Vulkan Update

Tom Olson, Arm
Vulkan Working Group Chair
Some thoughts about Vulkan in 2024
I’m so glad I don’t need this slide...

What is Vulkan?
A modern graphics and compute API for programming GPUs
- Focus on real-time / high performance applications, especially games
- Focus on cross-platform portability

Design principles
- Explicit control - no driver magic
- Trust the developer

Programmer responsibilities
- Dependency tracking
- Synchronization
- Memory and object lifetime management
- Optimization!
- Writing correct code!
Maybe Vulkan is growing up?

Budget of about $1.3M / yr for Khronos operations
  • Much more than that in member companies

About 40 people on our weekly development call
  • Ten elected officers, 5-6 meetings a week
  • Well-defined workflows for development and release

Tens of thousands of developers, and a complex ecosystem
  • A lot of people are depending on us to get it right
  • We take this very seriously. Challenge accepted.
  • We need your input
We have plenty of challenges

Fragmentation

• A huge problem for developers
• Drives much of our strategic thinking
• Profiles, Roadmap, etc

High level language ecosystem

• In theory, we don’t care...
• But in practice we have to
• Ecosystem split between GLSL and HLSL
• Glad to see lots of language talks here at Vulkanised!
Outline

Some thoughts about Vulkan in 2024

Vulkan profiles and the roadmap

What’s new?

• Vulkan Roadmap 2024
• New features in the API
• Adoption and applications
• Developer support
Vulkan Profiles (again)
Vulkan is caps-intensive

Implementations can differ in many, many ways
- Core version: 1.0, 1.1., 1.2, 1.3
- Extensions:
- Capabilities:
- Properties: how many render targets? How big can they be? ...
- Formats: What pixel formats can I use? Which can I sample? Which can I render to?
- And so on...
Result

Very hard to write portable code.
  • Unfortunately, it has to be this way...

The good news
  • Feature support isn’t completely random!
  • De facto standards exist for specific markets

Leads to the idea of profiles
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
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

External to the Vulkan specification
- Spec doesn’t know about them
- Vulkan drivers do not know what profiles they support
- You can write new profiles to describe old hardware.
Vulkan Profile Specification

JSON schema
- Machine-readable

Enables code generation
- Support queries
- Device creation
- Set operations
- ...

```json
"capabilities": {
  "baseline": {
    "extensions": {
      "VK_KHR_surface": 1,
      "VK_KHR_android_surface": 1,
      "VK_KHR_swapchain": 1,
      "VK_KHR_get_physical_device_properties2": 1,
      "VK_KHR_maintenance1": 1,
      ...
    },
    "features": {
      "VkPhysicalDeviceFeatures": {
        "depthBiasClamp": true,
        "FragmentStoresAndAtomics": true,
        "fullDrawIndexUint32": true,
        "imageCubeArray": true,
        "independentBlend": true,
        "robustBufferAccess": true,
        ...
      },
      "VkPhysicalDeviceMultiviewFeatures": {
        "multiview": true
      },
      ...
    },
    "properties": {
      "VkPhysicalDeviceProperties": {
        "limits": {
          "maxImageDimension1D": 4096,
          ...
        }
      }
    }
  }
}
```
Why Profiles are Awesome
Why Profiles are Awesome

It’s like having your own personal Vulkan spec
  • All your favorite extensions and features are supported
Why Profiles are Awesome

It’s like having your own personal Vulkan spec
• All your favorite extensions and features are supported

No driver update required!
• You can start using it today
Why Profiles are Awesome

It’s like having your own personal Vulkan spec
  • All your favorite extensions and features are supported

No driver update required!
  • You can start using it today

No code changes required!
  • (almost)
For example

Say you’re writing an Android game...

- Targeting 3-4 years of devices, 4 GPU vendors, 12 handset OEMS
- Capability management is going to be fun!

Suppose Google says

- “if you target this profile, you’ll reach 90% of devices”
- Seems like a win!
For example

Say you’re writing an Android game...
- Targeting 3-4 years of devices, 4 GPU vendors, 12 handset OEMS
- Capability management is going to be fun!

