Vulkan Update

Tom Olson
Distinguished Engineer, Arm
Chair, Vulkan Working Group
Vulkan: An API for Direct GPU Control
Vulkan: An API for Direct GPU Control

Complex drivers cause overhead and inconsistent behavior across vendors
- Difficult to optimize
- Error handling is always active
- Full GLSL preprocessor and compiler in driver
- Different APIs for desktop / mobile

High-level Driver Abstraction
- Error detection
- Context management
- Memory allocation
- Full GLSL compiler
- Synchronization

Application
- Single thread per context

A Graphics API
Vulkan: An API for Direct GPU Control

Complex drivers cause overhead and inconsistent behavior across vendors
Difficult to optimize
Error handling is always active
Full GLSL preprocessor and compiler in driver
Different APIs for desktop / mobile

Application
Single thread per context

High-level Driver Abstraction
Error detection
Context management
Memory allocation
Full GLSL compiler
Synchronization

GPU
A Graphics API

Application
Memory allocation
Thread management
Explicit Synchronization
Multi-threaded generation of command buffers

Front-end Compiler(s)
GLSL, HLSL etc.

Thin Driver
SPIR-V compiler back end

GPU
A GPU API

Loadable debug and validation layers

Simpler drivers - application has the best knowledge for holistic optimization - no 'driver magic'
Explicit creation of API objects before usage - efficient, predictable execution
Easier portability - no fighting with different vendor heuristics
Validation and debug layers loaded only when needed
SPIR-V intermediate language: shading language flexibility
Unified API across mobile and desktop platforms
Multithread / Multicore friendly
Vulkan AAA Content
Vulkan Mobile Content
New Functionality Highlights
Vulkan 1.2 Released in January

- Rolls up 23 previous released extensions into a new core API
  - Improved performance, enhanced visual quality and easier development
  - Pushes the industry toward a unified feature set and development target

Requests from Vulkan Developers

<table>
<thead>
<tr>
<th>VK_KHR_timeline_semaphore</th>
<th>- more manageable synchronization</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_EXT_descriptor_indexing</td>
<td>- reusing descriptor layouts for multiple shaders</td>
</tr>
<tr>
<td>VK_KHR_buffer_device_address</td>
<td>- bindless resources</td>
</tr>
<tr>
<td>VK_KHR_imageless_framebuffer</td>
<td>- framebuffer definition without images</td>
</tr>
<tr>
<td>VK_EXT_host_query_reset</td>
<td>- easier resetting of queries</td>
</tr>
</tbody>
</table>

API Usability Improvements

<table>
<thead>
<tr>
<th>VK_KHR_driver_properties</th>
<th>- reports latest passing CTS version</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_KHR_create_renderpass2</td>
<td>- more extensible renderpass objects</td>
</tr>
<tr>
<td>Vulkan 1.1/1.2 unified feature and property structs</td>
<td></td>
</tr>
</tbody>
</table>

Exposing New Hardware Capabilities

<table>
<thead>
<tr>
<th>VK_KHR_image_format_list</th>
<th>- improve image view performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>framebufferIntegerColorSampleCounts</td>
<td>- more multi-sample formats</td>
</tr>
<tr>
<td>VK_KHR_sampler_mirror_clamp_to_edge</td>
<td>- widely supported mode</td>
</tr>
<tr>
<td>VK_EXT_sampler_filter_minmax</td>
<td>- for newer GPUs</td>
</tr>
<tr>
<td>VK_KHR_shader_viewport_index_layer</td>
<td>- for newer GPUs</td>
</tr>
<tr>
<td>VK_KHR_shader_float16_int8</td>
<td>- proper fp16 support</td>
</tr>
<tr>
<td>VK_KHR_shader_float_controls</td>
<td>- control over rounding, etc.</td>
</tr>
<tr>
<td>VK_KHR_vulkan_memory_model</td>
<td>- precise memory model spec</td>
</tr>
<tr>
<td>VK_KHR_shader_subgroup_extended_types</td>
<td>- more subgroup types</td>
</tr>
<tr>
<td>VK_KHR_8bit_storage</td>
<td>- 8-bit types in SSBOs/UBOs</td>
</tr>
<tr>
<td>VK_KHR_shader_atomic_int64</td>
<td></td>
</tr>
<tr>
<td>VK_KHR_depth_stencil_resolve</td>
<td>- resolve modes for depth/stencil</td>
</tr>
</tbody>
</table>

