Vulkan Ecosystem Advancements to Aid Vulkan Developers

SIGGRAPH 2019

Agenda

- Khronos Validation Layer
- GPU-Assisted Validation
- Synchronization Validation Update
- SDK Update
- Graphics Reconstruct

Validation Layer Consolidation

- Validation Layer Consolidation is complete as of the 1.1.106 SDK release
- `VK_LAYER_KHRONOS_validation` layer incorporates validation previously implemented in:

VK_LAYER_LUNARG_object_tracker

VK_LAYER_GOOGLE_unique_objects

VK_LAYER_LUNARG_parameter_validation

VK_LAYER_GOOGLE_threading

VK_LAYER_LUNARG_core_validation

Validation Layer Consolidation

Improvements

- **Revamped infrastructure, more resistance to spec changes, and improved performance**
  - 5000+ line source code size reduction
  - Increased code-generation coverage
  - Generated code now checked into repository
  - `VK_LAYER_KHRONOS_validation` exhibits ~40% performance increase over deprecated layers

White Paper:
https://www.lunarg.com/wp-content/uploads/2019/04/UberLayer_V3.pdf

Validation Layer Consolidation

- **Legacy layers will be deprecated after the August Android NDK update**
- Object_tracker, threading, core_validation, parameter-validation, unique_objects
- `VK_LAYER_LUNARG_standard_validation` **meta-layer now loads *only* Khronos layer**
- `VK_LAYER_LUNARG_standard_validation` **will also be deprecated**
- **Khronos layer will be extended with other types of checks such as synchronization validation and best-practices (Assistant Layer)**

Khronos Validation Layer

Configuring Validation Layer features

- Use Vulkan Configurator (vkconfig, included in Vulkan SDK)
- Vk_layer_settings.txt file
- VK_EXT_validation_features extension
  - Allows enabling/disabling of various bits of layer functionality
    - VK_VALIDATION_FEATURE_DISABLE_**THREAD_SAFETY**_EXT
    - VK_VALIDATION_FEATURE_DISABLE_**API_PARAMETERS**_EXT
    - VK_VALIDATION_FEATURE_DISABLE_**OBJECT_LIFETIMES**_EXT
    - VK_VALIDATION_FEATURE_DISABLE_**CORE_CHECKS**_EXT
    - VK_VALIDATION_FEATURE_DISABLE_**UNIQUE_HANDLES**_EXT
    - VK_VALIDATION_FEATURE_ENABLE_**GPU_ASSISTED**_EXT
    - Other disable knobs

Agenda

- Khronos Validation Layer
- GPU-Assisted Validation
- Synchronization Validation Update
- SDK Update
- Graphics Reconstruct

What is GPU-Assisted Validation?

