Vulkanised 2026

The 8th Vulkan Developer Conference
San Diego, USA | February 9-11, 2026

Vulkan: 10 Year API and Ecosystem Retrospective

Ralph Potter, Vulkan WG Chair
Karen Ghavam, LunarG CEO

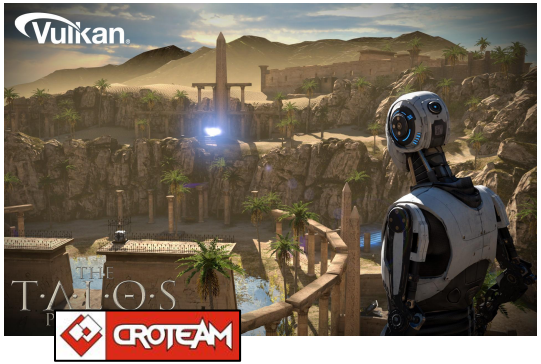
Happy Birthday Vulkan!

Vulkan 1.0 launched on February 16, 2016

A lot changed in the last decade

Significant portions of the API have been redesigned…

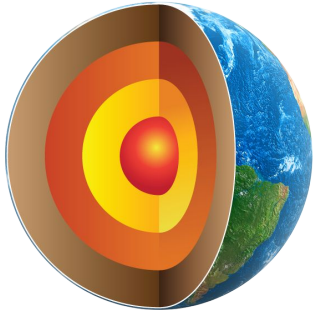
…but our original motivations and principles from 1.0 remain



Vulkan: the origin story

- **The Force Awakens: October 2012**
  - GL Common TSG formed to consider a ground-up redesign of OpenGL / ES
  - Brainstorming and design sketches

- **A New Hope: June/July 2014**
  - Effort rebooted as GL Next – becomes the top priority
  - Unprecedented participation from key ISVs
  - AMD donates Mantle as a starting point



- **Renamed and disclosed at GDC 2015**

- **Public launch on February 16, 2016**

Vulkan™

KHRONOS GROUP

Vulkan vision and goals at project launch

- **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
  - Full support for both desktop and mobile GPU architectures
  - More predictable performance – no driver magic
  - Improved reliability and consistency between implementations

- ***We think we nailed these***

What Developers have asked for (in 2016)…

… at least developers that need and can benefit from explicit control over GPU operation

Leading Edge Graphics Functionality
Equivalent to OpenGL in V1.0

Same API for mobile, desktop, console and embedded
Defined ‘Feature Sets’ per platform
No need for ‘Vulkan ES’

General Purpose Compute
Graphics AND compute queues in a single API

Explicit API
Direct control over GPU and memory allocation for less hitches and surprises

Efficient Multi-threading
Use multiple CPU cores to create command buffers in parallel

Precompiled Shaders
SPIR-V for shading language flexibility including C++ Programming (future)

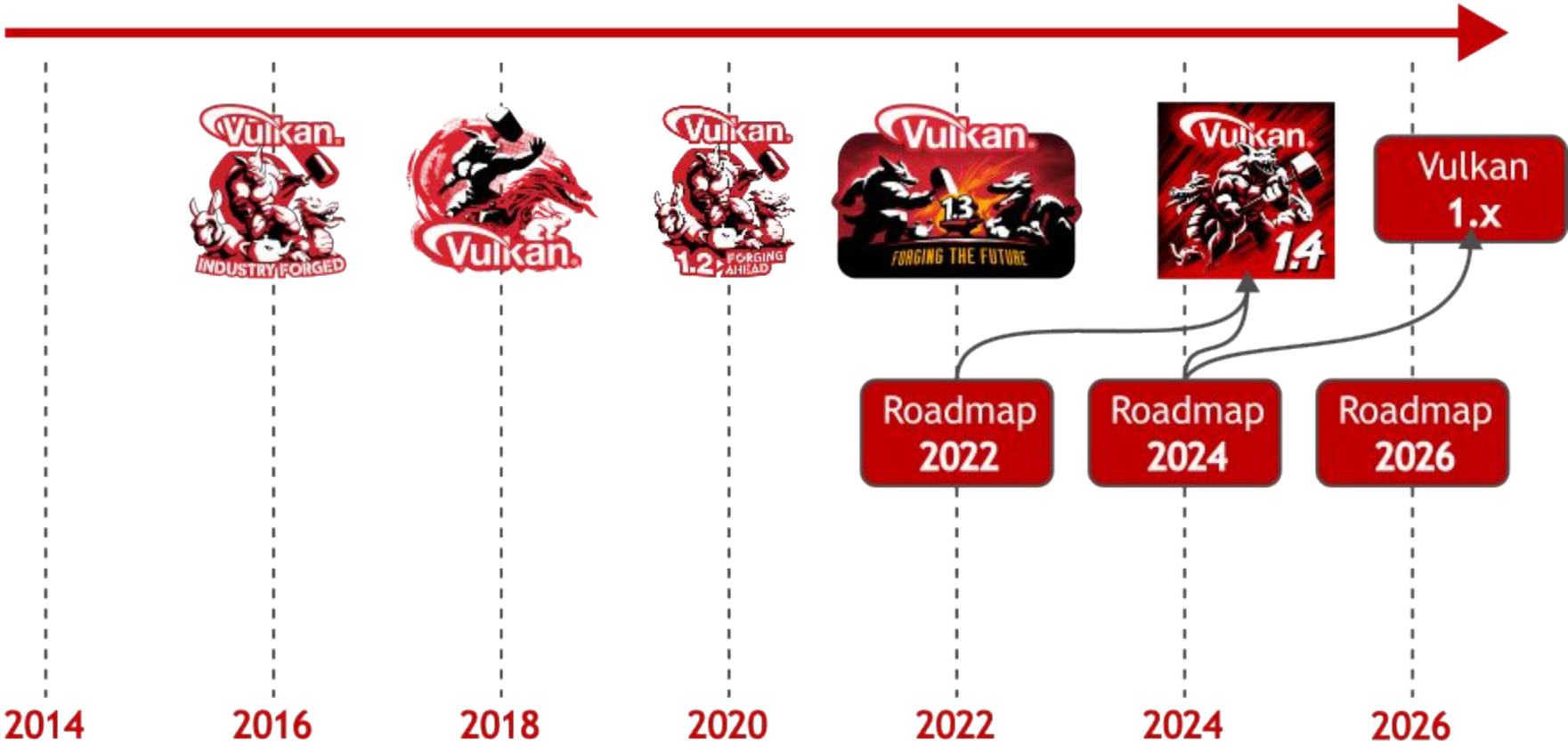
Low Driver Overhead
Thinner, simpler driver reduces CPU bottlenecks and latency

