© Copyright Khronos™ Group 2018

GDC Update
March 2018

Neil Trevett | Khronos President
NVIDIA | VP Developer Ecosystem
ntrevett@nvidia.com | @neilt3d
www.khronos.org
Topics

• Vulkan 1.1 Launched!
  - Strong ecosystem progress - not just a specification

• Porting Vulkan apps to Apple platforms
  - Vulkan Portability Initiative and MoltenVK in open source

• OpenXR making good progress
  - Implementations in progress, architectural details released

• glTF Widespread Industry Adoption
  - Facebook, Unity, Google, Adobe and more!
Vulkan 1.1 Launch and Ongoing Momentum

**Strengthening the Ecosystem**
- Improved developer tools (SDK, validation/debug layers)
  - More rigorous conformance testing
- Shader toolchain improvements (size, speed, robustness)
- Shading language flexibility - HLSL and OpenCL C support
- Vulkan Public Ecosystem Forum

**Vulkan 1.0 Extensions**
- Maintenance updates plus additional functionality
  - Explicit Building Blocks for VR
  - Explicit Building Blocks for Homogeneous Multi-GPU
  - Enhanced Windows System Integration
  - Increased Shader Language Flexibility
  - Enhanced Cross-Process and Cross-API Sharing

**Building Vulkan’s Future**
- Deliver complete ecosystem - not just specs
- Listen and prioritize developer needs
- Drive GPU technology

**Widening Platform Support**
- Pervasive GPU vendor driver availability
- Port Vulkan apps to macOS/iOS and DX12
- Open source drivers

**Vulkan 1.1**
- Specification launching March 7th with open source conformance tests and tools, and multiple vendor implementations!
New Generation GPU APIs

Non-proprietary, royalty-free open standard ‘By the industry for the industry’
Portable across multiple platforms - desktop and mobile
Modern architecture | Low overhead | Multi-thread friendly
EXPLICIT GPU access for EFFICIENT, LOW-LATENCY, PREDICTABLE performance

Vulkan Porting Tools

Vulkan is the primary platform 3D API on Android 7.0+
Explicit GPU Access

• Application tells the driver what it is going to do
  - In enough detail that driver doesn’t have to guess
  - When the driver needs to know it

• In return, driver promises to do
  - What the application asks for
  - When it asks for it
  - Very quickly

• Predictable performance costs
  - Creating pipelines, allocating memory, ...

• No driver magic - no surprises - simpler drivers
  - Remove guesswork and late decision-making

• Putting control in the hands of developers
  - Flexible scheduling of CPU and GPU workloads
  - Management of memory and synchronization
Pervasive Vulkan 1.0

Major GPU Companies supporting Vulkan for Desktop and Mobile Platforms

Platforms

- Desktop
- Mobile (Android 7.0+)
- Media Players
- Consoles
- Virtual Reality
- Cloud Services
- Embedded

Game Engines

- VALVE
- id
- CRYENGINE
- unity
- EPIC GAMES
- Croteam
- Xenko
Vulkan is Powering Mobile Gaming...

- Galaxy on Fire 3
- Heroes of Incredible Tales
- Lineage 2 Revolution
- Dream League Soccer
... and Enabling Cross-Platform AAA Titles

AAA Titles on PC

Vulkan-only AAA Titles on PC

Publicly announced games as of March 2018
#Vulkan = 34
#DX12 = 27

Dota 2 on PC and macOS

AAA on macOS
Vulkan Ecosystem Momentum

LunarG Vulkan SDK
Download rate increases every year since launch
http://vulkan.lunarg.com

Vulkan GitHub Open Source Projects end of 2016

Today

1,798 repository results
Vulkan Layer Architecture

- Layered design for cross-vendor tools innovation and flexibility
  - IHVs plug into a common, extensible layer architecture for code validation, debugging and profiling during development without impacting production performance
- LunarG open-source Vulkan SDK ships on Windows, Linux and Mac
  - Validation, debug, and device simulation layers

Production Path (Performance)

Vulkan-based Title

Vulkan’s Common Loader

IHV’s Installable Client Driver

Debug Layers can be installed during Development

Validation Layers

Debug information via standardized API calls

RenderDoc
Frame capture and introspection
https://renderdoc.org/
New Vulkan Developer Tools

- **Vulkan Layer Factory (VLF)**
  - Rapid layer development through hiding implementation details

- **Device Simulation Layer**
  - Simulate target device capabilities, without requiring actual physical hardware

