Vulkan: Forging Ahead!

Tom Olson (Arm), Vulkan Working Group chair
Today’s Presentations

Vulkan: Looking back, and looking forward
- Tom Olson (Vulkan Working Group Chair / Arm)

Vulkan SDK: Where we started, and where we are going
- Karen Ghavam (LunarG)

Vulkan: Crash Diagnostic Layer
- Jeremy Gebben (LunarG)

Slang in Vulkan
- Hai Nguyen (NVIDIA)

EVOLVE - Next Generation Benchmarking
- Jasper Bekkers and Darius Bouma (Traverse Research)

Adding Vulkan to Pixar’s Hydra Storm Renderer
- Henrik Edstrom (Autodesk), Ashwin Bhat (Autodesk), and Caroline Lachanski (Pixar)
Vulkan: Looking back, and looking forward

Tom Olson (Arm), Vulkan Working Group chair
Outline

Vulkan evolution, past and future

What’s new in the Vulkan API

What we’re working on for the future

What’s new in the Vulkan ecosystem
Vulkan evolution, past and future
Ten years ago in Vancouver...

OpenGL / OpenGL ES BOF, SIGGRAPH 2014
2014: The journey begins...

Problems with OpenGL today

- Programming model doesn’t match architecture of modern GPUs
  - Especially in mobile

- Arcane and archaic syntax
  - Twenty years of legacy cruft
  - Needless complexity

- CPU intensive
  - State validation, dependency tracking, error checking
  - Hard to predict where CPU load will occur - varies with implementation

- Not multicore-CPU-friendly
  - Primitive threading model, inconsistently implemented

- Implementation variability
  - Spec looseness, performance variation, driver bugs

Initial goal - fix OpenGL’s problems
The journey begins...

Next Generation OpenGL Initiative

- An open-standard, cross-platform 3D+compute API for the modern era
  - Compatibility break with OpenGL
  - Start from first principles

- Goals
  - Clean, modern architecture
  - Multi-thread / multicore-friendly
  - Greatly reduced CPU overhead
  - Architecture-neutral - full support for tile-based as well as direct renderers
  - Predictable performance
  - Improved reliability and consistency between implementations

- Key design principle: explicit control
  - Application is expected to tell the driver what it wants
A protostar is born!

February 16, 2016: Vulkan 1.0 launch
Open source is AMAZING
You can’t do too much testing
It takes an ecosystem

- Vulkan CTS
- validation layers
- implementations
- specs
- debug/perf tools
- debug extensions
- IHV tools
- Vulkan SDK
- documentation
- support
dev techs
tutorials
best practices
language toolchain
- spirv-opt
- spirv-val
- spirv-cross
- spirv-dis
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-val
- spirv-opt
- spirv-cross
- DXC
- glslang
- shaders
- shaderc
- spirv
- spirv-
How Vulkan evolves
How Vulkan evolves

- 1.0: 2016
- 1.1: 2018
- 1.2: 2020
- 1.3: 2022
How Vulkan evolves

Core Versions

- 1.0 (2016)
- 1.1 (2018)

Extensions

2020-2028
How Vulkan evolves

Core Versions

- 1.0 (2016)
- 1.1 (2018)
- 1.2 (2020)

Extensions
How Vulkan evolves

Core Versions

- 1.0 2016
- 1.1 2018
- 1.2 2020
- 1.3 2022

Extensions
How Vulkan evolves

Core Versions

<table>
<thead>
<tr>
<th>Versions</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2016</td>
</tr>
<tr>
<td>1.1</td>
<td>2018</td>
</tr>
<tr>
<td>1.2</td>
<td>2020</td>
</tr>
<tr>
<td>1.3</td>
<td>2022</td>
</tr>
</tbody>
</table>

Extensions

RM 2022 Profile

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

Extensions

Core Versions

<table>
<thead>
<tr>
<th>Versions</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2016</td>
</tr>
<tr>
<td>1.1</td>
<td>2018</td>
</tr>
<tr>
<td>1.2</td>
<td>2020</td>
</tr>
<tr>
<td>1.3</td>
<td>2022</td>
</tr>
</tbody>
</table>

RM 2022 Profile
RM 2024 Profile
RM 2026 Profile
RM 2028 Profile
RM 2030 Profile
How Vulkan evolves

