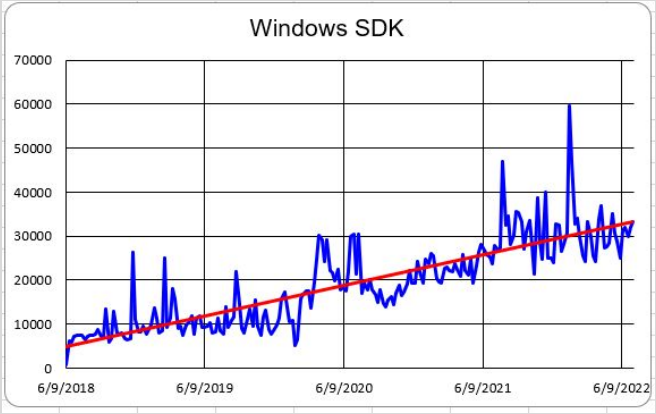
Vulkan SDK Download Page

Available Vulkan downloads for Windows, Linux, and macOS

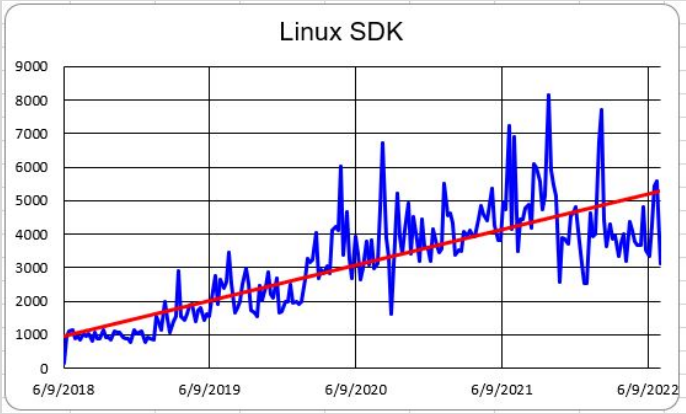


LUNARG

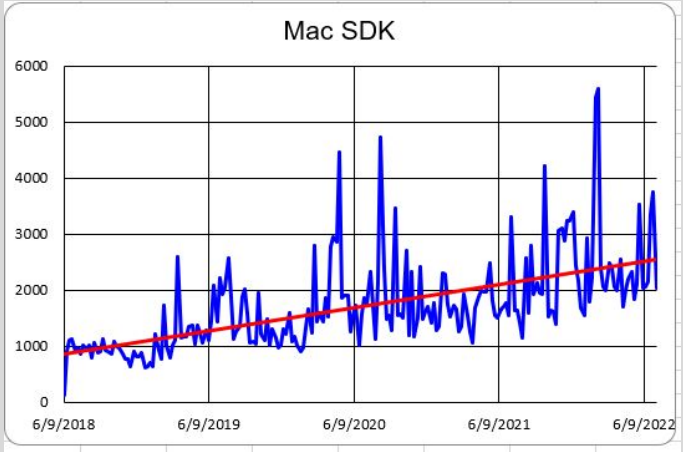
Vulkan SDK Downloads are Healthy and Continue to Grow



~32,000/week



~5100/week



~2,500/week

LUNARG

Why Use the Vulkan SDK?
– Overview –

- All SDK components come from open-source repositories
- A Vulkan application developer could build and install all of the content themselves
- However the SDK provides many benefits that save time for the application developer

See the LunarG white paper: ***Benefits of Using the Vulkan SDK***

LUNARG

Why Use the Vulkan SDK?
– Easy and Convenient –

- **An installation process that is easy and fast**
  - All of the developer tools are pre-built and installed into the correct system locations, ready for use.
- **Vetted and curated content to ensure compatibility and seamless integration**
  - The SDK components come from many open-source repositories
  - Diligence needed to ensure compatible versions of dependent repositories are used
  - LunarG collaborates with repository owners to ensure critical and compatible functionality is selected
- **Ready-to-use versions of the Vulkan Configurator**
  - The layers that the Vulkan Configurator expects are installed by the SDK
- **SDK release notes and user documentation**
  - The SDK release notes clarify new functionality released with the SDK
  - User documentation included removing need for developer to search within the repositories