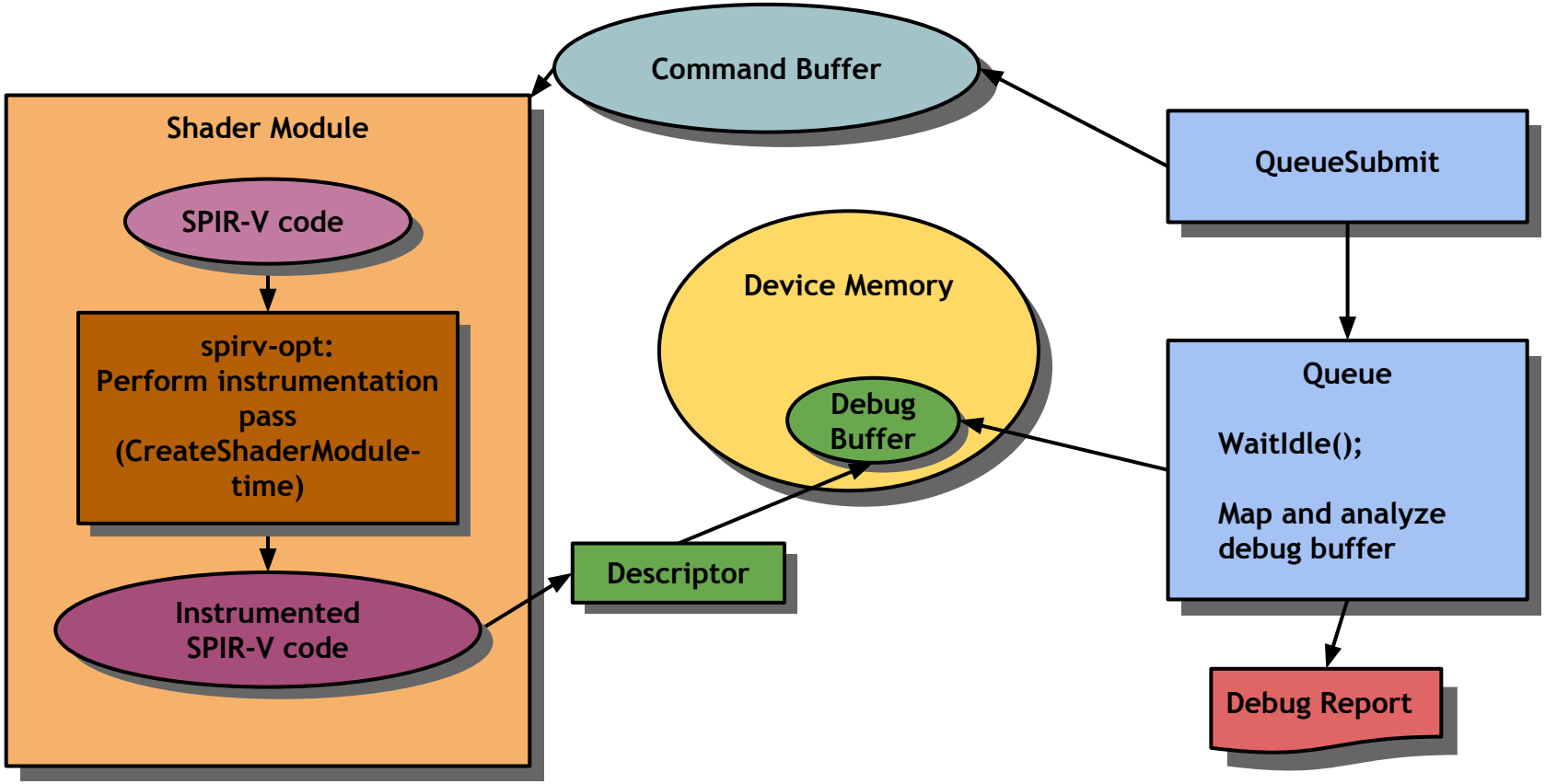
LUNARG

Uses GPU to perform validation at shader execution time

- **Part of Vulkan Khronos validation layer (disabled by default)**
- **With Nvidia's recent addition of instrumentation for the raytracing shaders, only mesh and task shaders are currently unchecked**
- **Simple and straightforward activation**
  - as opposed to other manual and targeted shader debug approaches

KHRONOS GROUP

How GPU-Assisted Validation Works



GPU-Assisted Validation Phases

- Bindless Descriptor Validation - complete
- Descriptor Indexing Validation - complete
- Buffer Device Address Validation - in development

Bindless Descriptor Access Validation

- The inspiration for GPU-assisted validation
- Descriptor from the array is not bound until run time

```
layout (set = 0, binding = 1) uniform sampler2D tex[6];
```

→ Array of descriptors

```
uFragColor = light * texture(tex[4], texcoord.xy);
```

→ Not bindless, bound at compile time

```
uFragColor = light * texture(tex[10], texcoord.xy);
```

→ Not bindless, compile time error

```
uFragColor = light * texture(tex[tex_ind], texcoord.xy);
```

→ Bindless -- descriptor not bound until run-time

Descriptor Indexing Access Validation

- `VK_EXT_descriptor_indexing` extension relaxes restrictions on descriptor initialization
- Phase 2 has added validation for the following cases

Descriptor Indexing Access Validation

runtimeDescriptorArray

The sizes of descriptor arrays can be determined at runtime rather than at shader compile time

descriptorBindingVariableDescriptorCount

An array at the last (highest) binding point can have a variable descriptor count from set-to-set

descriptorBindingPartiallyBound

A descriptor can be partially bound and only those elements accessed by the shader need to have been written

descriptorBindingSampledImageUpdateAfterBind

Descriptors can be written after the descriptor set has been bound, but before the command buffer is submitted to a queue