Core Versions

- 1.0: 2016
- 1.1: 2018
- 1.2: 2020
- 1.3: 2022

Extensions

Roadmap Milestones:
- Roadmap 2022 Milestone
- Roadmap 2024 Milestone
- Roadmap 2026 Milestone
- Roadmap 2028 Milestone
- Roadmap 2030 Milestone

Roadmap Profiles:
- RM 2022 Profile
- RM 2024 Profile
- RM 2026 Profile
- RM 2028 Profile
- RM 2030 Profile
How Vulkan evolves

Versions

<table>
<thead>
<tr>
<th>Core Versions</th>
<th>1.0</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>NEXT 1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2016</td>
<td>2018</td>
<td>2020</td>
<td>2022</td>
<td>2024</td>
</tr>
</tbody>
</table>

Roadmap Milestones

- Roadmap 2024 Milestone
- Roadmap 2026 Milestone
- Roadmap 2028 Milestone
- Roadmap 2030 Milestone
Vulkan Roadmap 2024 Profile

Represents the second milestone on the Vulkan Roadmap
- Captures expected feature set for “immersive graphics” 2024-2026+
Vulkan Roadmap 2024 requirements

Vulkan 1.3 plus Vulkan Roadmap 2022 profile

- Descriptor indexing
- *Real* (non-degenerate) subgroups
- Fragment shaders stores & atomics
- Many useful drawing features

Previously optional features

- `multiDrawIndirect`
- `shaderDrawParameters`
- `shaderImageGatherExtended`
- `shaderInt8, shaderInt16, shaderFloat16`
- `storageBuffer8BitAccess, storageBuffer16BitAccess`
- `shaderRoundModeRTEFloat16/32`
More Vulkan Roadmap 2024 requirements

Implementation minima

- maxBoundDescriptorSets $\geq 7$
- maxColorAttachments $\geq 8$

Required extension:

- VK_KHR_push_descriptor
- VK_KHR_subgroup_uniform_control_flow
- VK_KHR_index_type_uint8
- VK_KHR_line_rasterization
- VK_KHR_load_store_op_none
- VK_KHR_vertex_attribute_divisor
- VK_KHR_map_memory2
- VK_KHR_maintenance5
New extensions required in Roadmap 2024

VK_KHR_shader_expect_assume
• Tell the compiler what typical execution paths look like
• Lets it generate faster code

VK_KHR_shader_subgroup_rotate
• A commonly used special case of permutation - faster on some HW

VK_KHR_shader_float_controls2
• Fine-grained control of floating point behavior
• Applies to many more instructions
• Gives Vulkan parity with OpenCL
New extensions in Roadmap 2024
VK_KHR_maximal_reconvergence / VK_KHR_shader_quad_control

<table>
<thead>
<tr>
<th>Vulkan 1.0</th>
<th>Maximal Reconvergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (non_uniform_cond1) {</td>
<td></td>
</tr>
<tr>
<td>// Block A</td>
<td></td>
</tr>
<tr>
<td>if (non_uniform_cond2) {</td>
<td></td>
</tr>
<tr>
<td>// A then</td>
<td></td>
</tr>
<tr>
<td>} else {</td>
<td></td>
</tr>
<tr>
<td>// A else</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>// Block B</td>
<td></td>
</tr>
<tr>
<td>May not reconverge</td>
<td></td>
</tr>
<tr>
<td>Must reconverge</td>
<td></td>
</tr>
</tbody>
</table>

New extensions in Roadmap 2024

VK_KHR_dynamic_rendering_local_read
  • Allows pipeline barriers within dynamic rendering (sometimes)
  • Allows a later fragment shader to read data written by previous fragments

With VK_KHR_rasterization_order_attachment_access, gives you the functionality of “framebuffer fetch” in dynamic rendering

Blog posts
  • https://www.khronos.org/blog/streamlining-subpasses
### Other New Extensions, NOT in RM 2024

#### Vulkan Video
- VK_KHR_video_encode_queue
- VK_KHR_video_encode_h264
- VK_KHR_video_encode_h265
- VK_KHR_video_maintenance1
- VK_KHR_video_decode_av1

#### Shader / compiler support
- VK_KHR_shader_relaxed_extended_instruction
- VK_KHR_shader_replicated_composites

#### Programming model improvements
- VK_EXT_host_image_copy

#### Tool & debug support
- VK_EXT_layer_settings
- VK_EXT_frame_boundary
- VK_KHR_calibrated_timestamps
- VK_EXT_memory_map_placed