Vulkan 1.2 deliberately does not mandate new hardware functionality so that all Vulkan GPU drivers are able to be upgraded

Vulkan 1.2 deliberately does not mandate new hardware functionality so that all Vulkan GPU drivers are able to be upgraded
Vulkan Ray Tracing *(provisional specifications)*

**VK_KHR_ray_tracing**
- Build acceleration structures (AS) on host or GPU
- Cast rays via ray tracing pipelines or ray queries

**VK_KHR_pipeline_libraries**
- Assemble RT pipelines from separately compiled stages

**VK_KHR_deferred_host_operations**
- Let the driver parallelize AS or pipeline construction
- Application provides the thread pool

**Status**
- Good community / ecosystem feedback received
- Making good progress toward final specifications
Variable Rate Shading enables focusing of rendering power for more perf/less power.
Shading Rate selects how many pixels’ color values are affected by each fragment.
‘Spreads’ a fragment between 1 to 4 times independently on both X and Y axis.

Specifying Shading Rate:
- **Rate Per Draw Call**
  - Lower rate for background or low-detail objects
- **Rate Per Triangle** (optional)
  - Lower rate for low-detail primitives
- **Rate Per Region** (optional)
  - Lower rate for periphery

**COMBINE**
- Keep or Replace or Min or Max (per axis) or Mul (per axis)

Final Rate

Lower-resolution image (render pass attachment) defines a shading rate for each rectangular region.

Color overlay indicates region shading rates:
- Image courtesy of NVIDIA
Other new functionality

Ease of use
VK_EXT_extended_dynamic_state

Debug, tooling, development
VK_EXT_device_memory_report
VK_KHR_shader_non_semantic_info
VK_EXT_tooling_info
VK_KHR_performance_query
VK_EXT_private_data
VK_EXT_pipeline_creation_cache_control
VK_EXT_pipeline_executable_properties
VK_EXT_shader_clock

Rendering functionality
VK_EXT_line_rasterization
VK_EXT_fragment_density_map2
VK_EXT_texture_compression_astc_hdr

D3D / GL / WebGPU emulation
VK_EXT_robustness2
VK_EXT_image_robustness
VK_EXT_custom_border_color
VK_EXT_4444_formats
VK_KHR_shader_terminate_invocation

Odds and ends
VK_EXT_subgroup_size_control

Improving extension documentation
• Recent extensions include a statement of intended use in the Overview
• Extensions now have their own man (7) pages
• Additional information for some extensions is in the Vulkan Guide
Ecosystem
Vulkan SDK Advances

Vulkan Synchronization Validation!
• Phase 1 released in August: memory hazard detection within a command buffer

Reimagined Vulkan Configurator (vkconfig)
• Discover, enable, and configure Vulkan Layers interactively

GFXReconstruct
• New, improved trace / replay tool: Cross-OS, compression, trimming, ...

Better error reporting in the Validation Layers
• Error messages cross-link to VUID tags in the Vulkan Specification

Ecosystem Survey
• Your feedback produces results! See the 2019 Year End Report.
• 2020 Survey results coming soon!

Vulkan Developer Resources

www.khr.io/vked
  • A curated list of educational resources

www.khr.io/awesomevk
  • Another good list of resources

www.khr.io/vksamples
  • Khronos/community managed collection of high-quality sample code

Recent blogs / videos:
  • Jet Set Vulkan: Reflecting on the move to Vulkan
  • Porting Detroit: Become Human from PlayStation® 4 to PC - Part 1
  • Vulkan Synchronization Validation Quick Start Guide
  • Optimizing Roblox: Vulkan Best Practices for Mobile Developers
The Vulkan Working Group