KHRONOS® GROUP

Vulkan Ray Tracing Launch

17th March 2020

© Khronos® Group 2020

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

Vulkan Ray Tracing is Here!

Set of provisional extension specifications are publicly available today for industry feedback

First beta developer drivers are shipping today

Coherent ray tracing framework seamlessly integrated into existing Vulkan functionality - flexible merging of rasterization and ray tracing

Familiar to users of existing proprietary ray tracing APIs but also introduces new implementation flexibility

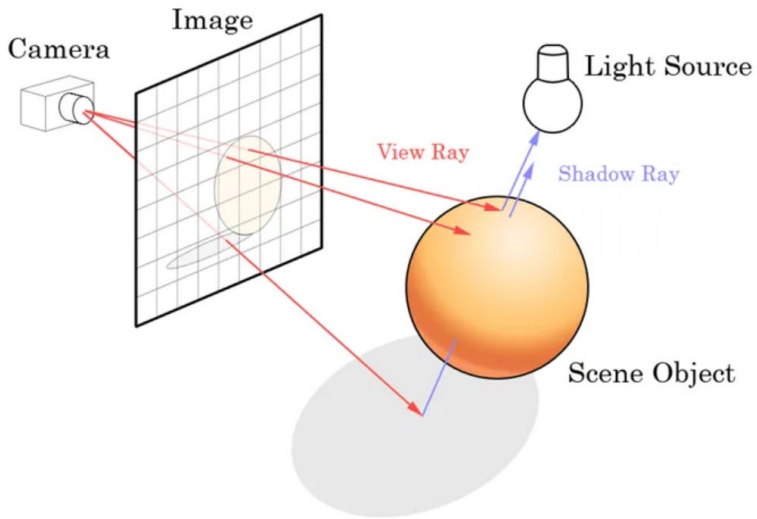
Hardware agnostic - can be accelerated on existing GPU compute and dedicated ray tracing cores

Primary focus on meeting desktop market demand for both real-time and offline rendering

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

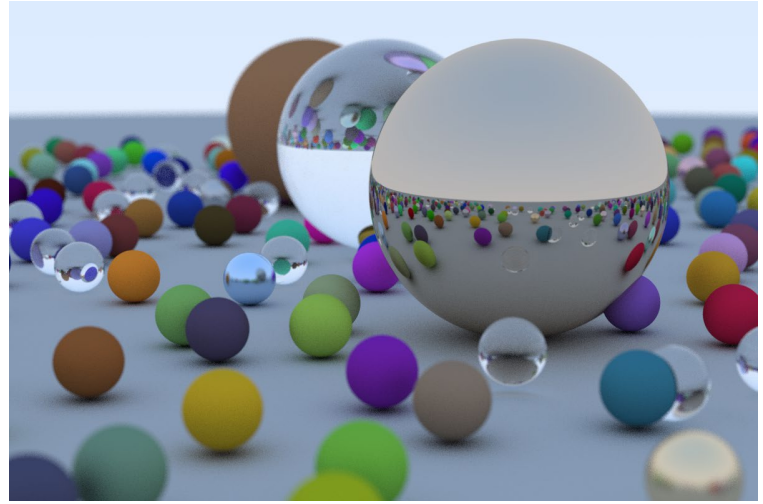
This work is licensed under a Creative Commons Attribution 4.0 International License

Ray Tracing Refresher



Ray tracing calculates how rays intersect and interact with scene geometry, materials and light sources

Rasterization and ray tracing can both use triangles to describe scene geometry - but only ray tracing calculates physical phenomena such as shadows, reflections and refractions



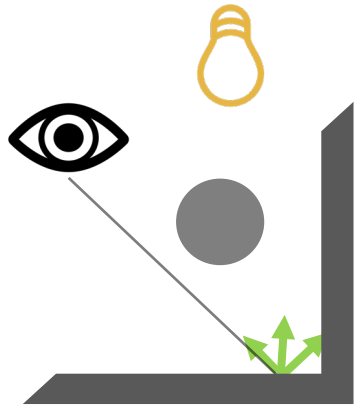
http://psgraphics.blogspot.com/2016/01/new-mini-book-ray-tracing-in-one-weekend.html

