Vulkan Fast Forward
New Features and Directions

Tom Olson
Distinguished Engineer, Arm
Chair, Vulkan Working Group
Where is Vulkan shipping today?

Desktop

Cloud Gaming

Mobile

Game Platforms
Vulkan on Android

- Rollout was initially slow

- Good news! Vulkan 1.1 is mandatory in recent 64-bit devices

- Vulkan is now supported on 2/3 of Android devices

- A big improvement!
What is Vulkan being used for?

Silly question...
Vulkan Games Shipping on Desktop

Over 160 Vulkan Titles shipping across PC, Linux, Stadia, and MacOS with Molten VK
Vulkan Games Shipping on Mobile

Vulkan’s lower CPU overhead enables better performance and/or lower power

Vulkan is the default renderer for mobile projects on Unity
It’s not just about games...

Adobe Premiere Rush
- A pro-quality, cross-platform video capture and editing system

Vulkan powered on Android
- On select phones
- Several hundred thousand lines of OpenCL C compute shaders
- Translated to SPIR-V by clspv
- Run in Vulkan command buffers

Eric Berdahl at SIGGRAPH 2018: https://youtu.be/5rxQ77nZits?t=21980
It’s not just about games...

Autodesk Fusion 360 uses Vulkan for cross-platform post-processing and display of simulation results.

See https://www.khronos.org/blog/vulkan-for-cloud-based-transient-compute
The Vulkan Ecosystem
The Vulkan Ecosystem

Vulkan CTS

debug/perf tools

debug extensions
IHV tools

Vulkan SDK

documentation
support
sample code
tutorials
best practices

language toolchain

Vulkan

specs

validation layers

spirv-opt
spirv-val
spirv-dis
spirv-cross
DXC
shaderc
glslang

implentations

spirv-cross

The Vulkan Ecosystem

spirv-val

support
tutorials
sample code

specs

validation layers
New website: https://www.vulkan.org

- Home page for Vulkan on the web
- Make tools and resources easier to find
- Highlight new Vulkan content
- Updated regularly
Download these essential development tools

Essentials tools, documentation and libraries for every Vulkan developer

- Vulkan SDK for Windows, Linux, and macOS

LunarG has developed the quintessential developer SDK including build tools, documentation, libraries and more.

FIND OUT MORE

- Android NDK
- Arm Mali SDK
- PowerVR SDK
New Youtube Channel

- www.youtube.com/vulkan

Videos from Vulkan events
- GDC
- CEDEC
- SIGGRAPH
- Vulkanised!
- Reboot Develop
- …
Vulkan Samples Repository

- A home for Vulkan sample code
  - Intended to help you learn to use Vulkan effectively
  - GPU, OS, and platform neutral, tested on a wide variety of implementations
  - Open Source under the Apache 2.0 license

- A community effort with participation from
  - Sample code developers, including Sascha Willems and others
  - Khronos member ISVs and IHVs

- Developed on GitHub - all are welcome to contribute and participate
Vulkan Samples available today

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

• Performance samples
  - 16-bit storage InputOutput
  - 16-bit arithmetic
  - Async compute
  - Basis universal supercompressed GPU textures
  - AFBC
  - Command buffer management
  - Constant data
  - Descriptor and buffer management
  - Impact of vkDeviceWaitIdle()

- Layout transitions
- Load/store operations
- MSAA
- Multi-threading
- N-buffering and presentation modes
- Pipeline barriers
- Pipeline cache
- Pre-rotation
- Specialization constants
- Subpass merging and G-buffer size
Vulkan Samples available today

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

- **API examples**
  - Compute shader N-body simulation
  - Dynamic uniform buffers
  - High Dynamic Range rendering
  - Instanced mesh rendering
  - Separate image/sampler
  - Dynamic terrain tessellation
  - Texture loading and display
  - Runtime mipmap generation
  - Vulkan-HPP

- **Extension examples**
  - VK_KHR_buffer_device_address
  - VK_EXT_descriptor_indexing
  - VK_KHR_timeline_semaphore
  - VK_KHR_fragment_shading_rate
  - VK_KHR_ray_tracing_pipeline
  - VK_KHR_ray_query
  - VK_KHR_synchronization2
  - VK_EXT_debug_utils
  - VK_EXT_conservative_rasterization
  - VK_KHR_push_descriptor
  - VK_KHR_external_memory
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

Combining rates:
- COMBINE Keep or Replace or Min or Max (per axis) or Mul (per axis)

Rate Per Region (optional)
Lower rate for periphery

Combining rates:
- 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

This work is licensed under a Creative Commons Attribution 4.0 International License
Vulkan Ray Tracing *(final specifications)*

- Core extension factored into three sub-extensions
- Changes to acceleration structure creation and layout
- Explicit stack size management for ray pipelines
- Consistency, ease of use, and efficiency

Provisional Vulkan Video Extensions

- Seamless integration of video encode / decode into Vulkan
- Expose resource sharing on discrete graphics cards
- Leverage existing Vulkan resources and synchronization mechanisms
Video Extensions - Status

- Provisional version released in April 2021

- Beta implementation at

- Detailed description in this blog post:
  - [https://www.khronos.org/blog/an-introduction-to-vulkan-video](https://www.khronos.org/blog/an-introduction-to-vulkan-video)

- Please give us your feedback!
  - [https://github.com/KhronosGroup/Vulkan-Docs/issues/1497](https://github.com/KhronosGroup/Vulkan-Docs/issues/1497)
VK_KHR_synchronization2

- Replaces pipeline stage flags and access flags with 64-bit types
  - Needed for the ray tracing and video extensions

- Restructures queue submission, event, and pipeline barrier APIs / types
  - Makes the API easier to understand, less error-prone

- Key changes
  - Use an array of structures rather than separate arrays
  - Store barrier pipeline stage masks in the barrier itself
  - Specify pipeline barriers at vkCmdSetEvent time rather than at Wait time
  - Image layout types are contextual, reducing the chance for error

- We strongly recommend using this synchronization API for your future projects
Hot topics in the Working Group

- Debugging VK_DEVELOPMENT_LOST situations
- Reducing hitching during pipeline creation
- Making renderpass / subpass handling less painful
- YOUR TOPIC HERE! Ping us at KhronosGroup/Vulkan-Docs on Github...
Thank You!