Vulkan Ecosystem & SDK Survey

Results from December 2019 Survey
October 2020 LunarG Status Update

Executive Summary

In December of 2019, LunarG conducted a Vulkan ecosystem survey. The results of that original survey can be found here. There were open-ended questions in the original survey (#21, #22, and #23) along with the ability to add comments to any of the questions throughout the survey. Through these open-ended questions, there were many requests for additions and enhancements to Vulkan ecosystem tools. This report is a status update a year later, itemizing the key requests from the survey and the status on those requests.

Look for a new 2020 Vulkan Ecosystem & SDK Survey from LunarG in November 2020. We will again be capturing the requirements of Vulkan developers to help shape the priorities of the LunarG software engineering investments in 2021.

Completed Ecosystem Improvements

1. Add the ability to check for the supported version on the platform
   - This is already achievable by using vulkaninfo

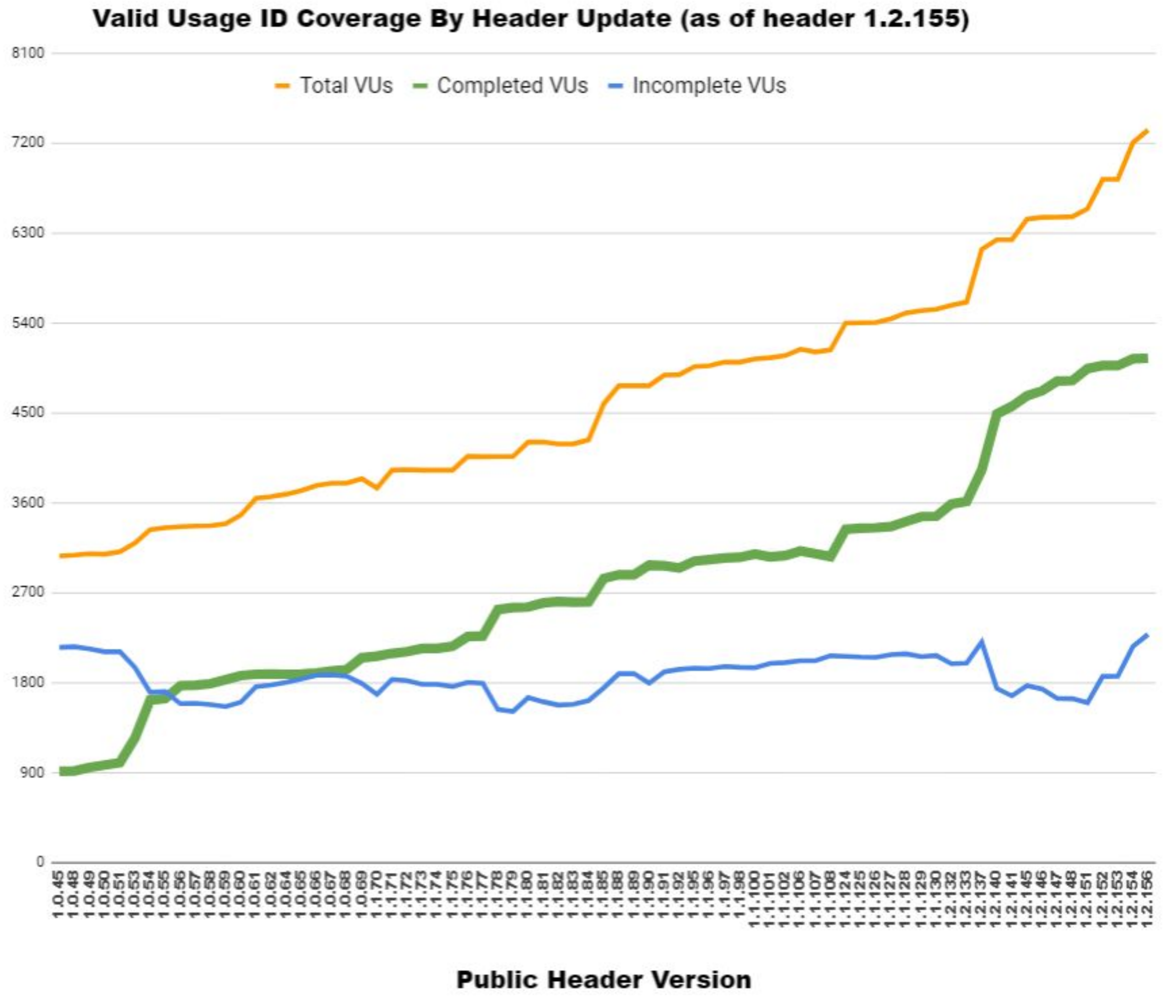
2. Add vk.xml to the SDK
   - This has been added to the registry directory in the SDK