#### Maintenance
- VK_KHR_maintenance6
- VK_KHR_maintenance7

#### Legacy API support
- VK_EXT_legacy_vertex_attributes
Vulkan Video Progress

VK_KHR_video_maintenance1
  • Cleanup and small enhancements

Video encode stack is now final
  • VK_KHR_video_encode_queue
  • VK_KHR_video_encode_h264
  • VK_KHR_video_encode_h265

VK_KHR_video_decode_av1

See blogs:
  • Khronos Finalizes Vulkan Video Extensions for Accelerated H.264 and H.265 Encode
  • Khronos Releases AV1 Decode in Vulkan Video with SDK Support for H.264/H.265...
What we’re working on
The information contained herein constitutes forward-looking statements. There can be no assurance that these statements will prove to be accurate, as actual results and future events could differ materially from those anticipated. The reader is cautioned not to place undue reliance on forward-looking statements.
Please don’t have one of these
Vulkan Roadmap 2026 Milestone

“You’ve got to be very careful if you don’t know where you’re going, otherwise you might not get there”
- Yogi Berra

What we plan for the third milestone on the Vulkan Roadmap
RM2026 themes: Robustness

Why
  • WebGPU will be HUGE
  • Support Rust / Ash

Work Items
  • VK_EXT_robustness2 (all features required)
  • Range-checked pointers (CHERI or something like it)
  • VK_KHR_pipeline_binary (forthcoming)
RM2026 themes: Unifying compute

Why

• Support HPC use cases
• Feature parity with OpenCL

Work items

• Generic memory with physical or virtual addressing
• Replace all buffer references with device addresses
• Untyped pointers
• 64-bit addressing
• Shader Int64 support
• Reorganize SPIR-V capabilities
RM2026 themes: State management

Why
• Ongoing point of pain for developers

Work items
• Shader objects, or something like it
• Pre-validated dynamic state bundles
• Clean, simple API for descriptor management
RM2026 themes: Machine learning

Why
  • Because machine learning

Work items
  • Cooperative matrix
  • Cooperative vector (dispatch small MLPs from a shader)
  • ML-friendly types, (bfloat16 etc)
RM2026 themes: Debugging

Why
• Point of pain for developers

Work items
• VK_EXT_device_fault
• Progress markers (breadcrumbs)
• Shader abort()
RM2026: Other stuff

Fragment shading rate (aka VRS)

VK_EXT_depth_clamp_zero_one

VK_EXT_host_image_copy

Compute shader derivatives (linear)

Maintenance extensions
Vulkan Roadmap 2028 candidates

Ray tracing
- Pretty much everything available today

GPU-driven rendering
- Work graphs, or something like them

Advanced robustness features
- Language features: poison/freeze, data race handling
- Per-device address spaces
Other things in flight (not in Roadmap)

Ray tracing
- KHR versions of vendor extensions: ray reordering, etc

Video
- AV1 encode
- Various advanced encoding tools
- VP9 in plan

Machine Learning
- Additional data types and primitives
Shading Language management

Vulkan is defined to accept shaders in the SPIR-V IR
- In theory, how you generate it is up to you
- But, the ecosystem needs standards and stability
Shading Languages: it’s complicated...

Vulkan is defined to accept shaders in the SPIR-V IR
- In theory, how you generate it is up to you
- But, the ecosystem needs standards and stability

GLSL is NOT going away - will live forever
- But no plan to evolve its syntax (templates, meta-programming, etc)

Some developers will always prefer HLSL
- Microsoft has been very welcoming and accommodating - thanks!
- But, resourcing is a problem
Shading Languages

Vulkan is defined to accept shaders in the SPIR-V IR
- In theory, how you generate it is up to you
- But, the ecosystem needs standards and stability

GLSL is NOT going away - will live forever
- But no plan to evolve its syntax (templates, meta-programming, etc)

Some developers will always prefer HLSL
- Microsoft has been very welcoming and accommodating - thanks!

