Vulkan Update
CEDEC 2021

Tom Olson
Distinguished Engineer, Arm Chair, Vulkan Working Group
What is Vulkan?

• A cross-platform graphics and compute API for modern GPUs
  - Logical successor to OpenGL / ES
  - Focus on real-time applications
  - One API across all markets and platforms

• Key features
  - Programming model matches the hardware
  - Multi-thread and multi-core friendly
  - Error checking and compilation are outside the driver
  - Much lower CPU overhead

• Our bargain with developers
  - You take more responsibility and do more work
  - You get better and more predictable performance
Vulkan is everywhere!

- Desktop OSes
- Streaming platforms
- Game platforms
- Android
- Apple platforms (subset, via shim libraries)
Vulkan support on Android

Vulkan is on 66% of Android devices - and growing fast!

https://developer.android.com/about/dashboards

<table>
<thead>
<tr>
<th>Vulkan version</th>
<th>January 2021</th>
<th>July 2021</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>42%</td>
<td>34%</td>
</tr>
<tr>
<td>Vulkan 1.0</td>
<td>22%</td>
<td>19%</td>
</tr>
<tr>
<td>Vulkan 1.1</td>
<td>36%</td>
<td>47%</td>
</tr>
</tbody>
</table>
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*
Not Just Games!

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

Vulkan powered on Android
- Select Samsung, Google, Sony, OnePlus, and Xiaomi phones
- Several hundred thousand lines of OpenCL C compute shaders
- Translated to SPIR-V by clspv
- Run in Vulkan command buffers

A testament to the power of SPIR-V!
Not Just Games!

Instrument Cluster Rendering – Vulkan is 138% Faster than OpenGL ES

- Various instrument cluster reference designs by Basemark, powered by Basemark’s Rocksolid Engine
- Rocksolid Engine has optimized OpenGL ES and Vulkan rendering pipelines
- Running on the same hardware, Vulkan brings substantial performance benefits compared to OpenGL ES

Basemark’s RockSolid engine uses Vulkan for high performance instrument cluster rendering on automotive GPUs
Not Just Games!

Holochip’s ADE uses Vulkan for cross-platform rendering of ray-traced scenes from multiple viewpoints.

Scenes can be presented using OpenXR on VR/AR devices, light field displays, and 2D monitors.
Not Just 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
New Functionality
Final Vulkan Ray Tracing Extensions

- See the next talk!

Without Ray Tracing

With Ray Tracing
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 here:
  - https://www.khronos.org/blog/an-introduction-to-vulkan-video

- Please give us your feedback!
  - 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
How to start using VK_KHR_synchronization2

• The Vulkan SDK provides an emulation layer
  - Advertises support even if the underlying ICD does not
  - Implements synchronization2 using old-style synchronization APIs
  - Very light-weight and efficient

• The new API is mostly backward compatible
  - Can mix and match new and old types freely
    - EXCEPT: must use SetEvent with WaitEvents, and SetEvent2 with WaitEvents2
  - You can convert your code a bit at a time

• Lots of information to help you
The Vulkan Ecosystem
The Vulkan Ecosystem

- Vulkan CTS
- Vulkan SDK
- documentation
- support
tutorials
best practices

debug/perf tools
- debug extensions
- IHV tools

language toolchain
- specs
- validation layers
- implementations
- Vulkan
- SPIR

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

- Vulkan
- spir-v
- spir-opt
- spir-dis
- DXC
- shaderc
- spirv-cross

This work is licensed under a Creative Commons Attribution 4.0 International License
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

https://www.vulkan.org
Videos

- New Youtube channel
- 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

- Three classes of samples
  - API Samples - Demonstrate how to use the API
  - Performance Samples - Show the impact of good and bad choices
  - Extension Samples - Show how to use new functionality

- Collected / maintained by Khronos Dev Rel together with
  - Community members including Sascha Willems and others
  - Khronos member ISVs and IHVs
  - Developed on GitHub - all are welcome to contribute and participate
Frame Times: 16.7 ms

Vertex Compute Cycles: 98.0 M/s

Fragment Cycles: 667.3 M/s

Pipeline barrier stages:
- Bottom to top
- Frag to vert
- Frag to frag
Choosing the right number of swapchain images

Overview

Vulkan gives the application some significant control over the number of swapchain images to be created. This sample analyzes the available options and their performance implications.

Choosing a number of images

The control over the number of swapchain images is shared between the application and the platform. The application can ask for a minimum number of images by setting the `minImageCount` parameter in `vkCreateSwapchainKHR`. The exact number of images created can then be polled via `vkGetSwapchainImagesKHR`.