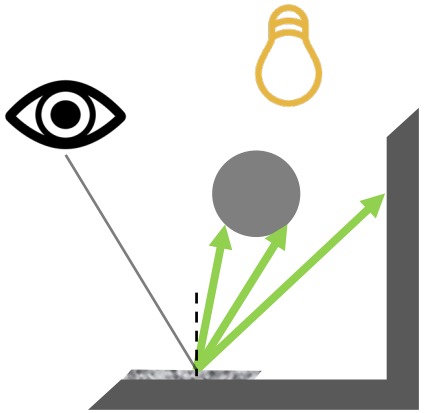
https://www.online-tech-tips.com/computer-tips/what-is-path-tracing-and-ray-tracing-and-why-do-they-improve-graphics/

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

Ray Tracing is a Flexible Technique

Ambient Occlusion

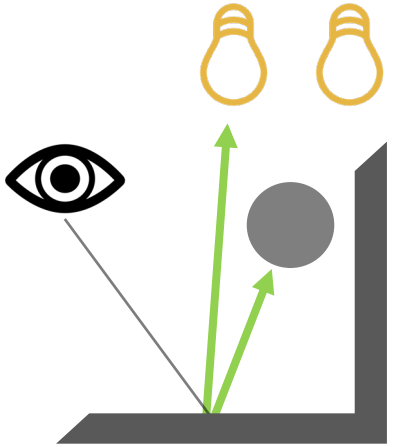


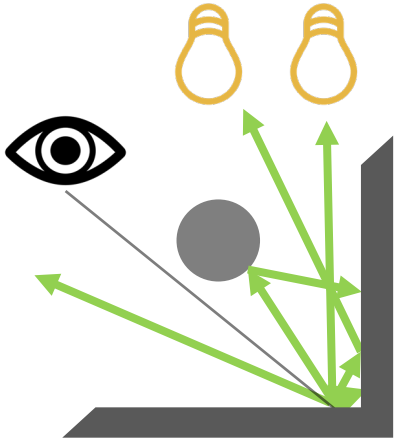


Reflections

Programmers need programmable flexibility to trace rays through scenes for a wide variety of visual effects - some examples…

Shadows





Global Illumination

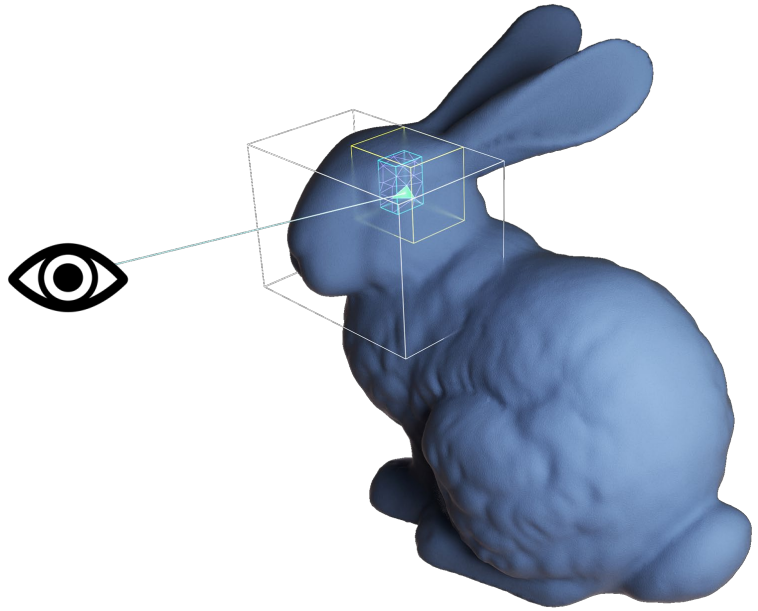
This work is licensed under a Creative Commons Attribution 4.0 International License

