Best Practices for Using and Contributing to the Vulkan Validation Layers

SIGGRAPH 2019

Introductions

- **If you develop Vulkan applications, Validation Layers are your friend!**
  - Validates correct API usage by the application
  - Critical Khronos-branded Vulkan ecosystem component
- **Project leads for the Khronos Vulkan Validation Layers**
  - Google:
    - Tobin Ehlis, Cody Northrop
  - LunarG
    - Mark Lobodzinski, John Zulauf
- **These slides are posted at:**
  https://www.lunarg.com/siggraph-2019-lunarg-presents-vulkan-ecosystem-topics/

Agenda

- **We really want this session to be interactive**
  - Will start with some context setting presentation:
    - How to configure Validation Layers
    - Validation Layer Status Update
      - Unified Validation Layer
      - GPU-Assisted Validation
      - Synchronization Validation
    - Best practices for validation layer contributions

- **Q&A - we are here to answer your questions**

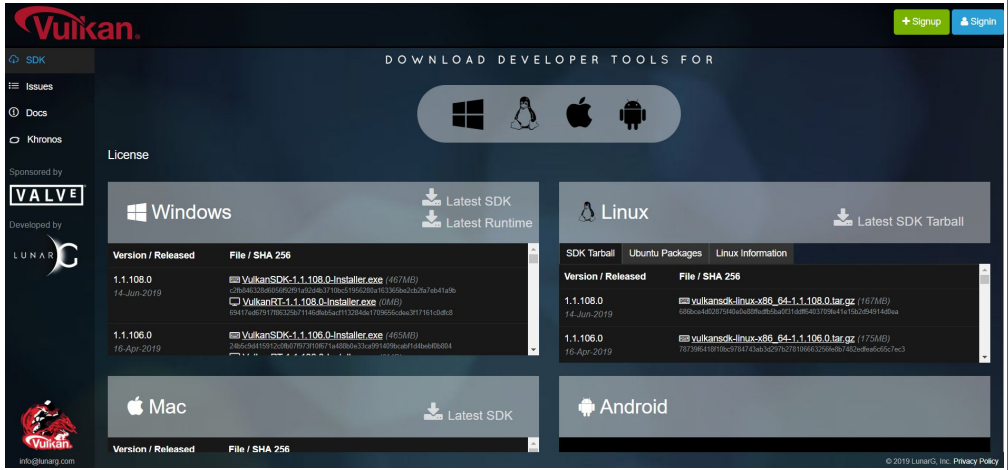
Audience Poll

- Who has used the validation layers?
  - Anybody planning to use them in the future?
- Who uses them on desktop? On Android?
- Who has contributed to the validation layers?
  - Anybody planning on contributing in the future?

What are Vulkan Validation Layers?

- Vulkan drivers by design do no error checking
- Validation Layers verify correct Vulkan API usage
- Validation Layers are available from:

The Vulkan SDK: https://vulkan.lunarg.com
for Windows, Linux, macOS



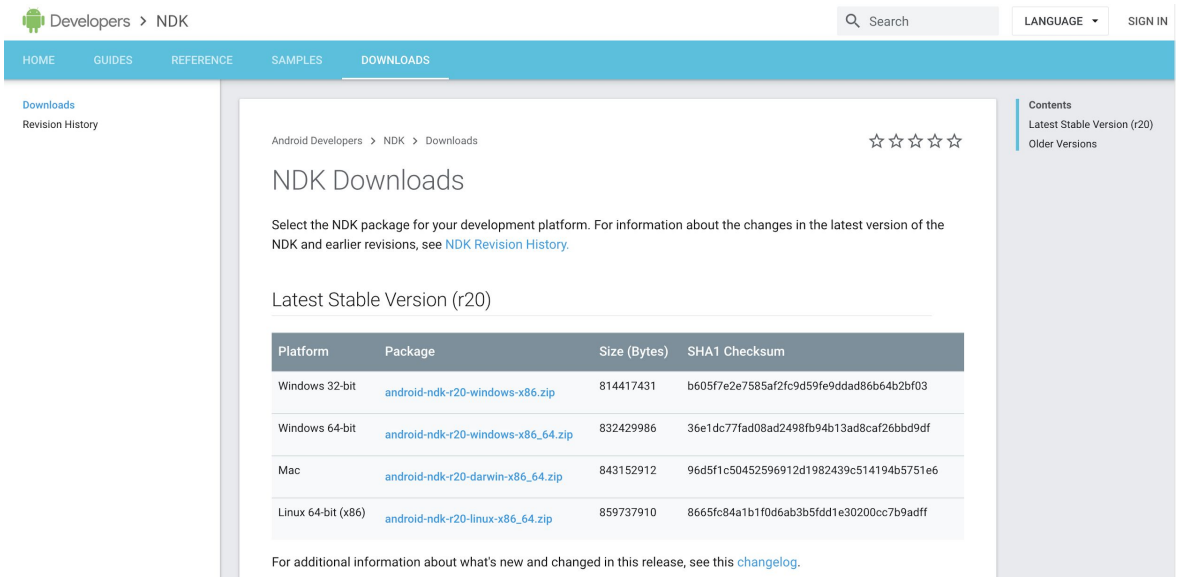
The Khronos Group Validation Layers Github Repository:

https://github.com/KhronosGroup/Vulkan-ValidationLayers

How to get the Validation Layers - Android

- Prebuilt binaries in the Android NDK (Native Development Kit)
- Available via Android Studio:

- Or the web: https://developer.android.com/ndk/downloads



Configuring the Validation Layers - Desktop

- Vulkan Configurator (vkconfig)
  - GUI front-end allowing control of layer loading, order & features
  - Available in SDK or from LunarG VulkanTools repository
- **`vk_layer_settings.txt`** configuration file
  - Text-based configuration file allowing low-level layer control
- **`VK_EXT_validation_features`** extension
  - Allows direct application control of major layer features

Configuring the Validation Layers - Android

- Configure layer list in the application, package them in the APK
- Or over ADB:
  - https://developer.android.com/ndk/guides/graphics/validation-layer
- New in Android Q
  - Load layers from another APK

```
adb shell settings put global enable_gpu_debug_layers 1
adb shell settings put global gpu_debug_app my.vulkan.app
adb shell settings put global gpu_debug_layers VK_LAYER_KHRONOS_validation
adb shell settings put global gpu_debug_layer_app my.validation.layers
```

Unified Validation Layer

- `VK_LAYER_KHRONOS_validation` layer incorporates validation previously implemented in threading, parameter_validation, object_tracker, core_validation, and unique_objects layers
- Legacy layers will be deprecated after the August Android NDK update
- Khronos layer will be extended over time with other types of checks such as synchronization validation and best-practices

Unified Validation Layer

Why?

- **Smaller**
  - more shared source code
- **Better**
  - improved codegen, less duplication
- **Faster**
  - One-third faster than legacy layers
- **Extensible**
  - simplifies adding new layer functionality

GPU-Assisted Validation

- Bindless Descriptor Validation
  - Descriptor from the array is not bound until run time
- Descriptor Indexing Validation
  - **`VK_EXT_descriptor_indexing`** extension relaxes restrictions on descriptor initialization
- Buffer Device Address Validation - in development
  - Shaders directly access device physical storage based on values returned by **`GetBufferDeviceAddress`**

Synchronization Validation (WIP)

- **Real-time validation of Vulkan resource synchronization**
  - Optional feature for VK_LAYER_KHRONOS_validation layer
  - Identify RAW, WAR, and WAW hazards for Vulkan resources
- **Initial Implementation Priorities -- based on developer feedback**
  - Record-time hazard detection within a single command buffer
  - Record-time hazard detection between command buffers within a single queue
  - Submit-time hazard detection between command buffers across/among queues

Contributing Best Practices

- **File bugs!**
  - Be specific, with example code if possible (we love working examples)
  - Answer our questions. Issues with open questions get ignored
  - Be available to test pending PR's (especially when there's no example)

Contributing Best Practices (cont'd)

- **Write Code! -- Coding/design considerations**
  - Be sure you're in the right layer object (Stateless vs. CoreChecks)
  - Code is not stylistically consistent
    - New code should be, beware bad examples
    - Use clang-format
  - State/Checks refactor in process on CoreChecks, look for updates to documentation
    - New code should be careful to segregate validation and state tracking
    - Validation paths should be const clean
  - Validation messages
    - String manipulation only if a message is going to be logged.
    - Use FormatHandle
    - Use string_<typename> stringifiers for Enums, Bitfields, etc.

Contributing Best Practices (cont'd)

- **Contribute Code! -- Pull Requests**
  - Read CONTRIBUTING.md
  - Commit message guidelines -- keywords, style, length
  - Separate layer and test changes in separate commits -- bisectable!
  - Note new *Generated Source Code* guidelines
  - Ensure CI is passing (including format) and that rebase is clean
  - Respond to review feedback
  - Check the git blame and @ tag within the Pull Request

Contribution Statistics

- **As of today**
  - 205 K LOC in 166 files
  - 158 individual contributors to the repo
  - 47 repo watchers, 113 stars, 73 forks

- **July 2019**
  - 438 unique visitors
  - 16 Unique authors w/ 112 CLs in 35 merged PRs
  - 12 active PRs
  - 22 closed/4 new issues

Who is LunarG?

- **3D Graphics Software Consulting Company**
  - Based in Colorado
  - Vulkan, OpenGL, OpenXR, SPIR-V, ...
- **Sponsored by Valve & Google to deliver critical pieces of the Vulkan Ecosystem**
  - Vulkan Loader & Validation Layers
  - Vulkan tools (GFX Reconstruct, apidump, Assistant Layer, …)
  - Vulkan SDK
  - Close collaboration with the Khronos Vulkan Working Group
- **Come learn more about Vulkan at the Khronos BoF day**
  - Wednesday, July 31st
  - J.W Marriott hotel LA Live, Diamond Ballroom 7-10
  - Vulkan sessions beginning at 2PM
  - Networking session with refreshments starts at 5:30PM
    - Visit the LunarG table to get a FREE GIFT!