Vulkan Ecosystem Advancements to Aid Vulkan Developers

SIGGRAPH 2019
Agenda

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

- `VK_LAYER_LUNARG_standard_validation` meta-layer now loads only Khronos layer
- 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` will also be deprecated
- Khronos layer will be extended with other types of checks such as synchronization validation and best-practices (Assistant Layer)
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

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
What is GPU-Assisted Validation?

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, the only shaders currently unchecked are mesh and task shaders
- Simple and straightforward activation
  - as opposed to other manual and targeted shader debug approaches
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 runtime
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.
Descriptor Indexing Access Validation

GPU-Assisted Validation detects out of bounds indexing of descriptor arrays and use of unwritten descriptor elements when the descriptor indexing features are used.
Buffer Device Address Access Validation

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

Physical Address = GetBufferDeviceAddressExt(VkBuffer)

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
How GPU-Assisted Validation Works

Shader Module
- SPIR-V code
  - spirv-opt: Perform instrumentation pass (CreateShaderModule-time)
- Instrumented SPIR-V code

Command Buffer

Device Memory
- Debug Buffer
- Descriptor

QueueSubmit
- Queue
  - WaitIdle();
  - Map and analyze debug buffer

Debug Report
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

White Paper

Agenda

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

Full set of SDK contents and associated documentation
GFX Reconstruct (https://github.com/LunarG/gfxreconstruct)

- MUCH improved capture/replay tool
- Currently in Beta mode
- 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
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!
  - Visit the LunarG table to 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