- **Assistant Layer**
  - Best practices layer that highlights potential performance issues, questionable usage patterns, common mistakes, and items not specifically prohibited by the Vulkan specification but that may lead to application problems

Delivered with the LunarG Vulkan SDK

[https://vulkan.lunarg.com/](https://vulkan.lunarg.com/)

Source available in the LunarG Vulkan Tools repository

[https://github.com/LunarG/VulkanTools](https://github.com/LunarG/VulkanTools)
CTS: Conformance Test Coverage

Khronos’s #1 engineering project

- Raise issues and PRs at https://github.com/KhronosGroup/VK-GL-CTS
Vulkan Ecosystem Complexity

Stuff the Vulkan Working Group does
- Specifications (API, SPIR-V, GLSL)
- Conformance tests
- Validation layers

Stuff the Working Group is involved in
- SDKs
- Compilers and SPIR-V tools

Stuff other people do
- Trace and debug tools (RenderDoc etc)
- Documentation, best practices, example code
- https://vulkan.gpuinfo.org/
- ....lots of other cool stuff

Tricky for developers and contributors to navigate...
Public Vulkan Ecosystem Forum

A new public forum to share ecosystem issues and opportunities - and coordinate solutions!

Vulkan-related open source projects

Vulkan Ecosystem Forum
(open to both Khronos members and non-members)

Vulkan Working Group and Advisory Panel

Vulkan Developer Community

https://github.com/KhronosGroup/Vulkan-Ecosystem
Bringing Vulkan 1.0 Apps to Apple Platforms

VALVE

Dota 2 running on Mac up to 50% faster than native OpenGL

Applications

Vulkan macOS SDK

Open source SDK to build, run, and debug applications on macOS including validation layer support

SPIRV-Cross
Convert SPIR-V shaders to platform source formats

macOS / iOS Run-time
Maps Vulkan to Metal

MoltenVK for macOS and iOS
For macOS 10.11, iOS 9.0 and up

Previously a paid product
Now released into OPEN SOURCE
Completely free to use - no fees or royalties - including for commercial applications
SPIR-V Ecosystem

Open source tools and translators
https://github.com/KhronosGroup/SPIRV-Tools

Third party kernel and shader languages

GLSL
HLSL

glslang
DXC

SPIR-V
- Khronos defined cross-API IR
- Native graphics and parallel compute
- Easily parsed/extended 32-bit stream
- Data object/control flow retained for effective code generation/translation

OpenCL C Front-end
OpenCL C++ Front-end
SYCL Front-end

SPIRV-Cross

SPIR-V (Dis)Assembler

SPIR-V Validator

SPIRV-opt | SPIRV-remap

SPIRV Optimizations
- Inlining (exhaustive)
- Store/Load Elimination
- Dead Code Elimination
- Dead Branch Elimination
- Common Uniform Elimination
- Loop Unrolling and Constant Folding
- Common Subexpression Elimination

Additional Intermediate Forms

LLVM to SPIR-V Bi-directional Translator

Khrnos liaising with Clang/LLVM Community
E.g. discussing SPIR-V as supported Clang target

SPIR-V 1.3 released with Vulkan 1.1
Clspv OpenCL C to Vulkan Compiler

- Experimental collaboration between Google, Codeplay, and Adobe
  - Successfully tested on over 200K lines of Adobe OpenCL C production code
  - Released in open source [https://github.com/google/clspv](https://github.com/google/clspv)
  - Tracks top-of-tree LLVM and clang, not a fork

- Compiles OpenCL C’s programming model to Vulkan’s SPIR-V execution environment
  - Proof-of-concept that OpenCL compute can be brought seamlessly to Vulkan
New Functionality in Vulkan 1.1

• Protected Content
  - Restrict access or copying from resources used for rendering and display
  - Secure playback and display of protected multimedia content

• Subgroup Operations
  - Efficient mechanisms that enable parallel shader invocations to communicate
  - Wide variety of parallel computation models supported

Example Subgroup Operations

A subgroup is a set of invocations (tasks) running on a GPU Compute Unit
(Note many GPUs typically support subgroup sizes of 32/64 invocations)
Proven Extensions Now in Vulkan 1.1 Core

- **Multiview**
  - A single render pass can render to multiple image views simultaneously
  - Use cases include rendering left and right eye views to a stereo VR headset, or six face views of a cube map, with a single draw call