Why Use the Vulkan SDK?
– Linux –

- Most up-to-date set of Linux Vulkan components
  - Linux distributions SDK components may not be updated frequently
- Linux distributions do not include all of the SDK components
- Linux tarball: Enables SDK components on many Linux distributions via the “vulkansdk” build script

See the LunarG white paper: ***Benefits of Using the Vulkan SDK***

Why Use the Vulkan SDK?
– macOS –

- All necessary macOS binaries pre-built with an option for system-level installation of MoltenVK as an ICD
  - Enables usage of Vulkan layers such as the Validation Layers via the Loader
- Building the Vulkan Loader, MoltenVK, the layers, and the associated shader tools takes work!
  - Ensuring compatible versions takes diligence
- Binaries support both Intel and Apple Silicon processors

See the LunarG white paper: ***Benefits of Using the Vulkan SDK***

LUNARG

Why Use the Vulkan SDK?
– License Registry –

A License Registry that details all of the licenses included by the SDK components

- There are many open-source licenses being used by the SDK components
- The License Registry details ALL of the open-source licenses and copyrights in use by the SDK
- Beneficial to corporations requiring license scrutiny
- Delivered via a downloadable CSV file

See the LunarG white paper: ***Benefits of Using the Vulkan SDK***

LUNARG

Why Use the Vulkan SDK?
– Shader Toolchain Tools –

Complete package of available shader toolchain tools

- There are multiple workflows used for creating SPIR-V shaders to be used with Vulkan applications
  - HLSL→SPIR-V
    - The Microsoft DirectX Shader compiler
    - Glslang (for HLSL versions of version shader model 5.1 or less)
    - Google/shaderc (via glslang)
  - GLSL→SPIR-V (Glslang, Google/shaderc)
- SPIR-V shaders -> HLSL/Metal/GLSL shaders (SPIRV-cross)
- The Vulkan SDK includes the complete package of available shader toolchain tools

SDK Content and Enhancements Over the Last Year

Developer tools in the Vulkan SDK

Vulkan Configurator - GUI application to configure layers used by Vulkan applications at runtime with built-in configurations for the SDK included layers:

1. **VK_LAYER_KHRONOS_validation** - validate application correct usage of the Vulkan API
   a. **GPU Assisted Validation** - runtime validation executed on the GPU (rather than the CPU)
   b. **Best Practice** - catch correct Vulkan API usage that still could cause application issues
   c. **Synchronization Validation** - identify resource access conflicts due to incorrect synchronization operations between actions
   d. **Debug Printf** - debug shader code using printf inside a shader
2. **VK_LAYER_KHRONOS_synchronization2 -** Emulates the VK_KHR_synchronization2 API
3. **VK_LAYER_LUNARG_api_dump** - ascii output of Vulkan API calls
4. **VK_LAYER_KHRONOS_profiles** - Downgrade the Vulkan developer's system capabilities to a specified Vulkan profile
5. **GFXReconstruct:** Capture (with VK_LAYER_LUNARG_gfxreconstruct) and Replay

Note: Underlined items are new additions in the past year

LUNARG

Additional developer tools in the Vulkan SDK

- **VOLK -** A meta-loader for Vulkan allowing dynamically loading of entry points required to use Vulkan without linking to vulkan-1.dll or statically linking the Vulkan loader
- **AMD Memory Allocator -** a library helping developers to manage memory allocations and resource creation
- **Vulkan Profiles Toolset**
  a. **Profiles Schema -** A JSON data format to communicate about Vulkan capabilities (extensions, features, properties, formats, and queue properties)
  b. **VK_LAYER_KHRONOS_profiles -** Downgrade the Vulkan developer's system capabilities
  c. **Vulkan Profiles Library -** A header-only C++ library to use Vulkan Profiles in Vulkan applications
- **Shader Tool chain** - offline executables and API libraries for:
  a. SPIRV-Tools (validator, optimizer, assembler, disassembler)
  b. glslang SPIR-V generator
  c. DXC (DirectX Shader Compiler)
  d. Shaderc SPIRV-Tools wrapper for better integration with build tools
  e. SPIRV-CROSS, a practical tool and library for performing reflection on SPIR-V and disassembling SPIR-V back to high level languages
- **Vulkaninfo** - Show GPU device properties and extensions, installed layers, supported image formats, properties...
- **vkvia** (Vulkan Installation Analyzer)