Clean, Streamlined API
Easier to program, implement and test for cross-vendor consistency

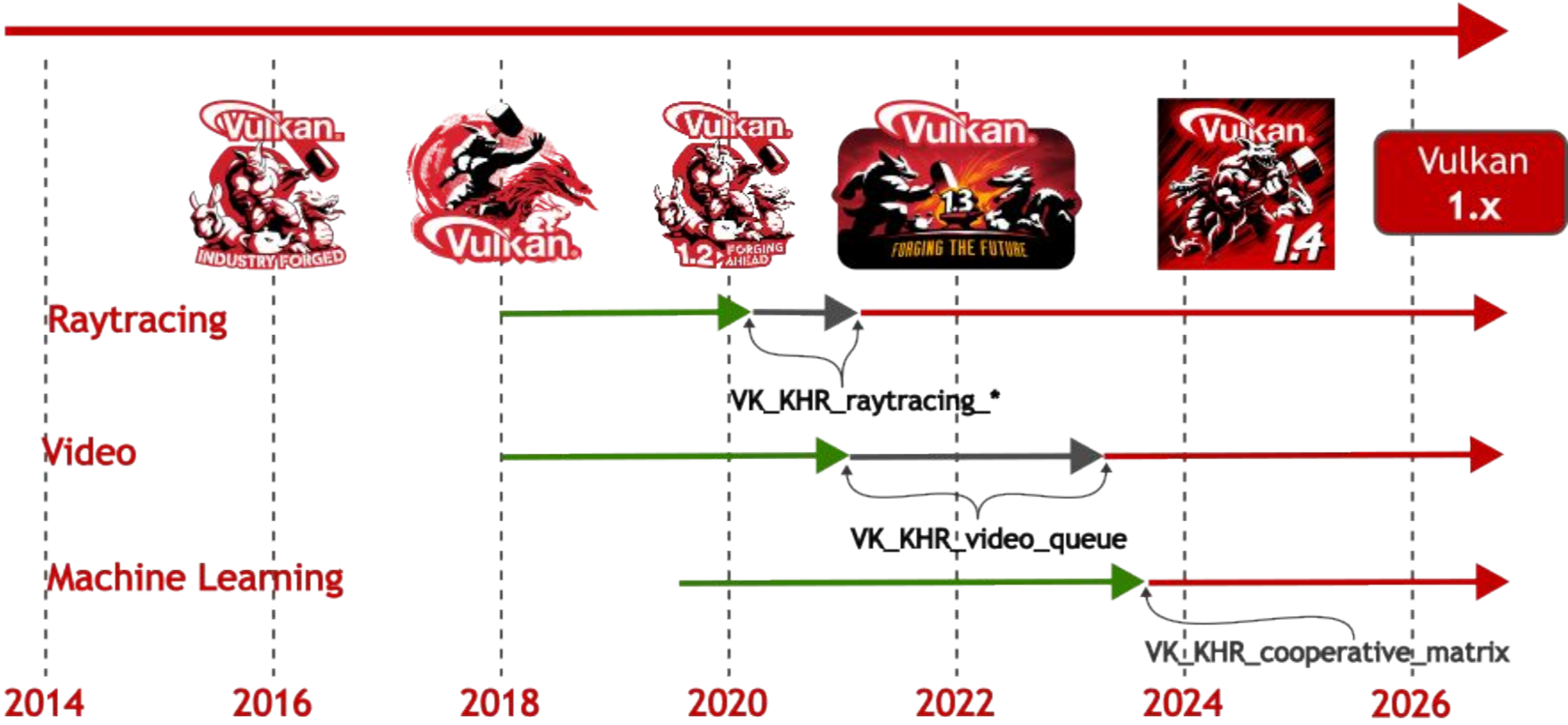
FUNCTIONALITY **PERFORMANCE** **PORTABILITY**