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

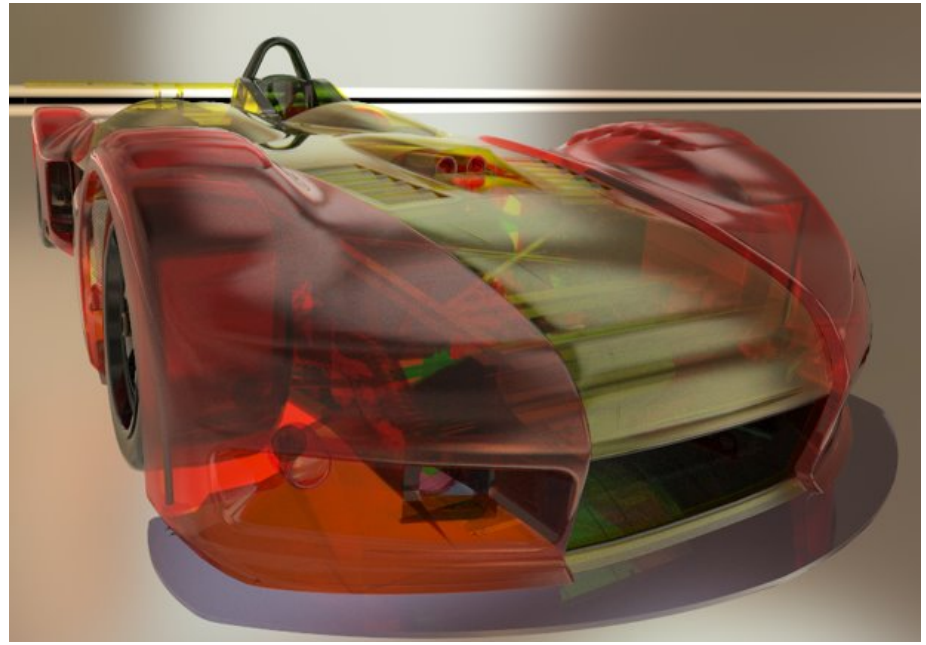
This work is licensed under a Creative Commons Attribution 4.0 International License

Step 2: Traverse Scene with Rays

Two ways to traverse Acceleration Structure
Launching rays into scene to generate results

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



Model courtesy of PTC

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

Traversal with Ray Tracing Pipelines

Implicit Ray and Shader Execution Management

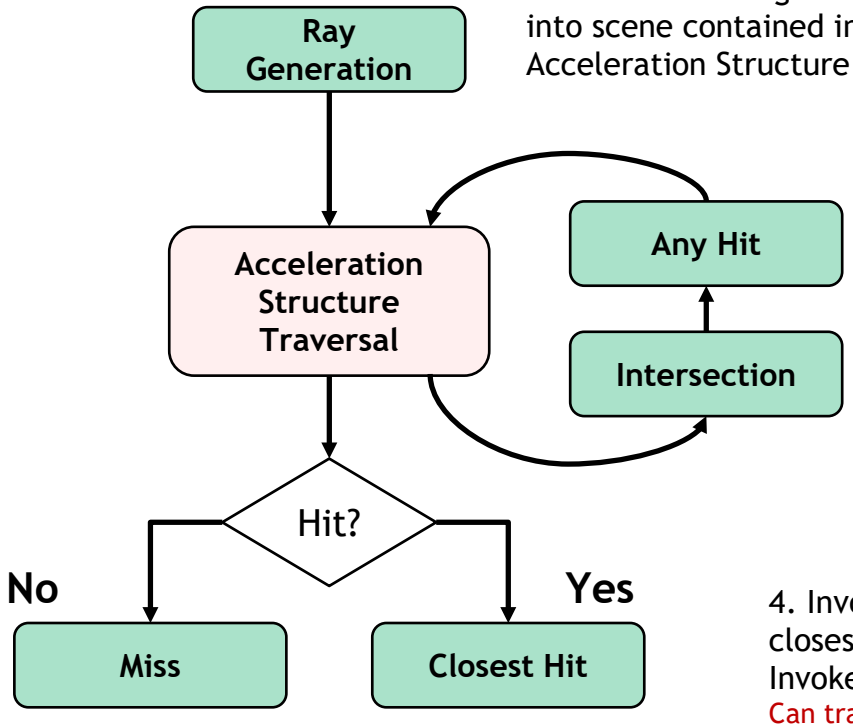
Application compiles collection of shaders to be invoked on ray/geometry intersection into the Ray Tracing Pipeline

Controlling which shaders are invoked during traversal enables a wide variety of ray tracing techniques

Hit Shaders can query the materials they intersect e.g. transparent materials can be handled differently than opaque

Intersection and Hit shaders can control how traversal proceeds

Shader stages can communicate parameters and results through ray payload structures



1. Launch 2D/3D grid of rays into scene contained in an Acceleration Structure

3. Invoke 'Any Hit' Shader if intersection is found.
Multiple intersections possible - arbitrary order

2. 'Intersection' Shader computes ray intersections
Ray-triangle intersections are built-in

4. Invoke 'Closest Hit' shader on the closest intersection of the ray OR Invoke 'Miss' Shader if no hit is found
Can trace more rays

