Vulkanised! Webinar

Exploring the Latest Updates to the Vulkan API and Ecosystem
The latest updates

Vulkan 1.3
• A new baseline for the Vulkan API

Vulkan Roadmap 2022
• Capabilities beyond Vulkan 1.3
• A new baseline for immersive graphics

Vulkan Profiles
• Tools for managing diverging device capabilities
• ‘caps bit hell’
Today’s Session

Reducing Fragmentation in the Vulkan Ecosystem
  • Tom Olson, Arm

Vulkan 1.3 Overview
  • Ralph Potter, Samsung

Roadmap 2022 Profile
  • Tobias Hector, AMD

Android Baseline 2021 Profile
  • Trevor Black, Google

Panel Discussion
  • Moderator: Piers Daniell, NVIDIA
Vulkanised! Webinar

Reducing Fragmentation in the Vulkan Ecosystem

Tom Olson, Arm
Vulkan Working Group chair
Outline

Vulkan Today

Evolution of Vulkan

The fragmentation problem

Vulkan 1.3 and the Vulkan Roadmap

Vulkan Profiles
### Vulkan Today - Adoption

<table>
<thead>
<tr>
<th>Desktop and Mobile GPUs and SOCs</th>
<th>Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD, ARM, Broadcom, Google, Huawei, Imagination, Intel, NVIDIA, MediaTek, Samsung, Qualcomm, Veri Silicon</td>
<td>CINDER, XENKO, GODOT, CRYENGINE, unity, source, Unreal, Serious Engine, UX3D, NetEase Games</td>
</tr>
</tbody>
</table>

**Note:** The version of Vulkan available will depend on platform and vendor

<table>
<thead>
<tr>
<th>Windows and Linux Desktops and Cloud</th>
<th>Mobile</th>
<th>Game Streaming Platforms</th>
<th>Gaming Platforms</th>
<th>Apple Platforms (via translation layer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 10, Windows 11, android</td>
<td>android</td>
<td>NVIDIA GeForce NOW, Stadia, SteamOS, Switch, macOS, iOS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vulkan Today - Adoption on Android

Vulkan is on 71% of active Android devices!

<table>
<thead>
<tr>
<th>Vulkan version</th>
<th>January 2021</th>
<th>July 2021</th>
<th>January 2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>42%</td>
<td>34%</td>
<td>29%</td>
</tr>
<tr>
<td>Vulkan 1.0</td>
<td>22%</td>
<td>19%</td>
<td>17%</td>
</tr>
<tr>
<td>Vulkan 1.1</td>
<td>36%</td>
<td>47%</td>
<td>54%</td>
</tr>
</tbody>
</table>

https://developer.android.com/about/dashboards
A Wide Range of Content

Desktop Games

Mobile Games

Applications
A look back

Ten years of API evolution
From API to Ecosystem

Ten years of increasing commitment

- **DEVELOPER ENGAGEMENT**
- **CONFORMANCE TESTING (CTS)**
- **KHR / EXT VALIDATION**
- **DEV REL**
- **ROADMAP**
- **PROFILES**

1.0 2016
1.1 2018
1.2 2020
1.3 2022
Vulkan 1.0 Design Constraints
(many technical goals: multithreaded, light weight, ...)

Cross-platform

Support a very wide range of hardware
- 4-5 years of desktop GPUs, e.g. NVIDIA Fermi / Kepler / Maxwell / Pascal
- Desktop and mobile - span 1 to 2 orders of magnitude in power / perf / area / cost

Get it done in a year!

How to do this?
Design of Vulkan 1.0

Step 1: Take OpenGL ES 3.1 as the hardware baseline

Step 2: List all the graphics features we want in priority order

Step 3: Specify as many of them as we can in the time available

Step 4: Make the ones ES 3.1 hardware can support mandatory

Step 5: Make the rest optional
  • i.e., hide them behind feature bits
Post-1.0 Extensions

Cleanup, fixes for bugs and API design flaws
- VK_KHR_maintenance1, 2, 3
- VK_KHR_dedicated_allocation

Make the API extensible
- VK_KHR_get_physical_device_properties2
- VK_KHR_get_memory_requirements2

OS integration
- VK_KHR_external_memory*, VK_KHR_external_semaphore*

Some hardware functionality
- VK_KHR_16bit_storage
- VK_KHR_shader_draw_parameters
Design of Vulkan 1.1

Import 23 extensions into core

- Bug and Design flaw fixes become mandatory (19 extensions)
- New hardware features become optional (4 extensions)

Result

- Vulkan 1.0 with the benefit of hindsight
Post-1.1 Extensions

A few more bug fixes
- VK_KHR_create_renderpass2

API and quality-of-life improvements
- VK_KHR_imageless_framebuffer
- VK_KHR_timeline_semaphore
- VK_KHR_shader_float_controls, etc

New hardware functionality!
- VK_KHR_buffer_device_address
- VK_KHR_descriptor_indexing
- VK_KHR_draw_indirect_count
- VK_KHR_scalar_block_layout
- VK_KHR_sampler_mirror_clamp_to_edge, etc
Design of Vulkan 1.2

Should we raise the hardware min spec?
• A further barrier to adoption in Android
• Not cost-effective for wearable, embedded, automotive...
• Would strand older a lot of devices in the field on Vulkan 1.1 permanently
• Conclusion: No