Slang is an option
- NVIDIA has offered to place it under community governance
- Khronos is one possible hosting consortium - under active discussion
Vulkan BOF Presentations

Vulkan: Looking back, and looking forward
- Tom Olson (Vulkan Working Group Chair / Arm)

Vulkan SDK: Where we started, and where we are going
- Karen Ghavam (LunarG)

Vulkan: Crash Diagnostic Layer
- Jeremy Gebben (LunarG)

--------------------------------------------- 10 minute BREAK ---------------------------------------------

Slang in Vulkan
- Hai Nguyen (NVIDIA)

EVOLVE - Next Generation Benchmarking
- Jasper Bekkers and Darius Bouma (Traverse Research)

Adding Vulkan to Pixar’s Hydra Storm Renderer
- Henrik Edstrom (Autodesk), Ashwin Bhat (Autodesk), and Caroline Lachanski (Pixar)
We need your help!

There’s a lot going on!
- Need help prioritizing
- Need help doing the work

You can help
- Talk to us
- Raise issues and share opinions on GitHub (KhronosGroup/Vulkan-Docs)
- Contribute to open-source repos

Consider joining Khronos and the Vulkan Working Group
- Not that expensive
- Associate memberships for small companies
- Academic memberships also available
What’s new in the Vulkan ecosystem
Vulkan BOF Presentations

Vulkan: Looking back, and looking forward
- Tom Olson (Vulkan Working Group Chair / Arm)

Vulkan SDK: Where we started, and where we are going
- Karen Ghavam (LunarG)

Vulkan: Crash Diagnostic Layer
- Jeremy Gebben (LunarG)

Slang in Vulkan
- Hai Nguyen (NVIDIA)

EVOLVE - Next Generation Benchmarking
- Jasper Bekkers and Darius Bouma (Traverse Research)

Adding Vulkan to Pixar’s Hydra Storm Renderer
- Henrik Edstrom (Autodesk), Ashwin Bhat (Autodesk), and Caroline Lachanski (Pixar)
Vulkan Video is a Thing!

- Vulkan Video expands Vulkan capabilities
  - Accelerated processing of streamed media into the Vulkan pipeline

Vulkan Video is increasingly providing cross-platform media framework acceleration

Status tracked at
https://blogs.igalia.com/vjaquez/vulkan-video-status/
Ray tracing is coming to mobile!
Games are a given
Professional rendering

Artwork by Emily Bisset, courtesy of Adobe

Vulkan Ray Tracing in Aurora:
An Open Source Real-Time Path Tracer

https://github.com/Autodesk/Aurora

Substance 3D Stager
Vulkan BOF Presentations

Vulkan: Looking back, and looking forward
- Tom Olson (Vulkan Working Group Chair / Arm)

Vulkan SDK: Where we started, and where we are going
- Karen Ghavam (LunarG)

Vulkan: Crash Diagnostic Layer
- Jeremy Gebben (LunarG)

Slang in Vulkan
- Hai Nguyen (NVIDIA)

EVOLVE - Next Generation Benchmarking
- Jasper Bekkers and Darius Bouma (Traverse Research)

Adding Vulkan to Pixar’s Hydra Storm Renderer
- Henrik Edstrom (Autodesk), Ashwin Bhat (Autodesk), and Caroline Lachanski (Pixar)
What’s New:

Documentation and Developer Support
Vulkan Documentation Project

Bring Vulkan documentation together in one place

- Specification, Vulkan Guide, Proposal documents, Samples...
- Easy navigation and cross-linking
- https://docs.vulkan.org
- Please report issues at https://github.com/KhronosGroup/Vulkan-Site
Vulkan Samples Repository

A home for Vulkan sample code
- Intended to help you learn to use Vulkan effectively
- GPU, OS, and platform neutral, well tested
- On github in open source (Apache 2.0)
- Access via docs.Vulkan.org or at github/KhronosGroup/Vulkan-Samples

A community effort
- Khronos member ISVs, IHVs, contractors
- Interested community members
Some recently added samples

Sparse Image / virtual texture (Mobica)

OIT using per-pixel linked lists (community)

Mobile NeRF (Qualcomm)

Vulkanised!

First full-scale Vulkanised was held in February 2023
  • Hosted by Google in Munich, Germany
  • Three days of talks, panels, demos, and a Vulkan course
    - All on line at https://vulkan.org/learn#videos

Second in February 2024
  • Hosted by Google in Sunnyvale, California
Vulkanised 2025

The 7th Vulkan Conference | Cambridge, UK | Feb 11-13, 2025
The Premier Vulkan Developer Conference

To be hosted by Arm in Cambridge, UK - submissions due Oct. 11
Final thoughts
Seriously, it’s been *ten years*?

Yes, it has

I’m retiring in October

New leadership coming!
Thanks!
Thanks!