Note: Underlined items are new additions in the past year

LUNARG

New with Vulkan 1.3: Vulkan Profiles

- A mechanism that enables the precise specification of capabilities
- Enables communication of capabilities between participants in the Vulkan Ecosystem
  - Streamline the development and deployment of portable applications
- Each profile specifies
  - A set of required extensions, with supported limits, features, and formats for
  - A core version of Vulkan

Vulkan Profiles Toolset

- Creating portable Vulkan applications in terms of Vulkan capabilities
  - Vulkan Profiles: Explicit Vulkan capability requirements and/or supports
    - Nothing groundbreaking, just a data convention and a toolset.
    - Not targeting homogeneity of the ecosystem, specifying a domain of relevance.
- Easier Vulkan development for a selected range of actual ecosystem devices

LUNARG

Example Profiles Usages

- Roadmap profiles: To express guidance on the future direction of Vulkan devices
  - In the Vulkan Specification: **Vulkan Roadmap 2022**
  - In the SDK: **VP_KHR_roadmap_2022**
- Platform profiles: To express the Vulkan support available on different platforms
  - In the SDK: **VP_LUNARG_desktop_portability_2021**
- Device Profiles: To express the Vulkan support of a single Vulkan driver for a Vulkan device
  - **Gpuinfo.org** provides device profiles
- Architecture Profiles: To express the Vulkan support of a class of GPUs
  - For example, all Nvidia RTX 2000 GPUs
- Engine Profiles: To express some requirements of the rendering code path

The Vulkan Profiles Toolset Components

- **The Vulkan Profiles schema**
  - A JSON data format to communicate about Vulkan capabilities: extensions, features, properties, formats, and queue properties.
  - Each revision of Vulkan API is represented by a schema that supersedes older versions of Vulkan API.
- **The Vulkan Profiles comparison table**
  - A markdown table representation comparing Vulkan Profiles in the SDK
- **The Vulkan Profiles layer**
  - Downgrade the Vulkan developer's system capabilities
  - Replaces the devsim layer
- **The Vulkan Profiles library**
  - A header-only C++ library for using Vulkan Profiles in Vulkan applications
  - Checking Profiles support on a device and creating a vkDevice instance with the profile features and extensions enabled
- Coming soon: **Profile combining tool**
  - Intersection and Union

The Vulkan Profiles Toolset - More information

See the LunarG white paper, ***The Vulkan Profiles Toolset Solution***

See the SIGGRAPH Birds of a Feather (BoF) session:
Vulkan SDK tools to use and create Vulkan Profiles

- August 9, 8 AM Pacific

LUNARG

Repackaged SDK

- The initial Windows SDK was released as one large blob
  - Didn’t allow for managing the SDK size
  - Couldn’t track usage of optional SDK components
- Repackaging applied to both the Windows SDK and the macOS SDK
  - Qt Installer framework (richer feature set)
  - Consistent look and feel
- Core packages
  - Validation Layers, Vulkan Configurator
  - …
- Optional packages
  - 32-bit versions of libraries, debuggable shader tool chain libraries
  - …
- **For more details**, see the LunarG white paper, “***The Repackaged Windows Vulkan SDK***”

LUNARG

Validation Layer
Performance Improvement Initiative

- Performance regression test suite
  - Catches performance regressions (avoid performance degradation over time)
- Primary Problem areas
  - Many active threads in Vulkan applications; but a single lock per Validation Object in the Validation Layer
  - Large “bindless” DescriptorSets
- Some performance optimizations done in the last year
  - Fine grained locking in the Validation Layer - Huge Gains!
  - Linear memory allocation for GPU-AV (**VMA documentation**)
- Current focus area
  - Bindless descriptor validation (moving it to GPU-AV and off of the CPU)

