Progress Report: Public Results of 12/21 LunarG Vulkan Ecosystem & SDK Survey

Karen Ghavam, LunarG
January 2023

Executive Summary

This report provides a status update relative to feedback and improvement requests from the LunarG SDK & Ecosystem Survey conducted in December of 2021. You can see the full report from the December 2021 survey in the white paper Results of December 2021 LunarG Vulkan Ecosystem & SDK Survey.

Progress Report

The sections below come from the original survey report: Results of December 2021 LunarG Vulkan Ecosystem & SDK Survey.

Text in blue are the updates being made today with the release of this whitepaper, indicating progress made on the requested ecosystem improvements.

Results from the 2022 Planned Actions

LunarG and the Vulkan Working Group prioritized the following projects in 2022:

1. Deprecate Visual Studio 2015 - Completed
    a. SDK and several repositories will no longer build, test, and support VS2015.
2. Validation layers. Keep doing what we are doing.
    a. Continue focus on validation layer performance initiative.
        i. LunarG is in the process of refactoring the descriptor index validation, which is currently the largest performance bottleneck. A design has been approved with some proof of concept work. Expect to see improvements added to the validation layers in the next several months.
    b. Continue focus on synchronization validation implementation.

i. This capability was first added to the 1.3.224.0 SDK released in August of 2022 and has had improvements in subsequent SDKs since the initial release.
ii. A Khronos Vulkanise presentation was done regarding synchronization validation and can be viewed on the Khronos Vulkan Youtube channel here:
How to Use Synchronization Validation Across Multiple Queues and…

c. Continue GitHub issue responsiveness and improved coverage.
d. Avoid releasing extensions without timely validation layer coverage.
i. As new extensions are released, Khronos Working Group members are responsible for the development of the validation for those extensions. LunarG does it's best to provide technical support and reviews of their development to help facilitate as timely releases as possible.

3. Vulkan learning resources (tutorials, samples, documentation, …)
a. We added 21 samples to the repo in 2021. For 2022 more complex samples will be explored.
i. We added an additional 15 samples in 2022 including several for recently introduced extensions (profiles, mesh_shader, portability_subset, graphics_pipeline_library and others). We also secured funding to start development on complex samples starting in early 2023.
b. Explore gathering some high-quality, non-Vulkan-specific resources to aid developers new to graphics programming.
i. To help spin up developers on the basics of computer graphics, we've added a page to our website with recommendations on resources for beginner and advanced graphics topics. As always, we're open to suggestions on further useful resources to add here. See the "New to Graphics Programming?" page of the Khronos.org website.
c. Evaluate currently available best practice documentation and potential ways to improve its visibility.
i. In this last year, NVIDIA contributed best practices code for their GPUs as part of the Validation Layer Best Practices object. So to date, there are runtime best practices checks for NVIDIA, ARM, AMD, and Imagination Technologies. These can be enabled by the Vulkan Configurator included in the Vulkan SDK.
d. Explore collaborations with external tutorials such as www.vulkan-tutorial.com to improve the quality of the tutorial by adding new content and/or making sure existing content is accurate.
i. We now have an ongoing collaborative project in place to update and improve vulkan-tutorial.com. In 2022, we've added several chapters to this tutorial covering topics such as swapchain recreation, command buffer re-recording, dynamic viewports, and improved memory handling. We have several more in flight and planned for 2023 including a chapter

on using compute in Vulkan, expanded info on shaders, portability for macOS, and more.

e. Continue Vulkanised webinars with a focus on educational content. All content is made available for free on the Vulkan Youtube channel.
   i. Vulkanised 1.3 SDK Webinar - March 10, 2022
   ii. Vulkan 1.3 - A Vulkanised Webinar
   iii. Accelerating Machine Learning with Vulkan
   iv. Vulkan SDK Tools to Use and Create Vulkan Profiles
   v. Reducing Draw Time Hitching with the Vulkan Graphics Pipeline Library Extension
   vi. How to Use Synchronization Validation Across Multiple Queues and Command Buffers
   vii. Cross-Vendor Mesh Shading and Shader Debugging in RenderDoc