Buffer Device Address Access Validation

Physical Address = GetBufferDeviceAddressExt(VkBuffer)

Shaders directly access device physical storage based on values returned by GBDA

GPU-Assisted Validation validates that all shader reads/writes based on those physical addresses are in-range of the queried buffers

- **In development -- planned for release in the fall 2019 timeframe**

GPU-Assisted Validation

Activate as any other Khronos layer feature using

- Vulkan Configurator (vkConfig)
- vk_layer_settings.txt config file
- VK_EXT_validation_features extension

GPU-AV Github tracking issue is Vulkan-ValidationLayers #852

White Paper
https://www.lunarg.com/wp-content/uploads/2019/06/GPU-Assisted-Validation-Phase-2_final.pdf

Agenda

- Khronos Validation Layer
- GPU-Assisted Validation
- Synchronization Validation Update
- SDK Update
- Graphics Reconstruct

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

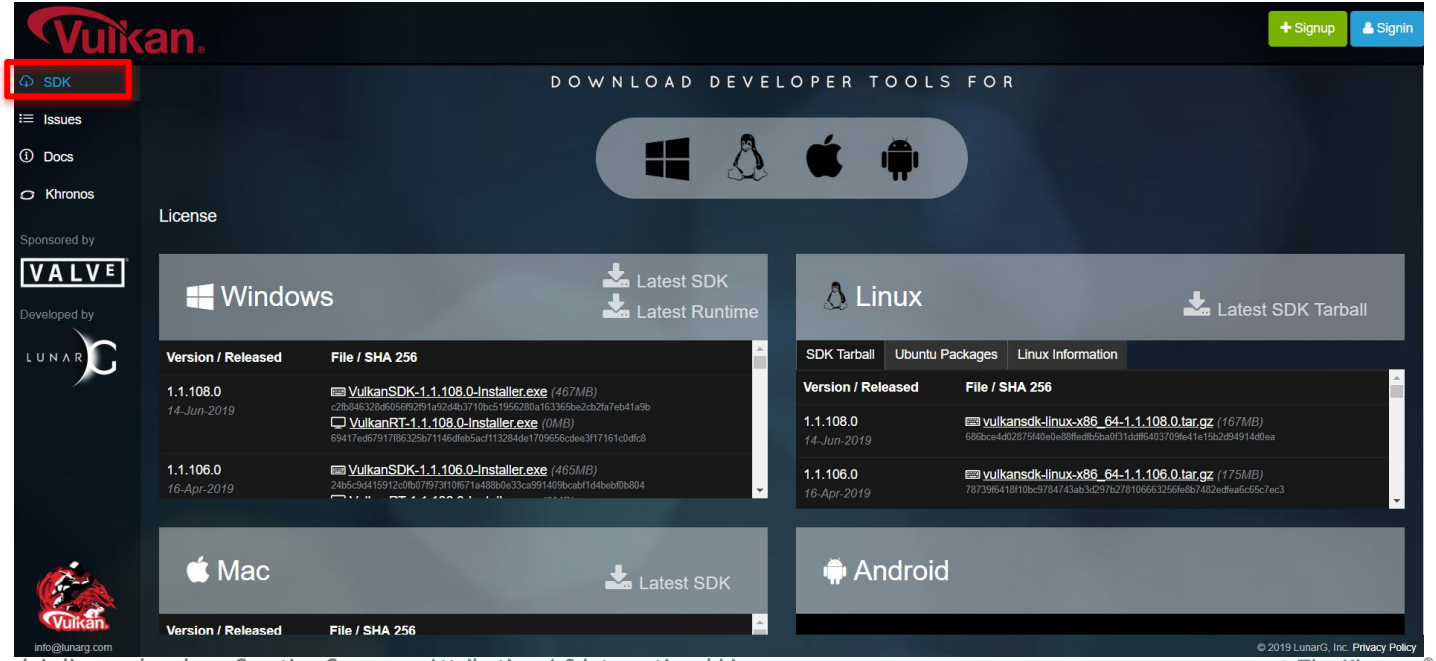
Agenda

- Khronos Validation Layer
- GPU-Assisted Validation
- Synchronization Validation Update
- SDK Update
- Graphics Reconstruct

What is the Vulkan SDK?

