Vulkan Ray Tracing
Developer Rollout
December 2020
Vulkan Ray Tracing

The industry’s first open, cross-vendor, cross-platform standard for ray tracing acceleration

- Extension specifications released November 2020
- Coherent ray tracing framework with flexible merging of rasterization and ray tracing
- Set of extensions to Vulkan, GLSL and SPIR-V seamlessly integrates ray tracing into Vulkan 1.X
- Familiar to users of existing proprietary ray tracing APIs but also introduces new implementation flexibility
- Hardware agnostic - can be accelerated on existing GPU compute and dedicated ray tracing cores
- Primary focus on meeting desktop market demand for both real-time and offline rendering today - but designed to encourage mobile ray tracing too
The LunarG Vulkan SDK now integrates all the components necessary for developers to easily use the new ray tracing extensions, such as new shader tool chains, without needing them to be built from multiple repositories, and supports ray tracing validation within the SDK validation layers.

https://vulkan.lunarg.com/

Rollout of Vulkan SDK, Development Tools, Production Drivers and Education Materials

Vulkan Ray Tracing is ready for developers to use in their applications!
Developer Vulkan Ray Tracing Resources

Production Vulkan Drivers with Vulkan Ray Tracing are shipping

AMD Radeon Adrenalin 20.11.3 drivers for Radeon RX 6000 Series

NVIDIA R460 drivers for All RTX GPUs
GeForce GTX 1660 with 6GB+ of memory
GeForce GTX 1060+ with 6GB+ of memory

Intel Xe-HPG GPUs, available in 2021

Vulkan Ray Tracing Samples
Vulkan Ray Tracing Guide
How to use the Vulkan Ray Tracing extensions
Exploring deeper technical details of the Vulkan Ray Tracing specifications
Best practices for blending Vulkan rasterization and ray tracing techniques

Khronos Member Materials
Deep dive Vulkan Ray Tracing Tutorial
How to use Vulkan Ray Tracing to Create a complete mini-path tracer

New textbook, and here, on principles and history of ray tracing by Jon Peddie
A Vulkan-based glTF ray tracing viewer with open source on GitHub

Khronos welcomes developer feedback on Vulkan GitHub issues tracker
Quake II RTX

Quake II RTX 1.4 uses Vulkan Ray Tracing extensions
The world’s first cross-vendor ray tracing Vulkan application
Source code available on GitHub. Get it on Steam.
Holochip Light Field Display Rendering

Uses Vulkan Ray Tracing to optimize light field rendering at the same rendering cost as producing a single view for that same display.

Blog: how Vulkan Ray Tracing has enabled Holochip to innovate on efficient rendering techniques for light field displays.
## Vulkan Ray Tracing and DXR

<table>
<thead>
<tr>
<th>Feature</th>
<th>Vulkan Ray Tracing</th>
<th>DX12 / DXR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray Tracing Pipelines</td>
<td>At least one must be available</td>
<td>Yes</td>
</tr>
<tr>
<td>Ray Queries</td>
<td>DxR Tier 1.1</td>
<td></td>
</tr>
<tr>
<td>Language for Ray Tracing Shaders</td>
<td>GLSL or HLSL</td>
<td>HLSL</td>
</tr>
<tr>
<td>Pipeline Libraries</td>
<td>Yes</td>
<td>DxR Tier 1.1</td>
</tr>
<tr>
<td>Build Acceleration Structure on Host</td>
<td>Optional</td>
<td>AddToStateObject()</td>
</tr>
<tr>
<td>Deferred Host Operations</td>
<td>Optional</td>
<td>No</td>
</tr>
<tr>
<td>Capture/Replay Support for Tools (e.g., RenderDoc)</td>
<td>Optional</td>
<td>No</td>
</tr>
</tbody>
</table>

**Straightforward to port code between Vulkan Ray Tracing and DXR**

*Including re-use of ray tracing shaders written in HLSL*
Vulkan Ray Tracing includes GLSL and SPIR-V Extensions
Enabling compiled GLSL/SPIR-V shaders to operate in a Ray Tracing Pipeline - similar to HLSL features used in Direct3D’s DXR

HLSL and Vulkan with DXC
Microsoft’s DXC HLSL compiler was open sourced in Jan 2017
Google and others have added SPIR-V code generation to DXC with Microsoft’s knowledge and approval
Vulkan developers can now choose between GLSL and HLSL!

HLSL for Vulkan Ray Tracing
NVIDIA added code generation to DXC to generate SPIR-V for the Vulkan Ray Tracing extension from HLSL

Developers can port HLSL shaders with minimal changes between Vulkan Ray Tracing and DXR
## API Layering