Suppose Google says
- “if you target this profile, you’ll reach 90% of devices”
- Seems like a win!

They already have...

Android Baseline 2021
Profile Support
January 2023

Baseline 2021 87%
None 13%
Vulkan Roadmaps

**Vulkan Roadmap Milestones**
Reduce API Fragmentation

Milestones define a set of Vulkan extensions and capabilities that developers can expect to be widely supported on mid- to high-end “immersive graphics” devices.

**Core Specification: Mainstream GPUs**
What’s new: API
Second milestone on the Vulkan Roadmap

- Expressed as a profile, but forward looking
- Captures expected feature set for “immersive graphics” 2024-2026+
Vulkan Roadmap 2024 requirements

Vulkan 1.3 with Vulkan Roadmap 2022 profile

Implementation minima
- maxBoundDescriptorSets >= 7
- maxColorAttachments >= 8

Previously optional features and limits
- multiDrawIndirect
- shaderDrawParameters
- shaderImageGatherExtended
- shaderInt8, shaderInt16, shaderFloat16
- storageBuffer8BitAccess, storageBuffer16BitAccess
- shaderRoundModeRTEFloat16/32
Roadmap 2024 required extensions

Older extensions
- VK_KHR_push_descriptor (2019)
- VK_KHR_subgroup_uniform_control_flow (2020)
- VK_KHR_map_memory2 (March 23)
- VK_KHR_maintenance5 (July 23)

Extensions newly promoted from EXT
- VK_KHR_index_type_uint8 (newly promoted from EXT)
- VK_KHR_line_rasterization (newly promoted from EXT)
- VK_KHR_load_store_op_none (newly promoted from EXT)
- VK_KHR_vertex_attribute_divisor (Dec 2023 promoted from EXT*)
New extensions in Roadmap 2024

**VK_KHR_shader_expect_assume**
- Compiler hints

**VK_KHR_shader_subgroup_rotate**
- A handy special case of general permutation

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

**VK_KHR_maximal_reconvergence / VK_KHR_shader_quad_control**

New extensions in Roadmap 2024

VK_KHR_dynamic_rendering_local_read
- Allows pipeline barriers within dynamic rendering
- Must include VK_DEPENDENCY_BY_REGION_BIT
- 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](https://www.khronos.org/blog/streamlining-subpasses)
## Other New Extensions

### 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

### Programming model improvements
- VK_EXT_attachment_feedback_loop_dynamic_state
- VK_EXT_host_image_copy

### Tile-based optimizations
- VK_EXT_shader_tile_image

### Window System Integration
- VK_EXT_surface_maintenance1
- VK_EXT_swapchain_maintenance1
- VK_EXT_pipeline_protected_access

### Exploratory / Experimental
- VK_EXT_shader_object
- VK_AMDX_shader_enqueue

### Maintenance
- VK_KHR_maintenance6
- VK_EXT_depth_bias_control
- VK_EXT_image_sliced_view_of_3D

### Ray Tracing
- VK_KHR_ray_tracing_position_fetch
Vulkan Video Extensions

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
Vulkan Video Extensions

- Vulkan Core
  - Vulkan Video Decode Core
  - Vulkan Video Encode Core
  - Video Encode QM (Quantization Map) extension
  - Video Encode AQ (Adaptive Quantization) extension
  - Vulkan Video Decode h.264
  - Vulkan Video Decode h.265
  - Vulkan Video Decode VP9
  - Video Encode advanced high-quality encoding: QM, AQ, etc.
  - Vulkan Video Encode AV1
  - In active development
  - In the development pipeline
Vulkan Video is a Thing!