- **Device Groups**
  - Enables homogeneous multi-GPU systems such as AMD CrossFireX and NVIDIA SLI for high-performance gaming and VR
  - Device groups make the number of GPUs in the system relatively transparent to the application
  - Applications can be written to use one or many GPUs with a minimum of changes

- **Cross-process and Cross-API sharing**
  - Share memory and sync primitives (semaphores and fences) between APIs in a single application, or between multiple applications
  - Many applications, e.g. allowing a compositor to present images from Vulkan and OpenGL ES applications to the same display device
  - This feature is used in the Valve Steam VR SDK and other advanced mobile platforms

- **Advanced Compute Functionality**
  - Read and write 16-bit quantities stored in GPU memory, and to refer to data structures using a restricted form of pointers
  - Greatly expands Vulkan’s ability to support GPU compute kernels

- **HLSL support**
  - Relaxed block layout enables support for the same memory data layout constraints as Microsoft’s HLSL
  - Enables identical HLSL shaders in both Vulkan and DX applications
  - Easier translation of HLSL into SPIR-V, the portable compiled shader format accepted by Vulkan

- **YCbCr support**
  - Sample the YCbCr color formatted textures produced by many video codecs
  - Useful for compositing video streams and mixing them with other graphical content
Vulkan 1.1 Today

- **Specification**: in open source for community use and feedback
- **Conformance Tests**: in open source for responsive bug fixing and enhancements
- **Tools**: LunarG SDK and validation/debug/simulation/assistant layers - all in open source

- Conformant Vulkan 1.1 drivers and devices:

  - AMD
  - ARM
  - Imagination
  - Intel
  - NVIDIA
  - Qualcomm
  - AMD RADV
  - Intel ANV
  - Nintendo Switch
  - Android

Vulkan 1.1 in Android P developer preview

Khronos' Ongoing Vulkan Mission
Continue to build the complete Vulkan Ecosystem
Specifications, tests, tools and community
Listen and prioritize developer needs
Drive GPU technology in the industry
Pervasive Vulkan 1.0

Major GPU Companies supporting Vulkan for Desktop and Mobile Platforms

Platforms

- Desktop
- Mobile (Android 7.0+)
- Media Players
- Cloud Services
- Consoles
- Virtual Reality
- Embedded

Game Engines

- Valve
- id
- CRYENGINE
- unity
- EPIC GAMES
- Croteam Serious Engine
- XENKO
Market Demand for Universal Portability

Community Outreach at GDC 2017
Create a hybrid portability API?

Feedback - AVOID CREATING A FOURTH API!!!
Would need new specification, CTS, Documentation.
Additional developer learning curve.
A whole new specification to name, brand, promote.
Would INCREASE industry fragmentation
Bringing Vulkan Applications to Apple Platforms

Dota 2 running on Mac up to 50% faster than native OpenGL

Open source SDK to build, run, and debug applications on macOS including validation layer support

Previously a paid product
Now released into OPEN SOURCE
Completely free to use - no fees or royalties - including for commercial applications
Valve - Vulkan Dota 2 on macOS

Vulkan delivering up to 50% performance increase over native OpenGL
Draft Vulkan Portable Subset Definitions

- Metal and DX12 can support almost all of Vulkan - very little needs to be removed
  - Triangle fans are not supported
  - Separate stencil reference masks are not supported
  - Vulkan Event functionality is currently not supported
  - Support for only a limited set of texture-specific swizzles
  - Allocation callbacks in object creation functions will be ignored

Not all subset features are removed on all platforms. Supported features will typically be increased over time as underlying platform capabilities increase
Vulkan Portability Initiative Roadmap

Widened Platform Support
Diverse open source run-times over additional backends
E.g. Mozilla helping to drive gfx-rs for Vulkan over DX12, OpenGL and Metal
https://github.com/gfx-rs/gfx
https://github.com/gfx-rs/portability

Enhanced SDK Layers and Tools
E.g. Simulation layer to flag use of features not present in a selected target system

Coordinated Subset Management
Cross-platform standardized subsetted functionality queries
Conformance tests tailored per target - what is present must work!