<table>
<thead>
<tr>
<th>Layers Over</th>
<th>Vulkan</th>
<th>OpenGL</th>
<th>OpenCL</th>
<th>OpenGL ES</th>
<th>DX12</th>
<th>DX9-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulkan</td>
<td></td>
<td>Zink</td>
<td>clspv</td>
<td>GLOVE</td>
<td>vkd3d-Proton</td>
<td>DXVK</td>
</tr>
<tr>
<td>OpenGL</td>
<td>gfx-rs</td>
<td></td>
<td></td>
<td>Angle</td>
<td></td>
<td>WineD3D</td>
</tr>
<tr>
<td>DX12</td>
<td>gfx-rs</td>
<td>Microsoft 'GLOn12'</td>
<td>Microsoft 'CLOn12'</td>
<td></td>
<td>Microsoft D3D11On12</td>
<td></td>
</tr>
<tr>
<td>DX9-11</td>
<td>gfx-rs</td>
<td>Ashes</td>
<td></td>
<td>Angle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metal</td>
<td>MoltenVK</td>
<td>gfx-rs</td>
<td>clspv + SPIRV-Cross?</td>
<td>MoltenGL</td>
<td>Angle</td>
<td></td>
</tr>
</tbody>
</table>

Vulkan Ray Tracing designed to efficiently support layered DirectX 12 DXR

Wine 6.0 will support Vulkan specification version 1.2.162 which includes Vulkan Ray Tracing

E.g. Vkd3d-Proton used to port DX12 titles to Linux with Valve Proton

Vulkan is an effective porting target for multiple APIs e.g., for bringing DX 12 games to Linux.
Technical Background

December 2020
Step 1: Create Efficient Scene Geometry

Ray tracing may use a huge numbers of rays
Specialized data structures for interrogating scene geometry are necessary for efficient acceleration

Acceleration Structures
Contains low-level 3D geometry to be ray traced and high-level references into the geometry
Opaque internal organization details
Each vendor can optimize for processing for their hardware
E.g., Bounding Volume Hierarchy (BVH) for rapidly determining if there is any geometry in the path of a ray

Build Acceleration Structure
Vulkan driver integrates supplied geometry into its two-level Acceleration Structure

Using a BVH data structure to enable efficient ray tracing through a 3D scene
Step 2: Traverse Scene with Rays

Two ways to launch rays into the scene

Ray Tracing Pipelines
A new type of graphics pipeline
Implicit management of ray intersections
Application compiles a set of shaders into the pipeline to provide desired ray and material processing

Ray Queries
Any type of shader can launch a ray at any time
Shader can process intersection data however it wishes
Shader controls how traversal proceeds

Vulkan Ray Tracing Pipeline
Ray Tracing Pipelines can use many shaders
Potentially orders of magnitude more shaders (1000s) than traditional applications to handle diverse tracing techniques and material types

Compilation Bottleneck
Compiling many shaders into a Ray Tracing Pipeline can be computationally intensive and cause application bottlenecks and stuttering

Vulkan Pipeline Library Extension
Enables a library of SPIR-V shaders to be incrementally compiled into an existing Ray Tracing Pipeline saving significant processing load

Multiple shaders used to build complex lighting in a Quake 2 scene
Host Offload of Setup Operations

Ray tracing setup compute workloads can be intensive
Building Acceleration Structures and compiling Ray Tracing Pipelines
Two Vulkan mechanisms to offload and control setup workloads
on the host CPU(s) for smoother, faster rendering

Build Acceleration Structure on Host
Use the host to build Acceleration Structure in host memory and then
copy to the GPU - rather than build directly on the GPU
Final size of Application Structure is known before copying to the
GPU - enabling optimized GPU memory allocation

Deferred Host Operations
Driver returns deferred work handle to application for later execution
Application controls work execution and can chose to distribute onto
multiple cores and background threads

Deferred Host Operations can be used to
asynchronously use multiple CPU cores to build
Acceleration Structures on the host

Using Deferred Host Operations to build a
complex Acceleration Structure using
multiple CPU cores to offload the work from
the GPU for faster, smoother framerates
Vulkan Background

December 2020
Khronos Connects Software to Silicon

Open interoperability standards to enable software to effectively harness the power of multiprocessors and accelerator silicon

3D graphics, XR, parallel programming, vision acceleration and machine learning

Non-profit, member-driven standards-defining industry consortium

Open to any interested company

All Khronos standards are royalty-free

Well-defined IP Framework protects participant’s intellectual property

Founded in 2000
>150 Members ~ 40% US, 30% Europe, 30% Asia
Khronos Active Initiatives

3D Graphics
Desktop, Mobile, Web Embedded and Safety Critical

3D Assets
Authoring and Delivery

Portable XR
Augmented and Virtual Reality

Parallel Computation
Vision, Inferencing, Machine Learning

Guidelines for creating APIs to streamline system safety certification
Pervasive Vulkan

Desktop and Mobile GPUs

Platforms

- Desktop
- Android (Android 7.0+)
  (Vulkan 1.1 required on Android Q)
- Apple (via porting layers)
- Media Players
- Consoles
- Cloud Services
- AR and VR
- Game Streaming
- Embedded

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

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

Vulkan 1.1 Extensions
- Maintenance updates plus additional functionality
  - Timeline semaphores
  - DX/HLSL compatibility
  - Bindless resources
  - Reduced precision arithmetic
  - Formal memory model
  - Buffer references
  - SPIR-V 1.5

Roadmap Discussions
- Ray Tracing
- Variable Rate Shading
- Accelerated Video Encode/Decode
- Machine Learning Primitives
- Mesh Shaders

January 2020
Thank you!
Any questions?