Ray Tracing Pipeline

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

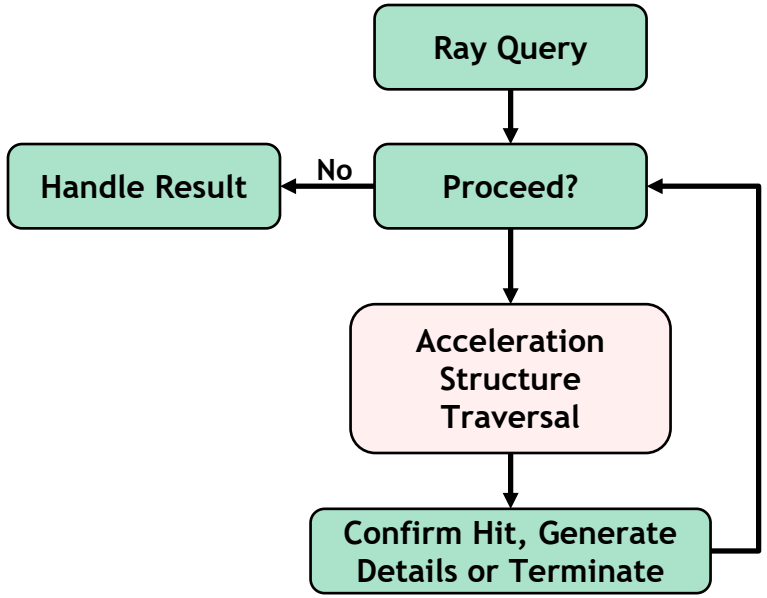
Traversal with Ray Queries

Explicit Ray Management within a Single Shader

Any standard shader (e.g. compute, vertex and fragment shaders) can invoke a single ray traversal at any time

Uses an Acceleration Structure and a geometric description of the ray being traced

Shader reads intersection properties during traversal and controls how materials are processed and how the traversal proceeds



1. Shader launches a single ray into scene contained in an Acceleration Structure

2. Shader takes action depending on intersection properties
Can trace more rays

Graphics, Compute or Ray Tracing Pipeline

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

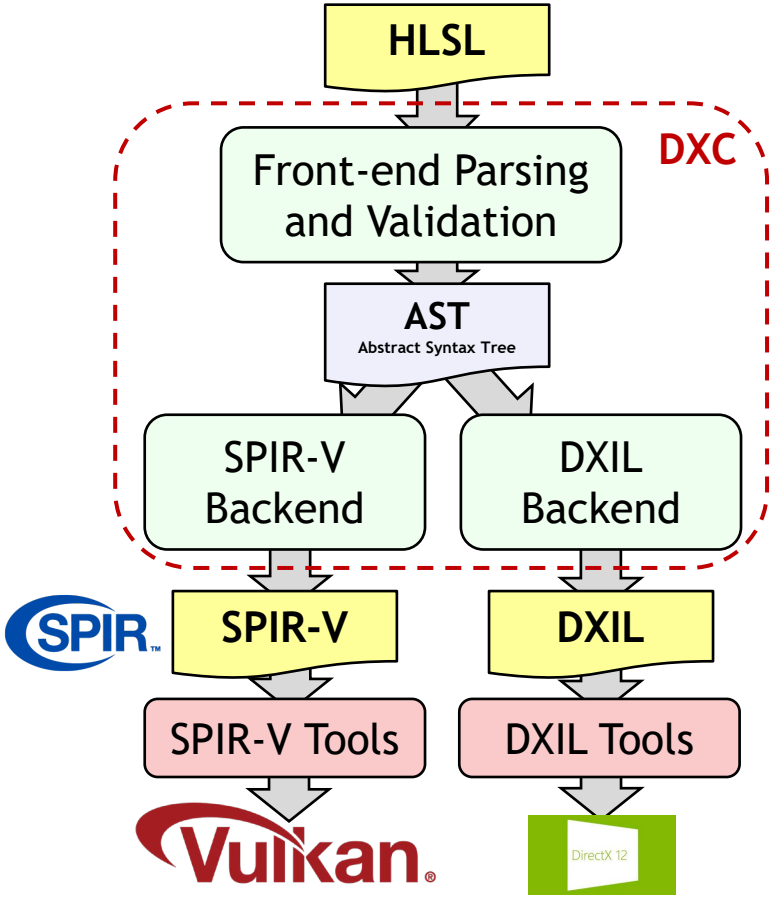
Vulkan Ray Tracing and Shading Languages

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
NVIDIA vendor ray tracing extensions from HLSL
Vulkan Ray Tracing Extensions supported in HLSL soon

Developers that wish to run their ray tracing applications on both Vulkan Ray Tracing and DXR will be able to port their HLSL shaders with minimal changes