TODAY
Free, open source tools, SDKs and libraries to bring Vulkan 1.0 applications to Apple using Metal
Future: Vulkan over D3D12 and more

- Motivation for building a Vulkan over D3D12 implementation:
  - Windows 10 S and Polaris are limited to Universal Windows Platform applications (UWP), which exclude Vulkan and OpenGL native drivers
  - Potential availability on XBox One
  - Even where Vulkan can be supported, it’s often not bundled with the system … making it harder for Firefox Quantum to run fast
- Gfx-rs is developing D3D12, Metal, and OpenGL backends for Vulkan Portability

Khronos welcomes collaboration with any open source library developers that wish to implement the Vulkan portable subset
Call to Action

- The Vulkan Portability Landing Page is here: [https://khr.io/vulkanpilandingpage](https://khr.io/vulkanpilandingpage)

- Download the free open source MoltenVK library [https://moltengl.com/moltenvk/](https://moltengl.com/moltenvk/)

- The free macOS SDK can be downloaded from LunarXchange. You can submit issues and feedback at the same website

- Let us know if you find functionality or performance issues! We appreciate feedback as we refine and improve this portability solution [https://khr.io/vulkanpifeedback](https://khr.io/vulkanpifeedback)
Companies Publicly Supporting OpenXR

OpenXR is a collaborative design for cross-platform AR/VR portability
Taking many lessons from proprietary ‘first-generation’ API designs
The Problem
The Solution

- **Current Device State:**
  - Normalized Predicted Poses
  - Controller / Peripheral State
  - Input Events

- **OpenXR Application Interface**

- **OpenXR Device Plugin Extension** (Optional)

- **VR & AR Vendor Runtime System**
  - Distortion Correction and Display Output
  - Coordinate System Unification & Prediction

- **Outgoing Requests:**
  - Pre-distortion image to display
  - Haptics

- **Outgoing Requests:**
  - Post-distortion image to display
  - Haptics

- **Device Vendor-Supplied Device Drivers**

- **Portable VR & AR Devices**
OpenXR Philosophies

1. Enable both VR and AR applications
   Unify common VR and AR functionality to streamline software and hardware development for a wide variety of products and platforms

2. Be future-proof
   While OpenXR 1.0 is focused on enabling the current state-of-the-art, the standard is built around a flexible architecture and extensibility to support rapid innovation in the software and hardware spaces for years to come

3. Do not try to predict the future of XR technology
   While trying to predict the future details of XR would be foolhardy, OpenXR uses forward-looking API design techniques to enable designers to easily harness new and emerging technologies

4. Unify performance-critical concepts in XR application development
   Developers can optimize to a single, predictable, universal target rather than add application complexity to handle a variety of target platforms
OpenXR Deployment Options

A lot more details here:
Input and Haptics

- Input uses abstracted Input Actions
  - E.g. “Move,” “Jump,” “Teleport”
- Many advantages
  - Existing content can easily use new devices
  - Mix-and-match multiple input sources to create a unified UI
  - Easy optional feature support (e.g. body tracking)
  - Future-proofing for innovation in input devices and form factors
OpenXR Viewport Configurations

- Applications can:
  - Query for runtime supported Viewport Configurations
  - Applications can then set the Viewport Configurations that they plan to use
  - Select and change their active configuration over the lifetime of the session

<table>
<thead>
<tr>
<th>Camera Passthrough AR</th>
<th>Stereoscopic VR / AR</th>
<th>Projection CAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>/viewport_configuration/ar_minitrack/monoscopic_window</td>
<td>/viewport_configuration/vr/hmd</td>
<td>/viewport_configuration/vr_cube/cave_vr</td>
</tr>
</tbody>
</table>

Photo Credit: Dave Pape
OpenXR Frame Timing

- **xrBeginFrame**
  - Ready to begin rendering pixels to the active image in our swap chain

- **xrEndFrame**
  - Finished rendering, and ready to hand off to the compositor for presentation
  - Takes a predicted display time, and layers to present

- **xrWaitFrame**
  - Called before we begin simulation of the next frame. This is responsible for throttling
OpenXR Compositor

- Compositor is responsible re-projecting, distorting, and displaying generated images to the device
  - Can have multiple Layers, up to the limit of the platform runtime
- Have XrCompositionData:
  - Type, display time, eye, and XrSpace
- Have XrCompositionLayerData:
  - Swap chain, and current index
OpenXR Extensibility

Core Standard
Core concepts that are fundamental to the specification for all use cases
Examples: Instance management, tracking, frame timing

KHR Extensions
Functionality that a large class of runtimes will likely implement
Examples: Platform support, Graphic API Extensions, Device Plugin, Headless, Tracking Bounds

EXT Extensions
Functionality that a few runtimes might implement
Examples: Performance Settings, Thermals, Debug Utils

Vendor Extensions
Functionality that is limited to a specific vendor
Examples: Device specific functionality
A Brief History of the Standard

Call for Participation / Exploratory Group Formation
Fall F2F, October 2016: Korea

Statement of Work / Working Group Formation
Winter F2F, January 2017: Vancouver

Specification Work
Spring F2F, April 2017: Amsterdam

Specification Work
Interim F2F, July 2017: Washington

Defining the MVP
Fall F2F, September 2017: Chicago

Resolving Implementation Blockers
Winterim F2F, November 2017: Washington

Raising Implementation Issues
Winter F2F, January 2018: Taipei

Present Day
First Public Information!
GDC, March 2018: Right Here, Right Now!