4. During 2022, LunarG will prioritize the following initiatives:
   a. Addition of VOLK and Vulkan Memory Allocator to the SDK
      i. VOLK was added to the SDK as of SDK 1.3.211.0 released in April of 2022.
      ii. The Vulkan Memory Allocator was added to the SDK as of 13.216.0 released June 2022.
      iii. The Vulkan Hardware Capability Viewer was added to the SDK as of SDK 1.3.231.0 in October of 2022
   b. Addition of GPU-AV and Debug Printf support to the macOS SDK
      i. Some investigation was completed to determine how to enable Debug Printf and GPU-AV. Once MoltenVK adds support for the VK_KHR_shader_non_semantic_info and VK_EXT_multi-draw extensions, this can be enabled. See Add support for extension VK_KHR_shader_non_semantic_info_extension #1214 for more information.
   c. Addition of support for iOS as a target (in other words, create a loader for iOS)
      i. Due to resource constraints, this has not yet been enabled.

Results from the Open-ended Feedback

There were a lot of open-ended comments received from the respondents. This section highlights open-ended feedback that was mentioned multiple times.

- How can the validation layers be improved?
  - Error Reporting:
    - Refer to the project in the Validation Layer repository that tracks enhancements to the formatting of validation layer errors.
    - Some of the comments indicated that individuals were not aware of validation layer error reporting improvements and the Vulkan Debug

Utilities white paper. These resources could prove helpful to these individuals.
        - Multiple individuals struggle with interpreting errors from the validation layers. We have ideas for tools to help with this. However, it hasn't made our to-do list yet.
    - Performance
        - Validation layers performance needs improvement. We have been working on performance improvements over the last year and have a significant improvement recently merged (fine-grain locking).
        - But there is still more to do…
            - See the section "Results from the 2022 Planned Actions" above. Improving validation layer performance was in the planned actions for the 2022 year and improvements were made.
    - Synchronization
        - The validation of Vulkan synchronization has been very helpful to many individuals and this was stated multiple times.
        - Request for synchronization validation across command buffers in a queue. This is under development and will be released in the near future.
            - See the section "Results from the 2022 Planned Actions" above. Advancements to synchronization validation coverage was in the planned actions for the 2022 year.
    - Better coverage
        - Although the survey feedback indicates that the validation layers are finding most of their application issues and rarely give false positives, we know there are many gaps to still be filled.
- What features, tools, and/or improvements would you like to see added to the LunarG Vulkan SDK (Windows, Linux, and/or macOS)?
    - Requests for more/better/improved tutorials and samples. The SDK no longer provides samples or tutorials because it is not considered a tool for teaching the Vulkan API, but rather a tool for developing to the Vulkan API. There are Khronos initiatives to improve samples (KhronosGroup/Vulkan-Samples) and tutorials.
        - See the section "Results from the 2022 Planned Actions" above. Improving Vulkan Learning resources was in the planned actions for the 2022 year.
    - Requests for the Vulkan Memory Allocator to be added to the SDK. Unfortunately, resources did not allow for integration of the Vulkan Memory Allocator, Vulkan Hardware Capability Viewer, and VOLK into the SDK during 2021. This year's survey also reveals high interest in GLFW and SDL.
        - See the section "Results from the 2022 Planned Actions" above. Improvements to the Vulkan SDK was in the planned actions for the 2022 year.
    - Requests for consistent tagging of SDK releases