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

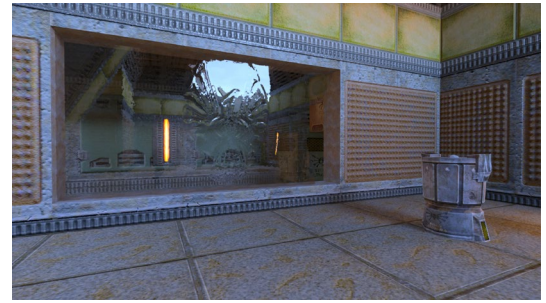
Pipeline Libraries

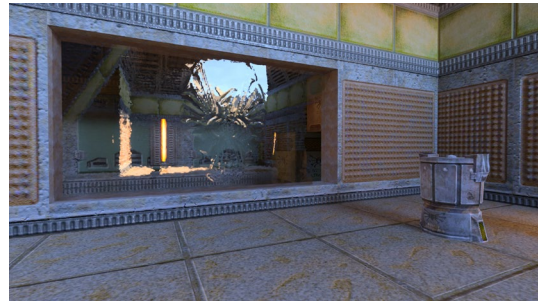
Ray Tracing Pipelines can use many shaders
Potentially orders of magnitude more shaders (1000s) than traditional applications to handle diverse tracing techniques and material types

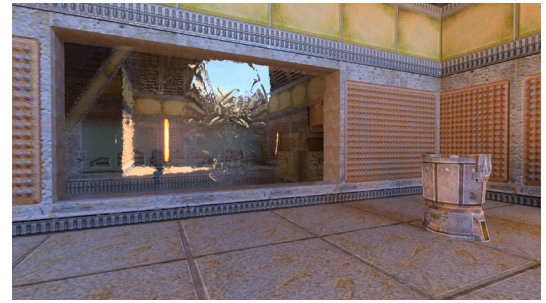
Compilation Bottleneck
Compiling many shaders into a Ray Tracing Pipeline can be computationally intensive and cause application bottlenecks and stuttering

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

Primary Ray + Ambient Light


Reflection + Direct Local light / Shadow


Indirect Bounce


Indirect Sun Light / Shadow

Multiple shaders used to build complex lighting in a Quake 2 scene

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

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

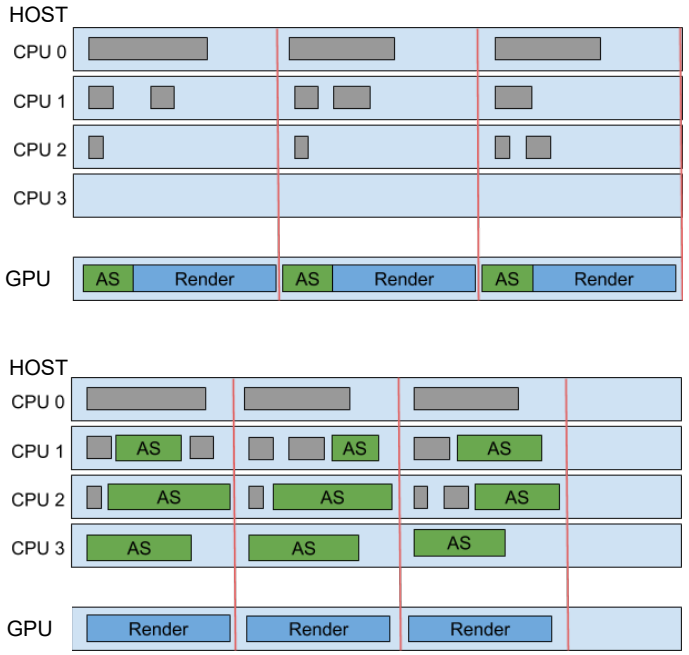
Build Acceleration Structure on Host

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 together 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

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

Vulkan Ray Tracing and DXR

Similar Acceleration Structure / Ray Tracing Shader architectures enables straightforward porting of raytracing functionality between Vulkan Raytracing and DXR

Including re-use of ray tracing shaders written in HLSL

	Vulkan Ray Tracing	DX12 / DXR
Ray Tracing Pipelines	Yes	Yes
Ray Queries	Optional	DXR 1.1
Language for Ray Tracing Shaders	GLSL or HLSL	HLSL
Pipeline Libraries	Yes	Yes
Build Acceleration Structure on Host	Optional	No
Deferred Host Operations	Optional	No
Capture/Replay Support for Tools (e.g. RenderDoc)	Optional	No