- Vulkan application developer tools comprised of 100% open source components
- Available since Vulkan 1.0 launch
- LunarG recently donated the SDK packaging technologies to Khronos
  - Enables Vulkan WG collaboration

Download SDK at: vulkan.lunarg.com (Windows, Linux - Ubuntu packages, Linux- Tarball, macOS):



SDK contents/docs viewable at vulkan.lunarg.com

Vulkan

+ Signup | Signin

SDK
Issues
Docs
Khronos

Sponsored by VALVE

Developed by LUNARG

1.1.108.0 (Latest)

- Getting Started
- Release Notes
- **Loader and Layers**
  - Loader
  - Layers Overview and Configuration
  - Validation Layers
  - GPU Assisted Validation
- **Utility Layers**
  - API Dump
  - Device Simulation
  - Assistant Layer
  - Monitor
  - Screenshot
- **Tools**
  - Vulkan Tools Framework
  - vkconfig
  - Layer Factory
  - VIA
  - vulkaninfo
  - Trace and Replay
  - SPIR-V Toolchain
- **Vulkan Samples**
  - Vulkan Tutorial
  - Build/Run the

info@lunarg.com

Full set of SDK contents and associated documentation

Vulkan

Getting Started with the Vulkan SDK

Version for Windows

This guide describes the requirements and procedure for installing the Vulkan SDK for Windows. It also includes compilation and runtime instructions for demo Vulkan applications. Refer to the Vulkan SDK, Documentation, and Known Issues at the Vulkan SDK Download Site for the most up to date SDK information.

The Vulkan API is a low-overhead, explicit, cross-platform graphics API that provides applications with direct control over the GPU, maximizing application performance. For more information on the Vulkan specification and API, refer to Khronos.org. For tutorial-level information, refer to the Vulkan tutorial, which can be found in the SDK in the `Documentation\Tutorial\html` directory and at the Vulkan SDK Download Site.

This SDK does NOT include a Vulkan driver. Please contact your GPU hardware vendor for a Vulkan Installable Client Driver (ICD). This SDK will allow you to build Vulkan applications but you will need a Vulkan ICD to execute them.

© 2019 LunarG, Inc.

Agenda

- Khronos Validation Layer
- GPU-Assisted Validation
- Synchronization Validation Update
- SDK Update
- Graphics Reconstruct

GFX Reconstruct

- **MUCH improved capture/replay tool**
- **Currently in Beta**
- **Performance Benefits (relative to vktrace/vkreplay)**
  - Up to 2X FPS improvement during capture replay
  - Capture file size reduced up to 50%
- **vktrace/vkreplay will be deprecated in favor of GFX Reconstruct**
  - Fall 2019

https://github.com/LunarG/gfxreconstruct

GFX Reconstruct Benefits

- **Android is given same priority as desktop in features and support**
- **Automatic code generation to accommodate evolving API**
- **Reliable trimming**
- **Increased portability**
  - X86 vs. x64 differences
  - Cross OS portability (i.e. capture on windows, replay on linux).
  - *Cross vendor GPU support (capture on one GPU, replay on another)*
- **LZ4 compression for capture data**
- ***Future valuable plug-ins with minimal code changes***
  - Generate C code program
  - Data mining utilities (search for feature usage)
  - Extract/replace shaders

******Items in Italics may not be ready until after vktrace/vkreplay deprecation*

Who is LunarG?

- **3D Graphics Software Consulting Company**
  - Based in Colorado
  - Vulkan, OpenGL, OpenXR, SPIR-V, …
- **Sponsored by Valve and 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 visit with us at the Khronos networking reception that begins at 5:30**
  - Share your feedback!
  - Ask your questions!
  - Get a Free Gift!

Backup

Synchronization Validation Update

- **Incremental Approach**
  - Synchronization validation is large and challenging
  - Progressively larger use case coverage
  - Balance coverage with performance impact and need to avoid false-positives
- **Configuration options -- programmatic control**
  - Level of hazard detection (single command buffer, etc.)
  - Resources/queues/command buffers of interest