In order to properly set the `minImageCount` parameter, the application should get the surface capabilities of the physical device via `vkGetPhysicalDeviceSurfaceCapabilitiesKHR`. The `VKSurfaceCapabilitiesKHR` structure has the `minImagecount` and `maxImagecount` parameters which set the boundaries for the image count that can be safely requested.

As a rule of thumb on mobile, `yoursurfcapabilities.minImagecount` is usually 2, while `yoursurfcapabilities.maxImagecount` is large enough to not pose any problem with common applications (though it is still good practice to check its value).

The most common values that an application may ask for are:

- 2 for double buffering
- 3 for triple buffering

The swapchain will then create a number of images based on both `minImagecount` and the requested present mode. We will discuss present modes in the next section.

Choosing a present mode

The available present modes can be queried via `vkGetPhysicalDeviceSurfacePresentModesKHR`.

There are several presentation modes in Vulkan, but mobile GPUs do not support the ones in which the image is displayed directly on the screen (immediate mode). The only ones which satisfy Android's VSync requirement are `VK_PRESENT_MODE_IMMEDIATE_KHR` and `VK_PRESENT_MODE_MAILBOX_KHR`.

- `IMMEDIATE_KHR` mode the presentation requests are stored in a queue. If the queue is full the application will have to wait until an image is ready to be acquired again. This is a normal operating mode for mobile, which automatically locks the framework to 60 FPS.
- `MAILBOX_KHR` is a single presentation request stored by the presentation engine. In case of a new request the previous image will be replaced and will be available for acquisition once again.

As a rule of thumb, if we ask for `VK_PRESENT_MODE_MAILBOX_KHR` we may get more images than `vkPresentKHR`, typically 4. The application can keep submitting new frames for presentation, without stalling. This is useful in some cases, e.g. for reducing input latency, but it is not optimal.
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
  - Dynamic terrain tessellation
  - Texture loading and display
  - Runtime mipmap generation

• **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_synchronization2
  - VK_EXT_debug_utils
  - VK_EXT_conservative_rasterization
  - VK_KHR_push_descriptor
  - VK_KHR_external_memory
Vulkan Samples available today

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

- **Performance examples**
  - 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
ASTC Encoder (ASTCenc)

- ASTC is a multi-format, multi-bit-rate, high quality texture compression format
  - Exposed as an extension in OpenGL ES 3.1
  - LDR subset is required in OpenGL ES 3.2
  - Supported on essentially all Vulkan-enabled mobile devices

- ASTCenc is Arm’s reference codec
  - Developed by Arm GPU R&D as part of format development
  - Optimized for exploring the design space, not for performance
  - Never intended to be a product

- Available on GitHub since 2012
  - Under a forbidding EULA
  - This limited adoption of ASTC

- Reboot in 2019
ASTC Encoder Improvements since 2019

• Licensing
  - Now available under Apache 2.0

• Productization
  - Refactored into an embeddable library and a front end
  - Builds robustly across many compilers and Oses
  - Output is now invariant across platforms
  - Proper CI prevents regressions

• Performance
  - Improved code quality
  - Improved search heuristics
  - Extensive vectorization using SSE2, SSE4.1, AVX2, Neon
Results

12x speedup (typical) at a cost of 0.1 dB PSNR

ASTCenc is shipping in the Unity toolchain

ASTC is now the default texture compression format for Unity on mobile

Source code: [https://github.com/ARM-software/astc-encoder](https://github.com/ARM-software/astc-encoder)


Thanks Pete!
SDK: Validation Improvements

- Synchronization validation
  - Phase 1: detect memory hazards within a command buffer (shipping)
  - Phase 2: multiple command buffers (in development)
  - Synchronization2 support (shipping)

- Better validation error reporting
  - Error messages cross-link to VUID tags in the specification

- GPU Assisted Validation
  - Various improvements

- Debug printf from a shader
  - Supported in the layer system - integrated with RenderDoc, vkconfig

- See LunarG BOFs at SIGGRAPH
Validation Layer Performance Improvements

• Performance Regressions tracked in CI
  - Catch problems early

• Analysis of problem areas
  - Locking, image layout validation, large bindless descriptor sets

• Initial optimizations
  - Layout transitions, state and object tracking

• Tracking improvements
  - Varies considerably with application!
  - Currently seeing 20% to 100% speedup since January
Vulkan Configurator (vkconfig)
Thank You!