Frameworks

- FFmpeg
- gstreamer

Status tracked at https://blogs.igalia.com/vjaquez/vulkan-video-status/

Several talks here at Vulkanised!
Vulkan Ray Tracing

VK_KHR_ray_tracing_position_fetch
- Query vertex positions after a ray hit
- Motivated by need for triangle normals in Lumen
- Substantial perf win!

```c
/* Without ray tracing position fetch */
uint triIndex0 = indexBuffer[firstIndex + gl_PrimitiveID*3 + 0];
uint triIndex1 = indexBuffer[firstIndex + gl_PrimitiveID*3 + 1];
uint triIndex2 = indexBuffer[firstIndex + gl_PrimitiveID*3 + 2];

vec3 vertPos0 = vertexBuffer[triIndex0];
vec3 vertPos1 = vertexBuffer[triIndex1];
vec3 vertPos2 = vertexBuffer[triIndex2];

payload.geoNormal = cross(vertPos1 - vertPos0, vertPos2 - vertPos0);

/* With ray tracing position fetch */
vec3 vertPos0 = gl_HitTriangleVertexPositionsEXT[0];
vec3 vertPos1 = gl_HitTriangleVertexPositionsEXT[1];
vec3 vertPos2 = gl_HitTriangleVertexPositionsEXT[2];

payload.geoNormal = cross(vertPos1 - vertPos0, vertPos2 - vertPos0);
```

- https://www.khronos.org/blog/introducing-vulkan-ray-tracing-position-fetch-extension
Ray tracing in mobile is a Thing too!

See Wednesday morning’s talk by Iago Calvo Lista!

What’s New:
Platforms
Vulkan on MacOS / iOS

The problem
- How do you run Vulkan apps without a driver?

MoltenVK
- Shim library for MacOS/iOS
- Maps Vulkan calls to native Metal API
- Shaders translated at create time
- Supports only an efficient subset

Current status
- Fully supported in LunarG SDK
- Vulkan 1.2 feature (sub)set (and growing)
- Will be Vulkan 1.0 conformant soon
Applications and Engines Using MoltenVK

- **Games shipping with MoltenVK**
  - DOTA 2
  - Metro Exodus
  - Final Fantasy XIV
  - Dark Souls: Remastered
  - Dark Souls III
  - DOTA Underlords
  - Aerofly Flight Simulator 2
  - Path of Exile
  - Raft
  - The Elder Scrolls Online
  - Celeste
  - Transport Fever 2
  - Shadow Warrior 2
  - Streets of Rage 4
  - Jupiter Hell
  - Wreckfest
  - Victoria 3
  - Artifact
  - GZDOOM
  - vkQuake/vkQuake2

- **Games runnable by users via Crossover and MoltenVK**
  - Halo: Combat Evolved
  - God of War (2018)
  - Grand Theft Auto V
  - Battlefront II
  - Elder Scrolls V Skyrim: SE
  - AGE Of Empires II: Definitive Edition
  - Witcher 3

- **Applications shipping with MoltenVK**
  - Autodesk Fusion 360
  - NAP
  - Autodesk Flame

- **Engines using MoltenVK**
  - Google Filament
  - Facebook IGL
  - LightweightVK
  - Defold
  - Clausewitz Engine (Paradox)
  - Ultra Engine
  - Diligent Engine
  - Ncnn
  - Godot

- **Platform emulators using MoltenVK**
  - VKD3D (Direct3D 12)
  - DXVK (Direct3D 9/10/11)
  - Google Android Emulator
  - Dolphin (Wii & GameCube)
  - Ryujinx (Switch)
  - Cemu (Wii U)
  - RPCS3 (PS3)
  - PCSX2 (PS2)
Vulkan Adoption on Android

In January 2023 Vulkan was available on 85% of active Android devices
What’s New:

Conformance Testing
Vulkan Conformance Test Suite (CTS)

~3.9 million tests as of May 2023

This work is licensed under a Creative Commons Attribution 4.0 International License
What’s New:
SDK and Tools
Hi Karen!
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

Resource Creation

Vulkan supports two primary resource types: buffers and images. Resources are views of memory with associated formatting and dimensionality. Buffers provide access to raw arrays of bytes, whereas images can be multidimensional and may have associated metadata.
New Vulkan Guide articles

HLSL in Vulkan

• Coming soon: HLSL / GLSL Mapping (Sascha Willems)
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

Welcome to Vulkanised 2024!