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

Call for Industry Feedback

- **Khronos seeking developer feedback before finalizing Vulkan ray tracing specs**
  - Including what functionality should be mandated
  - GitHub Release Tracker, GitHub Feedback Issue or through the Khronos Developer Slack
- **Additional Resources**
  - Specifications available on the Vulkan Registry
  - Blog with many more details
  - Press Release

Real time Vulkan Ray Tracing effects in Wolfenstein: Youngblood

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

Background

March 2020

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

Khronos Connects Software to Silicon

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

>150 Members ~ 40% US, 30% Europe, 30% Asia

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

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

Khronos Active Initiatives

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

Vulkan, ANARI, OpenGL ES, OpenGL, WebGL, EGL, OpenGL SC, Vulkan SC

3D Assets
Authoring
and Delivery

glTF, COLLADA, 3DCommerce

Portable XR
Augmented and
Virtual Reality

OpenXR

Parallel Computation
Vision, Inferencing,
Machine Learning

SPIR, SYCL, OpenCL, NNEF, OpenVX

KHRONOS SAFETY CRITICAL ADVISORY FORUM | SC™

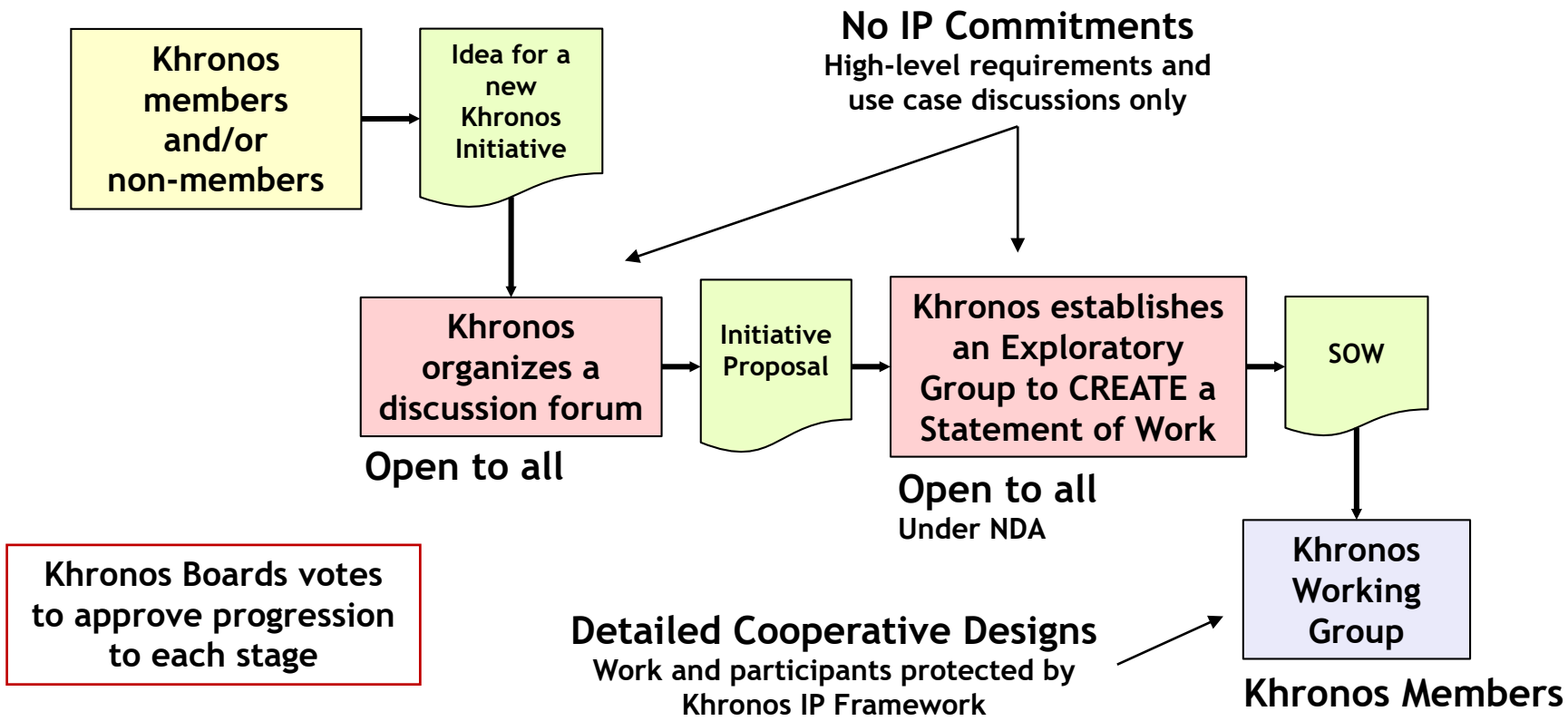
Guidelines for creating APIs to streamline system safety certification