KHRONOS GROUP

Core Releases



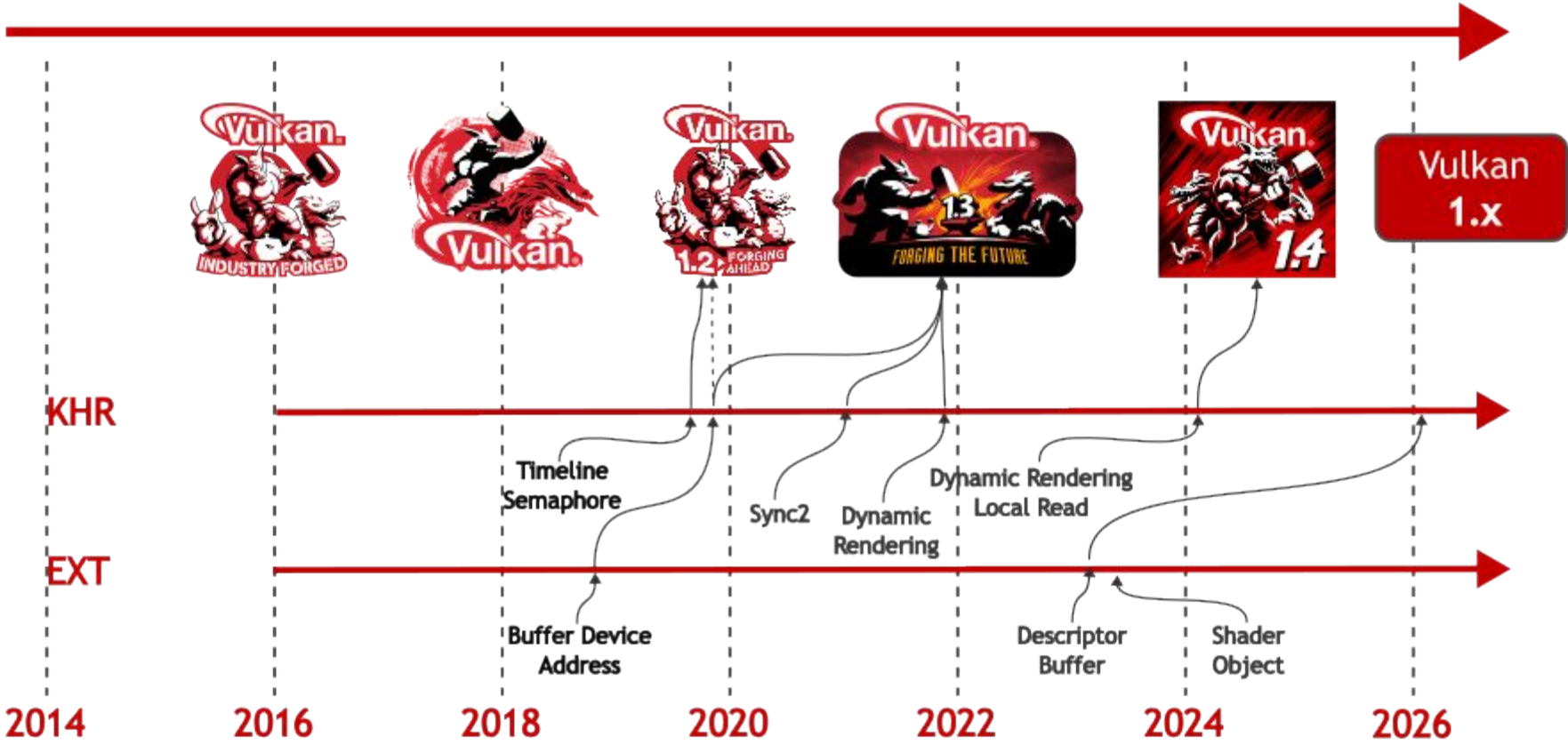
Technologies



Managing Legacy APIs

- We made some informed choices on how to define a forward looking API in 2016
  - The foundations of the API are solid
  - Not all of our predictions were perfect
- APIs are forever - dropping support for older content is not commercially viable
- We ***are*** continually working to make the API a better fit for today's content
- We are working to make the expanding API surface manageable - see my talk this afternoon

API Evolution



API Evolution

- Vulkan 1.0 leaned heavily on expecting applications to know state upfront
- Pipelines baked in any state that might affect the compiled binaries
  - Including the RenderPass
  - Limited dynamic state
- RenderPasses attempted to unify mobile and desktop GPU usage patterns
  - Proved awkward to use in practice even on mobile
  - Additional complexity for no reward if you only care about IMR GPUs
- Expected greater use of pre-recorded command buffers

API Evolution

Vulkan in 2026*:

- ~~RenderPasses~~ Dynamic Rendering
- ~~Descriptor Sets/Pipeline Layouts~~ Descriptor Heap
- ~~Image Layouts~~ KHR_unified_image_layouts
- ~~Static pipeline state~~ Increasing dynamic state (EDS1/2/3)

Predicting the Future:

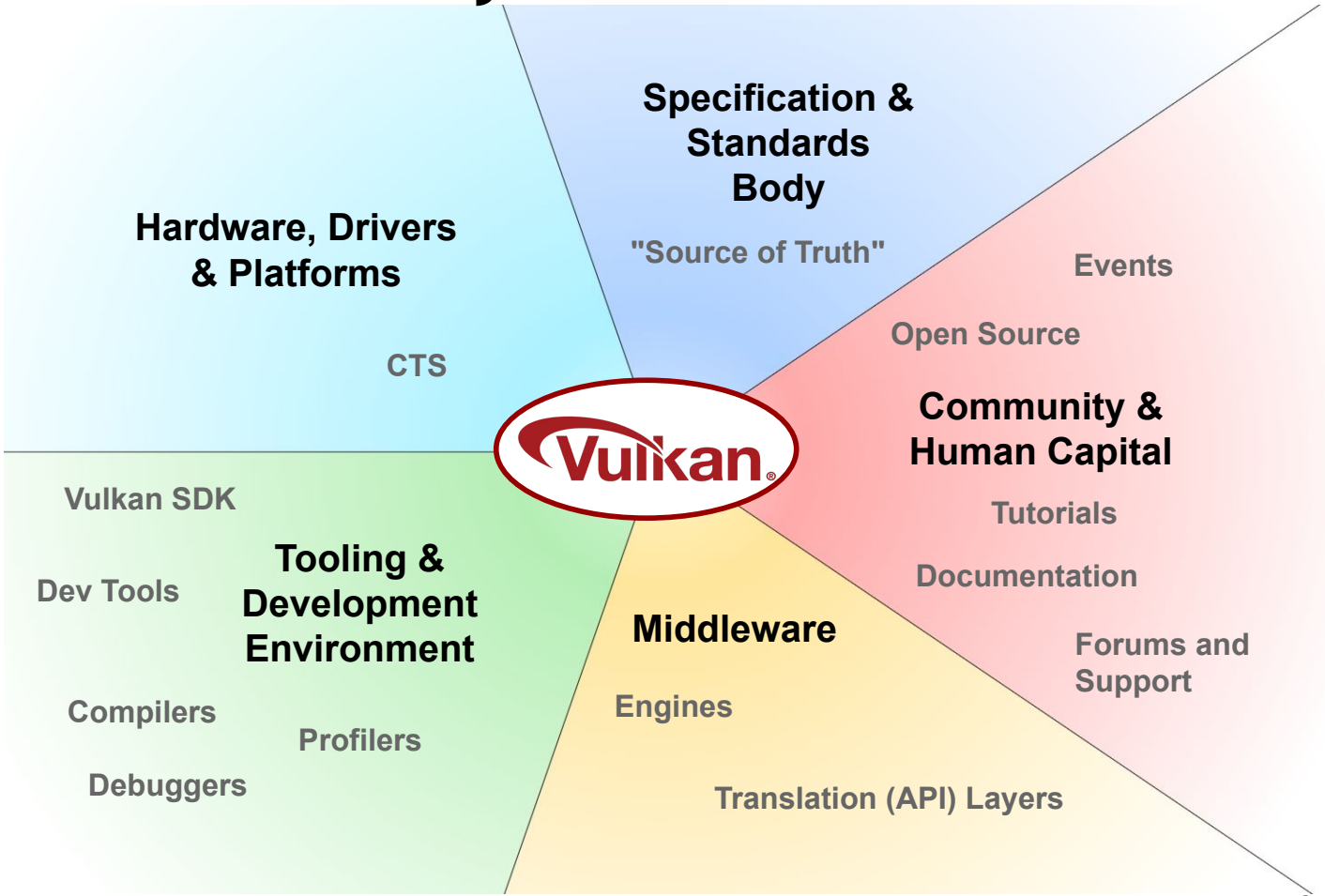
- Revisiting pipelines - ideas informed by EXT_shader_object
- Increasing use of buffer device address
- More device-driven workloads

* Some markets/products have slower upgrade cycles and require broad device support

