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

Application
- Single thread per context

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

GPU
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

A Graphics API

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

Application
- Single thread per context

GPU

A GPU API

**Thin Driver**
- SPIR-V compiler back end

Loadable debug and validation layers

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

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

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

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

This work is licensed under a Creative Commons Attribution 4.0 International License
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
- VK_KHR_timeline_semaphore - more manageable synchronization
- VK_EXT_descriptor_indexing - reusing descriptor layouts for multiple shaders
- VK_KHR_buffer_device_address - bindless resources
- VK_KHR_imagelessFramebuffer - framebuffer definition without images
- VK_EXT_host_query_reset - easier resetting of queries

API Usability Improvements
- VK_KHR_driver_properties - reports latest passing CTS version
- VK_KHR_create_renderpass2 - more extensible renderpass objects
- Vulkan 1.1/1.2 unified feature and property structs

Improved Layering Support for Other 3D APIs
- VK_KHR_uniform_buffer_standard_layout - support HLSL constant buffer layouts
- VK_EXT_scalar_block_layout - more layout support for HLSL
- VK_KHR_draw_indirect_count - for OpenGL
- VK_EXT_separate_stencil_usage - to streamline DX ports
- VK_KHR_separate_depth_stencil_layouts - to streamline DX ports
- SPIR-V 1.4 - many HLSL features
- SPIR-V 1.5 - to support Vulkan 1.2

Exposing New Hardware Capabilities
- VK_KHR_image_format_list - improve image view performance
- framebufferIntegerColorSampleCounts - multi-sample formats
- VK_KHR_sampler_mirror_clamp_to_edge - widely supported mode
- VK_EXT_sampler_filter_minmax - for newer GPUs
- VK_EXT_shaderViewportIndexLayer - for newer GPUs
- VK_KHR_shader_float16_int8 - proper fp16 support
- VK_KHR_shader_float_controls - control over rounding, etc.
- VK_KHR_vulkanMemoryModel - precise memory model spec
- VK_KHR_shader_subgroup_extended_types - more subgroup types
- VK_KHR_8bit_storage - 8-bit types in SSBOs/UBOs
- VK_KHR_shader_atomic_int64
- VK_KHR_depth_stencil_resolve - resolve modes for depth/stencil

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.

Color overlay indicates region shading rates (Image courtesy of NVIDIA)

- **Rate Per Draw Call** (optional)
  - 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.

Specifying Shading Rate
Other new functionality

Ease of use
- VK_EXT_extended_dynamic_state

Rendering functionality
- VK_EXT_line_rasterization
- VK_EXT_fragment_density_map2
- VK_EXT_texture_compression_astc_hdr

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

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