KHRONOS GROUP

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

Khronos Exploratory Group Process

Enables Khronos and the wider industry, to inclusively build consensus on whether to undertake a new standardization initiative - without committing any company to IP licensing obligations



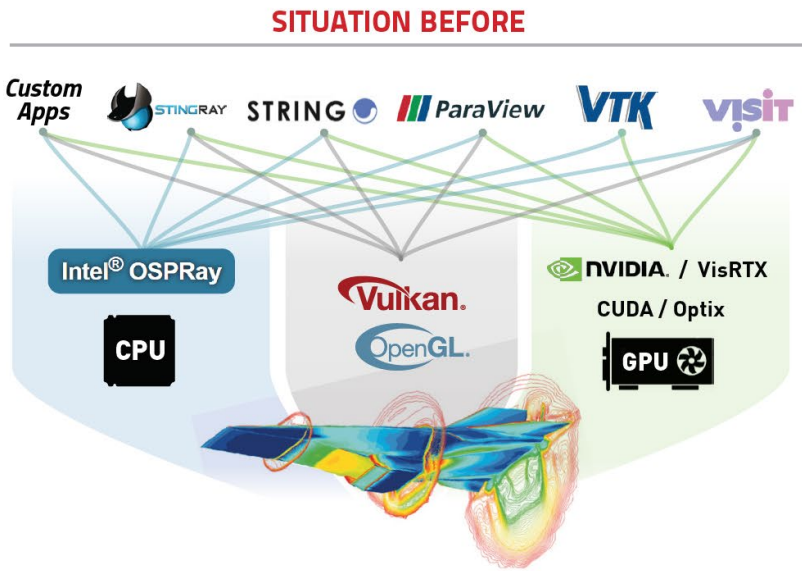
This work is licensed under a Creative Commons Attribution 4.0 International License

ANARI Working Group

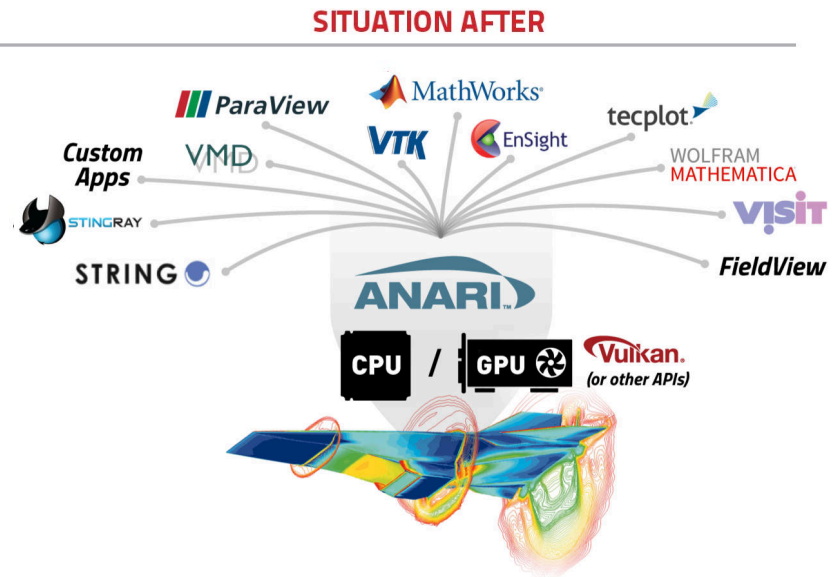
ANARI Working Group Announced 3rd March 2020

ANARI - Analytic Rendering Interface API

Analytic Rendering is image generation performed primarily to gain and communicate insights into complex data sets primarily for scientific visualization and data analytics