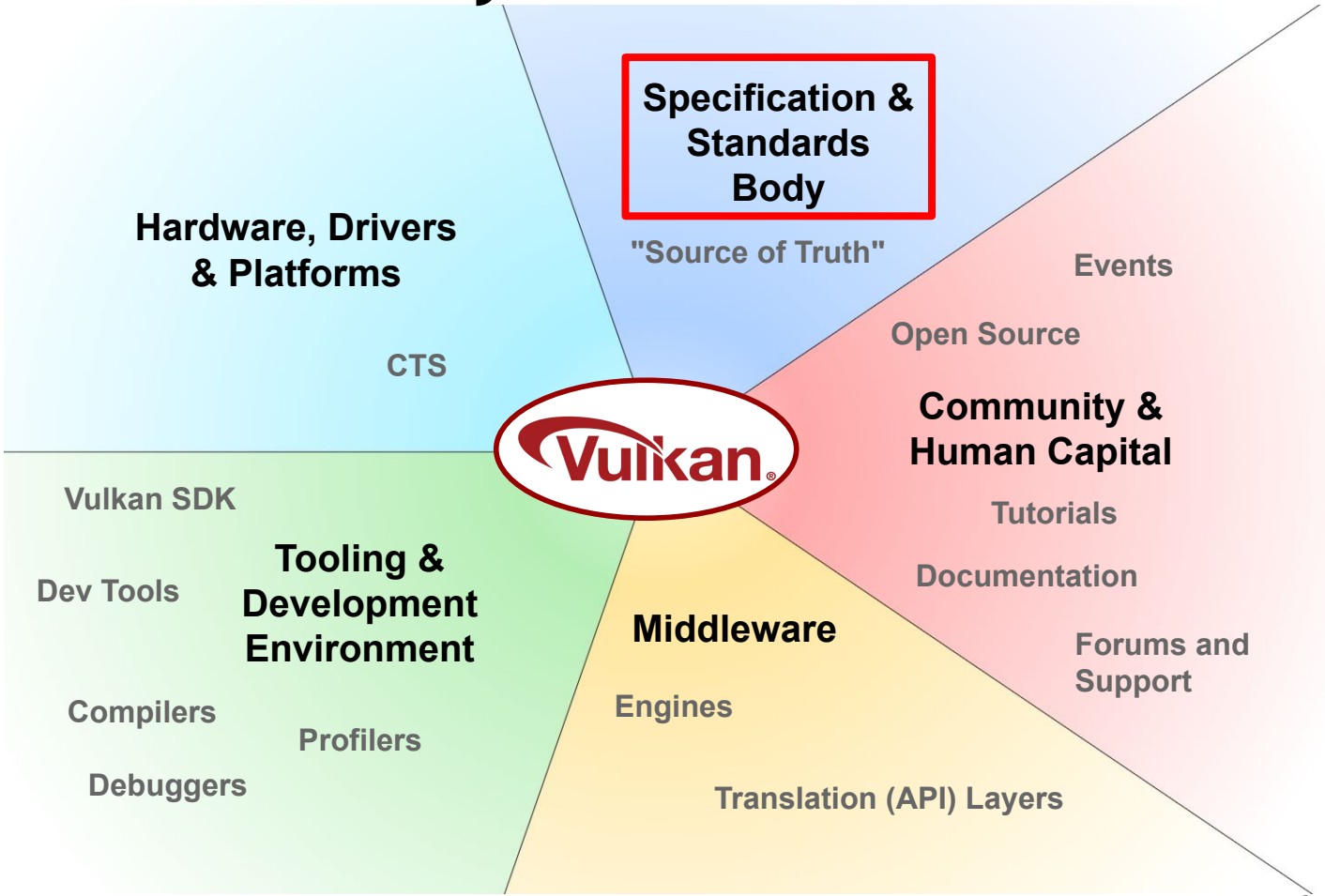
Observations

- Greater inertia than we anticipated - engines were designed for DX11 API model
  - Assumptions baked in
  - Rewrites take years and require a strong commercial need
- Pressure for API convergence
  - Commercial pressure for major engines to support DX12 for desktop/console
  - Major design divergence between DX and Vulkan is a pain point
  - Goes both ways - DX has picked up enhanced barriers and render passes

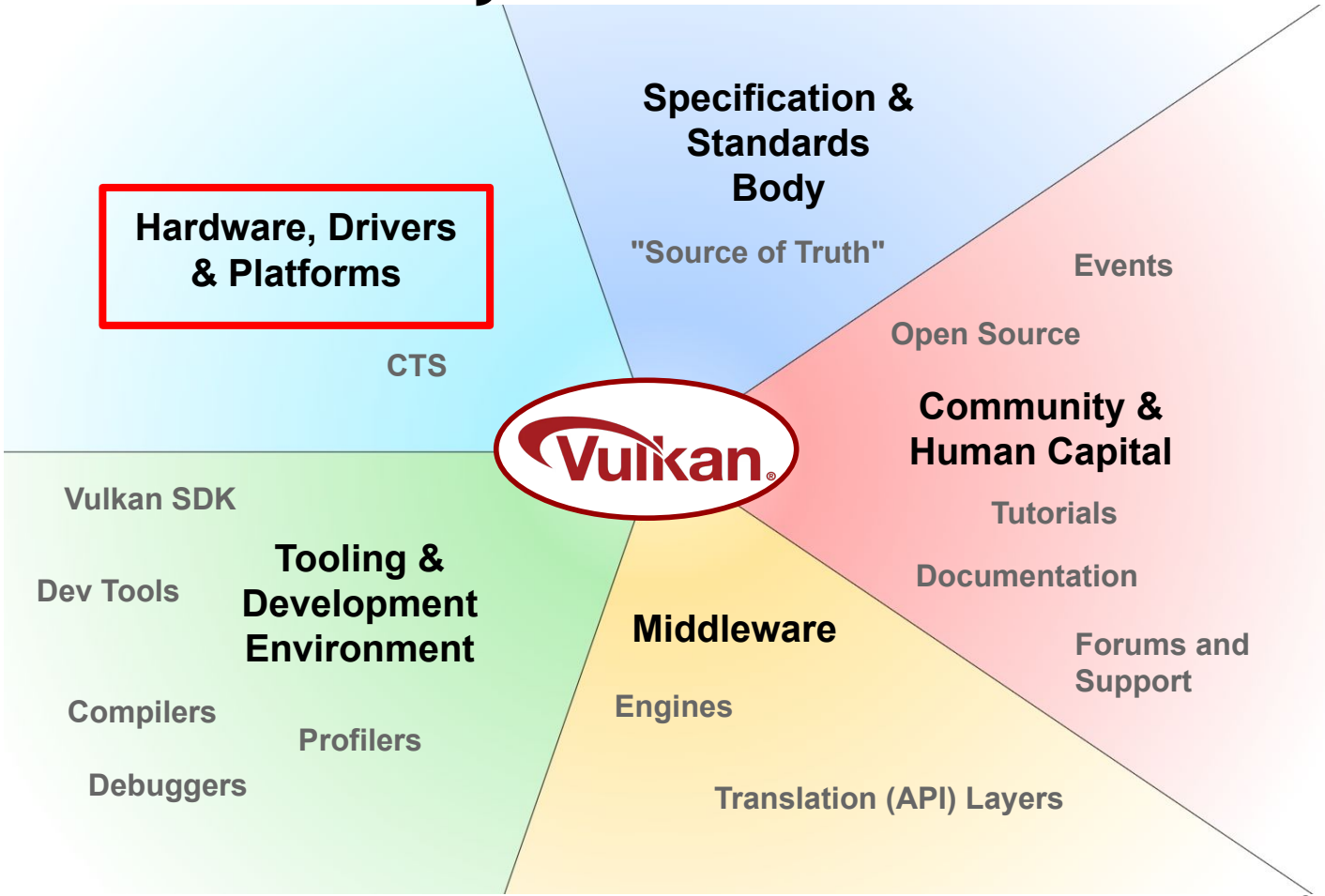
It Takes an Ecosystem!



It Takes an Ecosystem!



It Takes an Ecosystem!