3. Shader tool-chain requests for the SDK: Add SPIR-V Reflection tools, Add DXC, all Vulkan shader tool chain tools/libraries need to be included in the SDK
   - Status Update:
     i. March 2020: Shader tool chain API libraries have been added to the SDK from the SPIRV-Tools, glslang, shaderc, and spirv-cross repositories. Previously, only the offline executables were included. The API libraries enable shader compilation by applications at runtime.
     ii. June 2020: DXC (The DirectX Shader Compiler) was added to the Windows SDK. In August it was added to the macOS and Linux versions as well.

4. Add the Android build of the Validation Layers to the SDK
    - Beginning May 2020, with each SDK release, the Validation Layers github is updated with a release of the Android build of the validation layers.

5. Validation of Vulkan Synchronization
    - Extensive work was done over the last year to add some validation for synchronization. The first SDK to include the synchronization validation was released in August of 2020.
    - You can read more about the synchronization validation in the SDK documentation. There is also an extensive writeup about synchronization validation available.

6. Increase Validation Layer Coverage
    - At LunarG, we know there are many gaps still remaining in Validation Layer coverage. When the API first launched in February of 2016, the validation layers were incomplete. By the time 1.1 was released, we instituted a policy that all new Vulkan API entry points/extensions couldn't be released without validation, which avoids increasing the backlog. Our development at LunarG has focused on first implementing missing validation checks found by application developers since these are the ones most likely to be found by other developers. Then we can go work on the backlog. LunarG has also reworked much of the validation layer code to make it easier to use and add additional checks. Of course, pull requests are appreciated as well!
    - On the next page you will find a graph that shows valid usage checks additions/completions over time. As you can see, new VUIDs are always being added as new Vulkan API entry points and extensions are added (the yellow line). Validation layer checks for these VUIDs are keeping up (the green line). The blue line represents the backlog of VUIDs, which today do not have any validation checks.



7. Need to improve the visibility/usability of Validation Layer objects (Best Practices, GPU Assisted Validation, debug printf)
    - Knowing how to discover, activate, and configure layers can be confusing. We created “Vulkan Configurator” to simplify this for the user. It is included in the SDK and is supported on Windows, Linux, and macOS. Here is a short video demo. Full documentation is here.

8. Include "Resources" folder with URLs to all known tool repos by category w/ short description
    - The SDK Release Notes list all repositories used to build the SDK along with the corresponding commit IDs for each repository.
    - Beginning in October 2020, an API was added to vulkan.lunarg.com that allows you to download a config.json file for an SDK release. That config.json will list all the repository commits used to build the SDK At vulkan.lunarg.com/sdk/home you can view how to use this API.

9. Device Simulation Layer (devsim) needs updating
    - As of October 2020, the Device Simulation Layer (devsim) was updated to add support for the VK_KHR_portability_subset extension. This layer is included in the SDK.