Performance Improvements from Fine Grained Locking

Application	API	FGL disabled (FPS)	FGL enabled (FPS)	perf improvement
Ashes of the Singularity: Escalation	DX12 / VKD3D-Proton	24.16	59.36	145.70%
Deus Ex: Mankind Divided	DX11 / DXVK	33.3	31.3	-6.01%
Deus Ex: Mankind Divided	DX12 / VKD3D-Proton	17.7	31.9	80.23%
F12020	DX11 / DXVK	43	50	16.28%
Hitman2	DX11 / DXVK	28	29.54	5.50%
Hitman2	DX12 / VKD3D-Proton	13.77	32.73	137.69%
Rise of the Tomb Raider	DX12 / VKD3D-Proton	17	60	252.94%
Serious Sam Fusion 2017	Vulkan	112	115	2.68%
Sid Meier's Civilization VI	DX12 / VKD3D-Proton	4.3	11	155.81%
Strange Brigade	DX12 / VKD3D-Proton	19.6	54.5	178.06%
Strange Brigade	Vulkan	19.5	45.6	133.85%
AOE4	DX12 / VKD3D-Proton	120	110	-8.33%
Death Stranding	DX12 / VKD3D-Proton	10	25	150.00%
Market Of Light (UE5 demo)	DX12 / VKD3D-Proton	15	24	60.00%
Farming Simulator 22	DX12 / VKD3D-Proton	15	40	166.67%

LUNARG

Performance Improvements from Linear Memory Mapping in GPU-AV

Application	linear alloc OFF (FPS)	linear alloc ON (FPS)	performance improvement
gfxrecon DoomEternal	0.614	1.842	**200.00%**
gfxrecon RDR2	11.35	33.52	**195.33%**
DoomEternal homescreen	16	19	**18.75%**
Strange Brigade	7	29	**314.29%**
DOTA2	2.9	5.2	**79.31%**

LUNARG

Portability Enumeration

- Deploy Vulkan applications on systems without native Vulkan drivers
- More information about the Vulkan Portability Initiative: https://www.vulkan.org/porting#vulkan-portability-initiative
- Required extensions to use the portability solution
  - VK_KHR_portability_enumeration
    - Receive portable implementations during physical device enumeration
  - VK_KHR_portability_subset (currently provisional)
    - Identify differences between a portable implementation and a fully-conformant Vulkan implementation.
    - Enables writing applications to be portable
- As of SDK release 1.3.216.0, the VK_KHR_portability_enumeration extension support is included in the Vulkan Loader

Vulkan Loader Improvements

- **Improvements to assist developers in resolving difficult issues**
- **Loader Identification**
  - VK_LOADER_DEBUG=info (“all” will also identify the loader)
  - Identify loader version and where it was built from (Loader Git history)
- **Enhanced layer debugging**
  - VK_LOADER_DEBUG=layer
  - Identify which layers are enabled and their type (explicit vs. implicit)
- **Linux Consistent Device Ordering**
  - Consistent order of devices from run to run

For more details, see the LunarG white paper: ***1.3 Vulkan Loader Improvements***

LUNARG

Synchronization Validation

- **By design, Vulkan is an explicit API**
  - The programmer must tell Vulkan when 2 commands depend on each other
  - This is done by defining barriers
  - **Execution Dependencies**
    - Most Vulkan commands are started in queue submission order but may execute in any order
    - Even commands using the same pipeline stages!
  - **Memory Dependencies**
    - GPUs have lots of caches and are accessed by pipeline stages

Synchronization Validation - Phase II

Phase I implementation (August 2020)

- Identifies resource access conflicts due to missing or incorrect synchronization operations between actions (draw, copy, dispatch, blit) reading or writing the same regions of memory
- Functionality includes commands within a single buffer

Phase II implementation (Available soon!)

- Same as phase I, but adding multiple command buffers
- Becoming available as I speak. Will be available as alpha quality in the next SDK delivery

See the SIGGRAPH BOF:

Vulkan Synchronization Validation: Tutorial and Update

- August 10, 8-9:30 AM Pacific

LUNARG

In Summary…

- The benefits of using the Vulkan SDK
- The developer tools in the SDK
- New developer tools enabled over the last year
  - The Vulkan Profiles Tool set
  - VOLK - the meta-loader
  - AMD Memory Allocator - management of memory allocations and resources
  - Repackaged SDK for modularity and optional packages for download
  - Validation Layer Performance improvements over the last year
  - The Portability Enumeration extension
  - Vulkan Loader improvements
  - Phase II Synchronization Validation

LUNARG

Vulkan SDK Benefits and Enhancements Over the Past Year

Slides are available at:
https://www.lunarg.com/news-insights/white-papers/vulkan-sdk-enhancements-over-the-past-year/

Questions?

LUNARG


LUNARG®