Hardware, Drivers, and Platforms

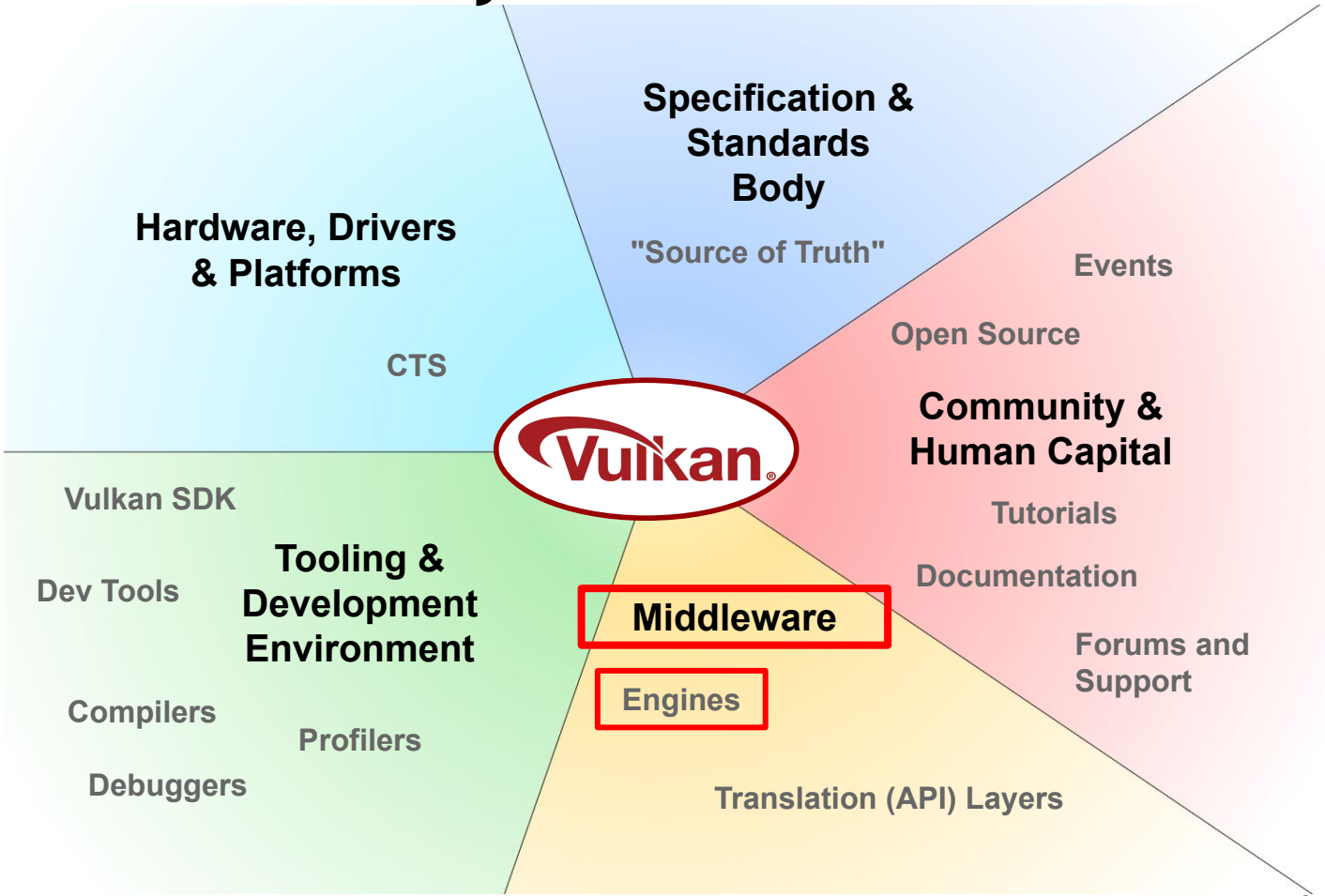
HARDWARE & DRIVERS:

AMD
BROADCOM
intel
NVIDIA
SAMSUNG
MEDIATEK
QUALCOMM
arm
Google
HUAWEI
VeriSilicon
Imagination

PLATFORMS:

Windows
Windows on ARM
OpenHarmony
STEAMOS™
Raspberry Pi OS
NINTENDO SWITCH
Android
chromeOS

It Takes an Ecosystem!