10. Add HTML and pdf versions of the specification
    - The SDK already includes both HTML and PDF versions of the specification

11. Capture Replay are missing functionality and constantly crash on our projects - there is no single one tool that can do it all and work at the same time
    - If the capture/replay being mentioned in this comment was referring to vktrace/vkreplay, then yes, it had quality and incompleteness issues. Vktrace/vkreplay has been deprecated and replaced with the redesigned and greatly improved GFXReconstruct. The GFXReconstruct tool was architected with vast improvements after learning from the mistakes and deficiencies of vktrace/vkreplay. It is now included in the Vulkan SDK and is under active sponsorship from multiple IHVs due to its robustness.
    - Key features of GFXReconstruct:
        - i. Improved Android capability & reliability
            1. Android is given same priority as desktop in features and support
        - ii. Automatic code generation to accommodate evolving API
        - iii. Improved trimming
        - iv. Increased portability
            1. X86 vs. x64 differences
            2. Cross-OS portability (i.e. trace on windows, replay on linux with compatible GPU/driver)
            3. Cross-vendor GPU support (trace on one GPU, replay on another)
                - a. “Memory translation” functionality for handling different capture/replay memory types
                - b. Use dev-sim layer to disable vendor specific extensions/features
            4. Compression for trace data
            5. LZ4
            6. Zstandard
        - v. Extract/replace shaders

12. Validation Layer Error Reporting Improvements:
    - The following Validation Layer error reporting enhancements have been added:
        i. VUID annotated specifications have been included with the SDK along with the validation layers linking directly to these VUID annotated specifications.
        ii. The validation layers have been updated and scrubbed to ensure that any object output in the text of a message is now also passed to the user through the callback, allowing programmatic access to these objects for debugging purposes.
        iii. Unique message identifiers have been added to the validation layers, allowing simpler handling of specific messages in message callbacks or debugging utilities.
        iv. Plain-text Object Types and Names were added to the validation layer error reporting
        v. A layer option has been added that allows specific validation messages to be suppressed.
        vi. A Message Repetition Limit was added to the validation layers to limit the number of times any single validation message would be reported.
    - You can read about these in detail in this whitepaper.

13. Validation Layer Performance Improvements
    - LunarG has begun a proactive performance initiative with some dedicated resources. Performance tuning in the past was "reactive" in response to a specific complaint from an application developer. This proactive performance initiative is taking a different approach:
        i. Creating a performance regression test suite that is run nightly to look for daily check-ins that result in negative performance impacts.
        ii. Performance analysis of many, many workloads to identify performance bottleneck themes to identify priority areas for tuning.
        iii. Compile-time options to the validation layers are being developed which, when coupled with existing performance analysis tools and other Vulkan ecosystem components, will allow more rapid discovery and location of validation layer performance regressions and bottlenecks.

14. Shader debugging
    - RenderDoc provides shader debugging capabilities.
    - As well, there is a debug printf validation layer capability that can be used to help debug shaders. Information on using this feature is included in the SDK documentation.

15. A full blown SDK like the Metal Frameworks or DirectX TK AND Please don't add more to the SDK, gets bloated
    - These two requests are in conflict with each other. However the current design of the Windows SDK as "one big blob" makes it difficult to add features without getting too large. As a result we are in the middle of an SDK re-design that will enable a more modular approach to delivering SDK components that can be user selectable.

Ecosystem Requests still pending

1. Validation Layer Performance improvements
    a. Work on this has started (see previous section), but much is still left to do before end users will realize the benefits.
2. Add the Vulkan Memory Allocator to the SDK
    a. This is on the TODO list. Will be coming in the future.
3. Add a Visual Profiling tool/layer
    a. Creating a cross platform/cross GPU profiler is on LunarG's radar. As other projects complete, resources will be directed towards this initiative.
4. A tool for analyzing synchronizations
    a. Currently we are focused on adding validation for synchronization. Additional tools could become available in the future.
5. MoltenVk extensions and validation layers are mutually exclusive.
    a. MoltenVK provides extensions to access Metal features (such as allowing artifact interop between Vulkan and Metal, or support for passing MSL directly to vkCreateShaderModule within layers) that are not included as extensions in the Vulkan specification resulting in the inability for the Validation Layers to execute properly when these MoltenVK extensions are used by applications. The Portability group, MoltenVK maintainer, and Vulkan Working Group are working to determine a solution to resolve this incompatibility. This MoltenVK issue can be used to track its progress.
6. More complicated samples, More documentation and tutorials
    a. The Khronos Working Group members and industry contributors continue to contribute in this space. LunarG is currently not contributing in this space to reserve precious resources for other ecosystem initiatives.