Import 24 extensions into core
• Bug fixes and API improvements become mandatory (14 extensions)
• New hardware features become optional (10 extensions)
Post-1.2 Observations

More and more of the core API was becoming optional

“Turning optional extensions into optional core features doesn’t give developers anything of value” - Jesse Hall, Google

We were also adding LOTS more optional functionality by extension

• Over 180 KHR and EXT extensions

Bad for everyone

• Developers don’t know what features they can rely on
• GPU vendors don’t know what features to provide in what markets
Learning to live with fragmentation

Observation: Fragmentation across Vulkan as a whole is inevitable
  • BUT there is a lot of commonality within individual market segments

Market-specific profiles can capture that commonality
  • Led to the design of the profile system
Profiles vs core versions

Observation: Profiles ease the pressure to raise the HW bar in core
- Can keep the min spec at ES 3.1
- New core versions get API improvements / quality-of-life features
- Profiles will mandate hardware features where widely supported
- Nothing is optional (except extensions)

“Option three, my favorite: put programming model stuff, things that are universally supportable, in core and make them required – things like new versions of SPIR-V, synchronization2, timeline semaphore, et cetera. Use profiles for things that are universally supported in one segment of the ecosystem, but not other segments. Leave the truly experimental toys in extensions.”

- Shannon Woods, Google
Vulkan Profiles

Minimum capabilities across a set of Vulkan implementations
  • Core version
  • List of additional requirements for feature, property, and format support
  • List of required extensions

Or, an alternate version of the core spec with tighter requirements
  • NOT a different API

Semi-orthogonal to the specification
  • A profile does not have to be mentioned in the Vulkan spec
  • Vulkan drivers do not know what profiles they support
  • New profiles can describe the functionality of old devices
Things you can do with profiles

Anything that requires precise communication of device capabilities

Platform recommendations / requirements

Application requirements

Market segment definitions

Roadmap direction
Learning to live with fragmentation, part 2

Observation: Profiles are not enough
  • They don’t address fragmentation within market segments

Need a way to develop and communicate a consensus direction
  • Leads to the Vulkan Roadmap
  • Kudos to Tobias Hector (AMD), who drove this direction

Not easy!

Key design decisions
  • What market segment? What devices?
  • What timeframe?
  • What features are required?
Vulkan Roadmap

What market segment?
• Mid- and high-end devices across smartphone, tablet, laptop, console, and desktop
• Aka “immersive graphics”

What timeframe?
• First milestone: Vulkan Roadmap 2022
• “Every GPU vendor targeting these devices will support these features in at least some devices by late 2022 / early 2023”

What features?
• The feature set for 2023 GPUs is already frozen
• We took a poll
Future Roadmap Milestones

Will be further out in time
- Far enough to influence hardware roadmaps

Will be less precise initially
- Defined in terms of issues to address
- E.g., “address hitching during pipeline compilation”

Will result in profiles as the target timeframe approaches
- When all functionality is precisely defined
Vulkan 1.3 Status

Specification available:  https://khr.io/vulkan13spec

Conformance test: coming at  https://github.com/KhronosGroup/VK-GL-CTS

Language toolchain components are rolling out this week

Vulkan 1.3 SDK planned for mid-month

Conformant drivers! © The Khronos® Group Inc. 2022 - Page 25
Profiles and Roadmap Status

Profiles Repository created
- [https://github.com/KhronosGroup/Vulkan-Profiles](https://github.com/KhronosGroup/Vulkan-Profiles)
- Beta tool support coming in the Vulkan 1.3 SDK

Roadmap 2022 Milestone Profile published
- See VP_KHR_roadmap_2022.json in the Profiles repository
- Also defined in the Vulkan 1.3 Specification
1.3 SDK Highlights (Vulkan Profiles Toolset)

Profiles Tooling for Developers (Beta)
- Vulkan Profiles API Library - header-only C++ library
- Vulkan Profiles Layer
- Vulkan Profiles JSON Schema

SDK Included Profiles (JSON format and human readable format)
- VP_KHR_roadmap_2022
- VP_ANDROID_baseline_2021
- VP_LUNARG_desktop_portability_2021

Devsim layer deprecated and replaced with Vulkan Profiles Layer

Beta Quality

Vulkanised Deep Dive Tutorial coming soon!
1.3 SDK Highlights (Validation Layer Performance)

Significant performance improvement to Validation Layers!

- Validation Layer Fine Grained Locking
  - Global locks for every Vulkan call have been removed
  - Significant cause of performance issues since most Vulkan applications are multi-threaded
  - Tested with many release game titles
    - Improved performance in most titles by ~150%!

- Off by default for first release
  - Can be easily enabled with the Vulkan Configurator
  - Once proven to be stable, will be on by default

Vulkanised Deep Dive Tutorial coming soon!
More Information

Vulkan: https://www.vulkan.org/
- Press Release: https://khr.io/xj
- Final specification: https://khr.io/vulkan13spec
- Spec GitHub Repo: https://khr.io/vulkan13github
- Discord Link for community discussion: https://khr.io/vulkan13discord
- Vulkan 1.3 Blog post: https://www.khronos.org/blog/vulkan-1.3-and-roadmap-2022
- Google Android Baseline 2021 Profile blog: https://www.khronos.org/blog/vulkan-1.3-and-roadmap-2022

Another Vulkanised Webinar coming after SDK release
Will cover SDK features in detail