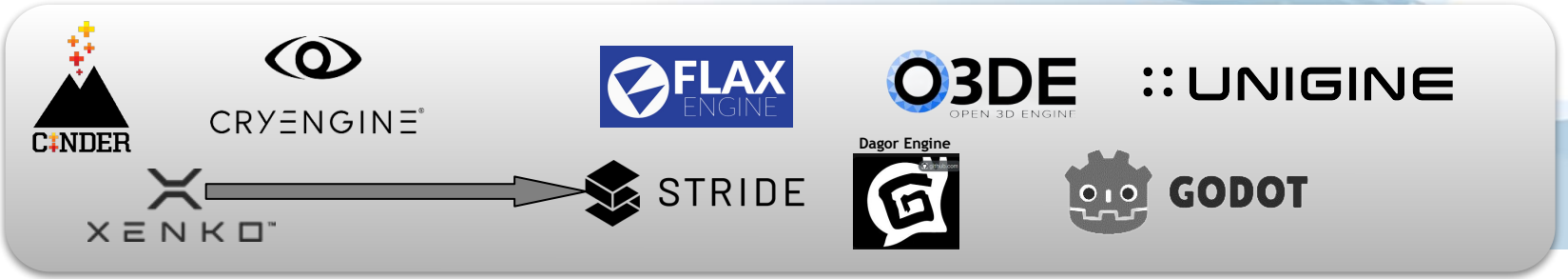


Middleware - Engines

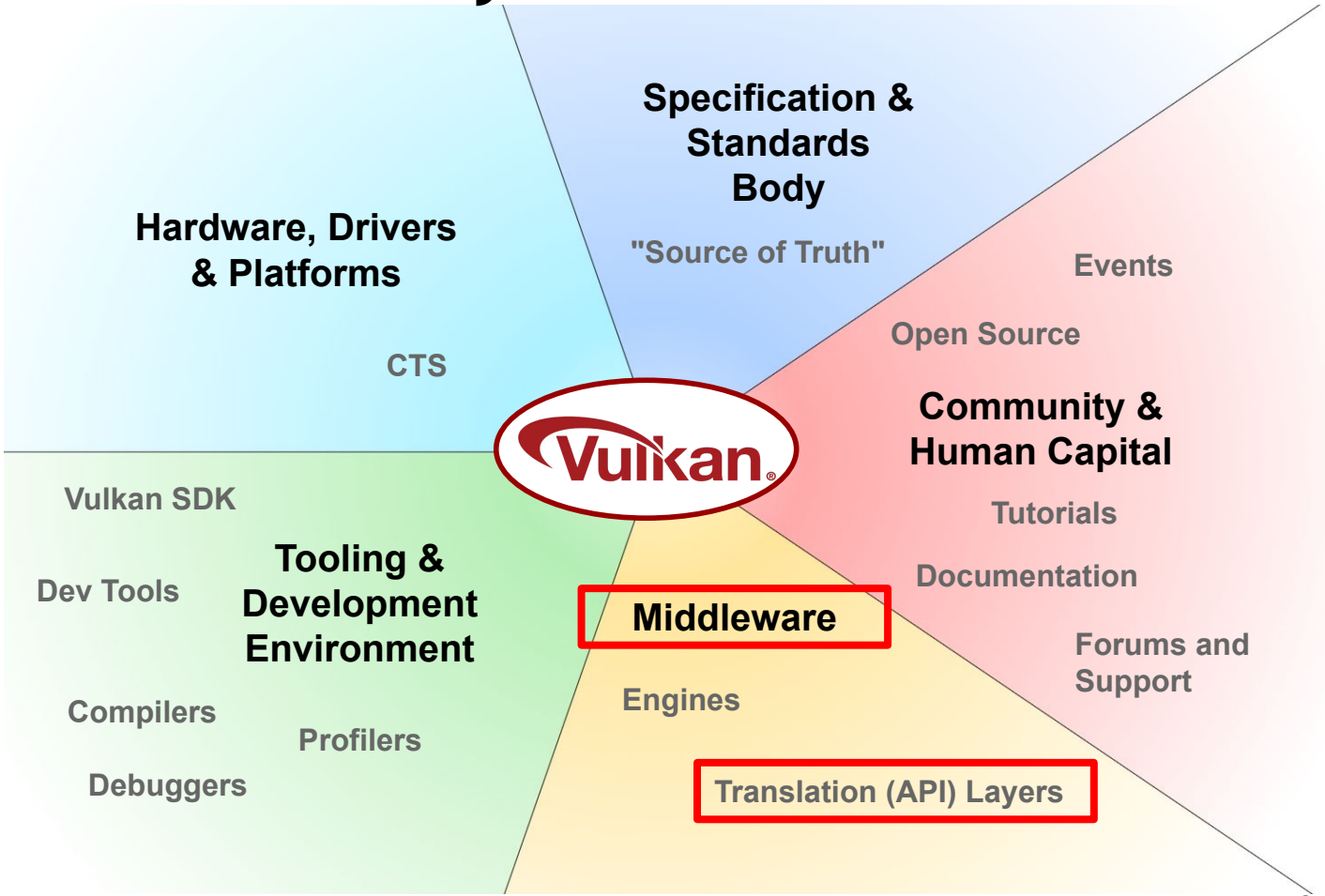
2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026

Vulkan Delivers Value Driving Engine Ports

Vulkan Quickly Gains Momentum!



It Takes an Ecosystem!



MiddleWare - API Layering

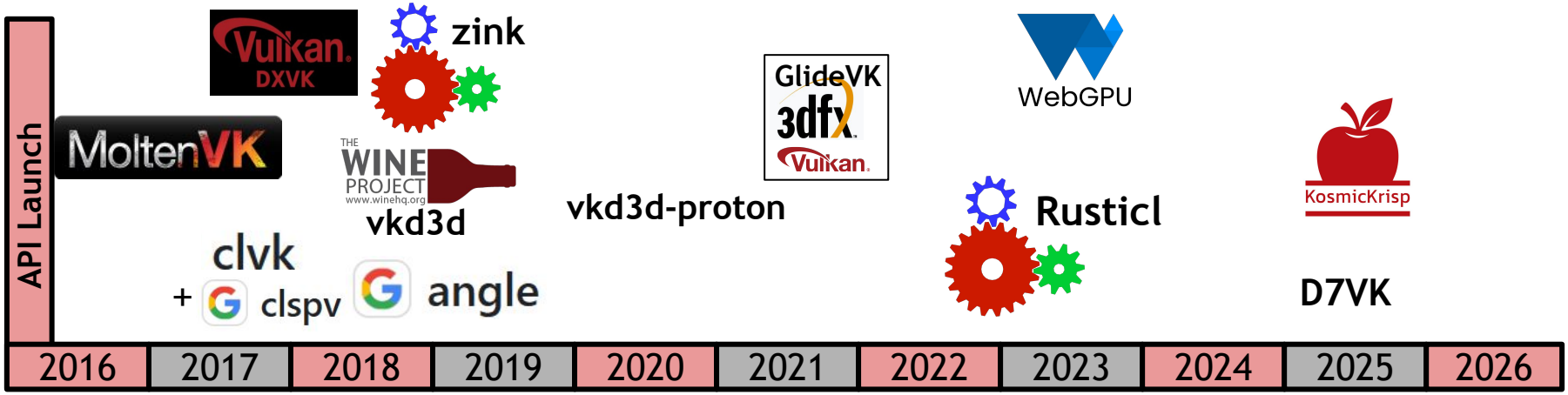
Enabled by growing robustness of open-source compiler ecosystem using SPIR-V