        - For each SDK release, a tag is added to all Khronos repositories except for the KhronosGroup/SPIRV-Cross and the KhronosGroup/SPIRV-Reflect repositories. The branches are named SDK-X.Y.ZZZ (where X.Y.ZZZ is the Vulkan Header version for the SDK) and the tags are on that branch named SDK-X.Y.ZZZ.release where the release is usually “0” but will be incremented if additional SDKs are released for that header version.
    - Request for more complete macOS support using MoltenVK. During 2022 LunarG will be investing resources to enable GPU-AV and Debug Printf on macOS as well as to develop a loader for iOS.
        - See the section “Results from the 2022 Planned Actions” above. macOS support of the GPU-AV and Debug Printf was in the planned actions for the 2022 year.
    - Requests for Vulkan synchronization debugging tools (online or offline).
        - This has not been investigated or worked on yet due to limited resources.
    - Request for a tool for inspecting calls from GFXReconstruct trace..
        - GFXReconstruct has released the inspection tool “convert” as of October 2022. You can read more about it’s usage in the GFXReconstruct usage documentation found in the repository.
- What inhibits you from effectively and efficiently developing Vulkan applications?
    - Again, the request for more/better/improved samples, tutorials, and documentation.
        - See the section “Results from the 2022 Planned Actions” above. Improving Vulkan Learning resources was in the planned actions for the 2022 year.
    - Learning the Vulkan API takes significant time and effort.
    - VK_ERROR_DEVICE_LOST is hard to debug.
    - Shader debugging is a challenge.
        - The ability to do source code level debugging of shaders has been enabled in RenderDoc as of October 2022. See the white paper, Source-level Shader Debugging in Vulkan with RenderDoc for more information.
        - A Khronos Vulkan webinar provided details on how to use the source-level shader debugging and can be found on the Khronos Vulkan Youtube channel here: Cross-Vendor Mesh Shading and Shader Debugging in RenderDoc.
    - GPU driver bugs can be a challenge.
    - Layer ecosystem issues. Bad implicit layers in the ecosystem can cause hard to diagnose problems.
        - Vulkan Loader enhancements have been made to improve the capability to debug issues caused by bad implicit layers. See the white paper The Vulkan Loader and Vulkan Layers: Diagnosing Layer issues for more details.

    - Glslang missing features. The pipeline to compile GLSL into spir-v modules needs some improvement.

- Which aspects/features of developing with the Vulkan API do you like?
    - API Definition is great
        - Low-level approach and exactness of the API
        - Vulkan is a GPU specification
        - The amount of low-level control I now have
        - High-quality specification (precise, clear, complete)
        - Performance that the Vulkan API enables
        - Multi-threading
        - Independence of shader language
    - Cross-platform
    - Validation Layers and Tooling
        - Good error validation
        - RenderDoc!
        - Debug Printf
        - Synchronization validation
- Is there any feedback you would like to share?
    - Please make the chunked specification the default.
    - When are we going to get a cross-vendor mesh shading extension?
        - The extension VK_EXT_mesh_shader was released in September 2022 and was included in the 1.3.221.0 SDK released in October of 2022.
    - An official open-source library on top of Vulkan that could simplify things to a level like Metal
    - VK_KHR_dynamic_rendering was a big step in the right direction.

Do you have additional suggestions for the improvement of the Vulkan Configurator?

There were several comments about bad implicit layers having negative interactions with applications.

- Developers want to have the ability to disable these layers via vkconfig
    - This feature is available today
- Developers want to see the list of environment variables that an application needs to set to opt-out of any implicit layers.
    - All implicit layer environment variables are now logged by the loader (1.2.189 or later) by setting the loader debug environment variable *VK_LOADER_DEBUG=layer* or *VK_LOADER_DEBUG=all*. The loader also lists what layers are used during either CreateInstance or CreateDevice
- Developers want a “safe mode” loader that would automatically disable implicit layers. This was considered but due to bad unintended consequences for necessary and good

implicit layers, it was decided not to implement this suggestion. See Vulkan-Loader issue #513.

- Note: We are in the process of documenting the implicit layer issue in the ecosystem as well as solution alternatives that have been considered to increase ecosystem understanding of the situation and potential mitigations that can be taken.
- Vulkan Loader enhancements have been made to improve the capability to debug issues caused by bad implicit layers. See the white paper The Vulkan Loader and Vulkan Layers: Diagnosting Layer issues for more details.