Visualization Apps and Engines must be ported to multiple APIs

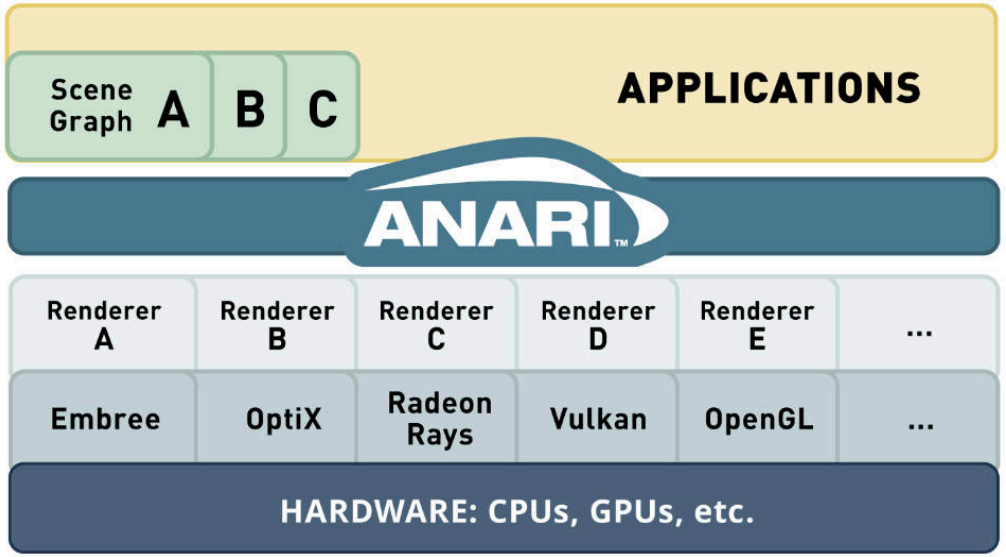


Cross-vendor API to provide access to state-of-the-art rendering across multiple platforms

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

ANARI Architecture and Support

Rather than specifying the details of the rendering process, ANARI will enable a visualization application to simply describe the relationship between objects in a scene to be rendered and leave the details of the rendering process to a backend renderer

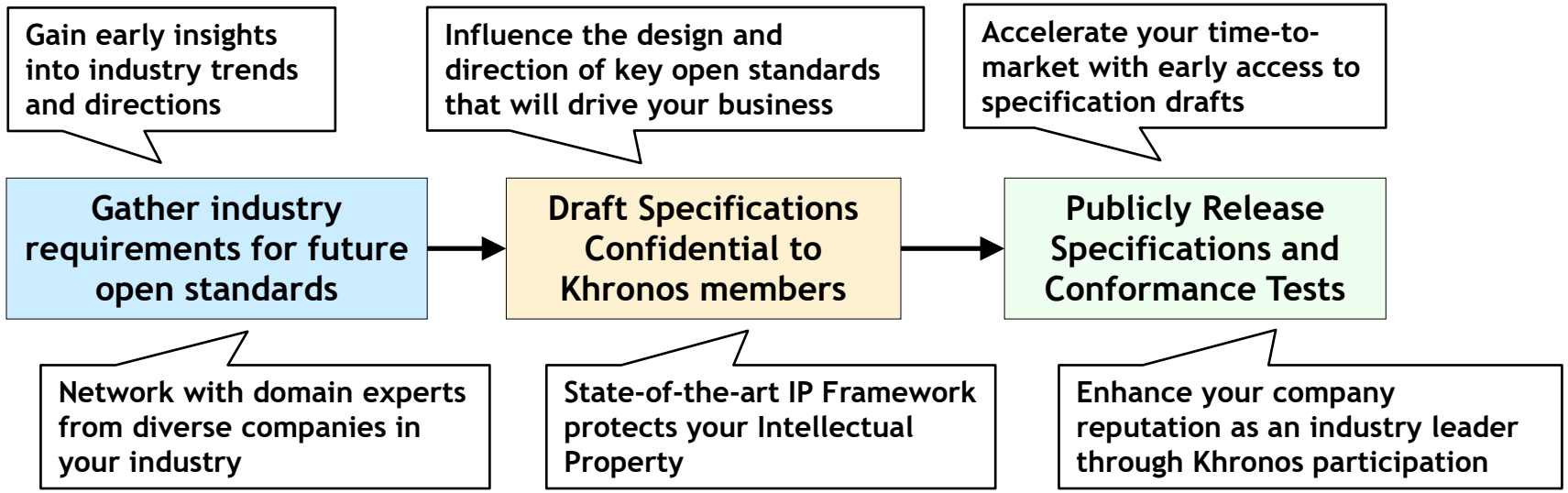


ANARI
Industry Support

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

Get Involved!

- Information on Khronos Standards: www.khronos.org
- Any company is welcome to join Khronos: https://www.khronos.org/members/



Benefits of Khronos membership

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