Coming Soon
Provisional Release
Conformance Testing and Implementation
Ratification and Release
glTF Highlights at GDC

- Facebook now supports glTF - drag and drop models to your feed
  - Driving exporter/tool demand: FBX2glTF, Modo, ...

- Sketchfab has over 150K glTF models
  - Under Creative Commons license

- Adobe Dimension is adopting glTF
  - glTF publishing service for interactive marketing materials

- Unity helping open source glTF importer/exporter
  - Plans to have a Unity package before the end of year

- Unreal 4.19 has experimental glTF import

- Focus on ecosystem robustness
  - gltf-vscode & glTF-Validator
  - glTF-Asset-Generator
  - glTF-Sample-Models

- Vibrant open-source community

- Not just for the web
glTF - Cross-Platform 3D Asset Transmission

Audio
- MP3

Video
- H.264

Images
- JPEG

3D
- glTF

- Compact to Transmit
- Fast to Load
- Describes Full Scenes
- Runtime Neutral
- Open and Extensible

OpenGL Transmission Format
Efficient transmission of 3D scenes and assets

glTF 1.0 - Primarily for WebGL
Uses GLSL for materials
Released December 2015

All glTF spec development on open GitHub:
https://github.com/KhronosGroup/glTF

glTF 2.0 - Physically Based Rendering!
Metallic-Roughness and Specular-Glossiness Materials
Rendering API independence
Released June 2017
glTF Industry Support

Publicly Stated Support for glTF
glTF Momentum

Creation Tools
- blender
- MINECRAFT
- Pixyz Software
- SketchUp
- Sony 3D Creator
- Substance Painter
- Oculus
- medium
- Paint 3D
- Instant3Dhub
- SIMPLYgon
- Assimp
- Open Asset Import Library
- glTF-validator
- FBX2glTF
- Collada2glTF
- glTF-asset-generator

3D Content
- Sketchfab
- Remix 3D
- Poly
- poly.google.com

Apps and Engines
- Unreal Engine
- Unity
- Godot Game engine
- PLAYCanvas
- three.js
- babylon.js
- A-Frame
- React VR
- Cesium
- Autodesk Forge
- UX3D Engine
- Xeogl
- Office
- Windows Mixed Reality Home

150k glTF Models in Creative Commons!
Google Draco Geometry Compression Extension

- **Draco glTF Extension** released for geometry compression in February 2018
  - Draco designed and built for compression efficiency and speed - great fit with glTF!

- **Up to 12X Model Compression**
  - With no loss of visual fidelity

- **Google Draco encoders and decoders in open source**
  - C++ source code encoders | C++ and JavaScript decoders
  - [https://github.com/google/draco](https://github.com/google/draco)

- **Draco compression already in use**
  - glTF pipeline, FBX2glTF, AMD Compressorator, three.js, and glTF sample models

![Comparing Mesh File Compression](image)
glTF Roadmap Discussions

- Careful balance: building ecosystem vs. moving the spec forward
  - Draco filled a key missing component for mesh compression - what is next?

- Texture Transmission Extension
  - Optimized transmission format with efficient local expansion to any GPU format

- Submitted to Ratification
  - Unlit (#1163)

- Last call for feedback
  - Lights (#1223)
  - Texture Transforms (#1015)
Industry Calls to Action

- Adopt glTF in your runtime and content pipeline - don’t be left out!
  - But - keep the ecosystem robust: use validation tools and provide spec feedback

- Join the spec and extension discussions on GitHub!
  - https://github.com/KhronosGroup/glTF/issues/456

- Contribute to the open-source ecosystem and let us help you spread the word!
  - Tutorials, sample models, answering questions, etc.

- Share your roadmap priorities with us!
  - https://github.com/KhronosGroup/glTF/issues/1051

- Join Khronos!
  - Get directly involved in the glTF Working Group