Layers Over	Vulkan	OpenGL	OpenCL	OpenGL ES	DX12	DX7-11	3dfx Glide
Vulkan	Venus	Zink	clspv + clvk/ANGLE Rusticl + Zink	ANGLE Zink	vkd3d-Proton vkd3d	DXVK D7VK WineD3D	GlideVK
OpenGL	Ashes			ANGLE		WineD3D	
DX12	Dozen	Microsoft 'GLOn12'	Microsoft 'CLOn12'			Microsoft D3D11On12	
DX9-11	Ashes			ANGLE			
Metal	MoltenVK KosmicKrisp			ANGLE MoltenGL			

ROWS Benefit Platforms by enabling content without additional kernel level drivers

COLUMNS Benefit ISVs by making an API available everywhere
Application deployment flexibility by fighting platform fragmentation

Middleware - Expanding Vulkan's Footprint with API Layering

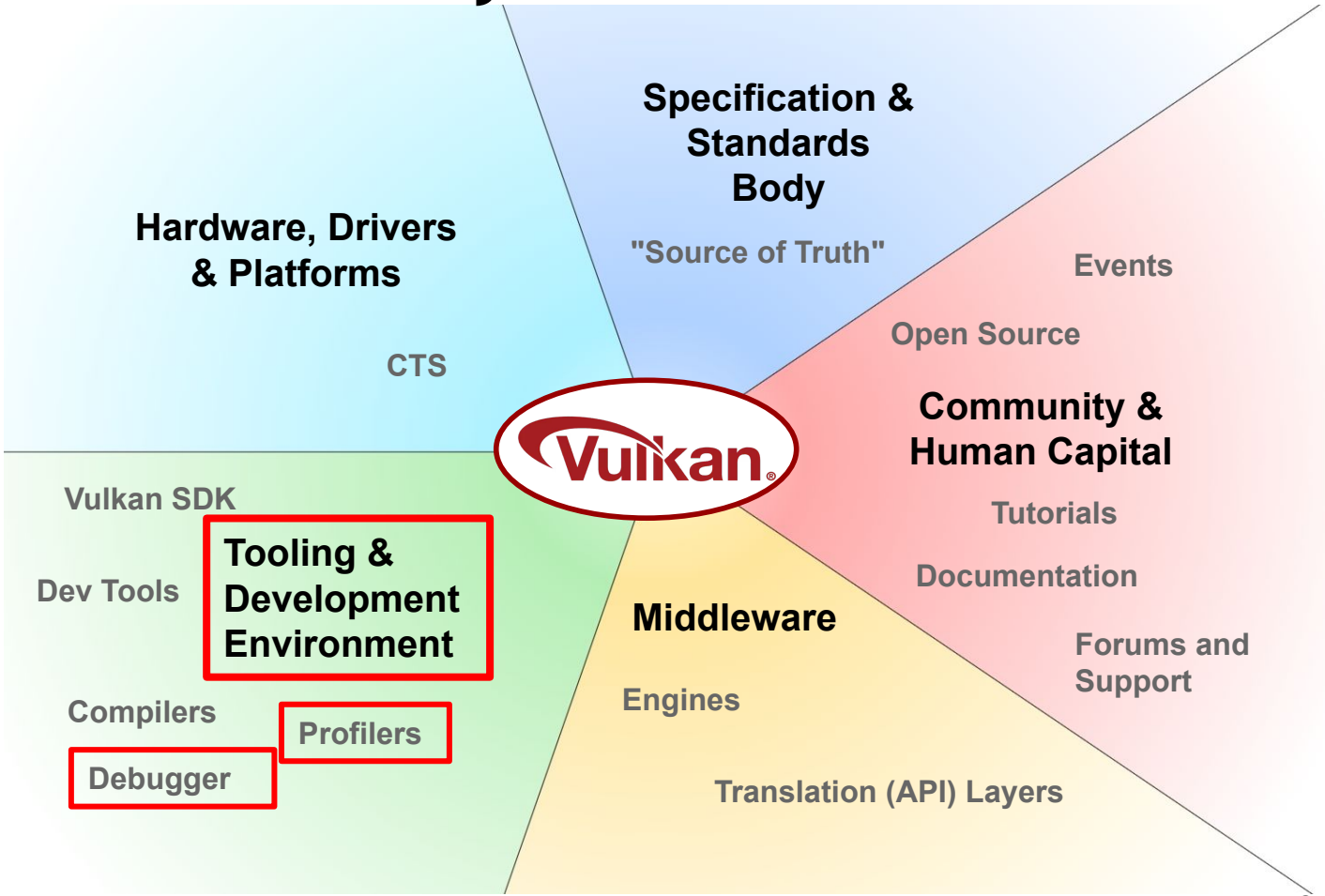


Applications that can take advantage of Vulkan as a HAL

Vulkan-based apps that can run on more platforms

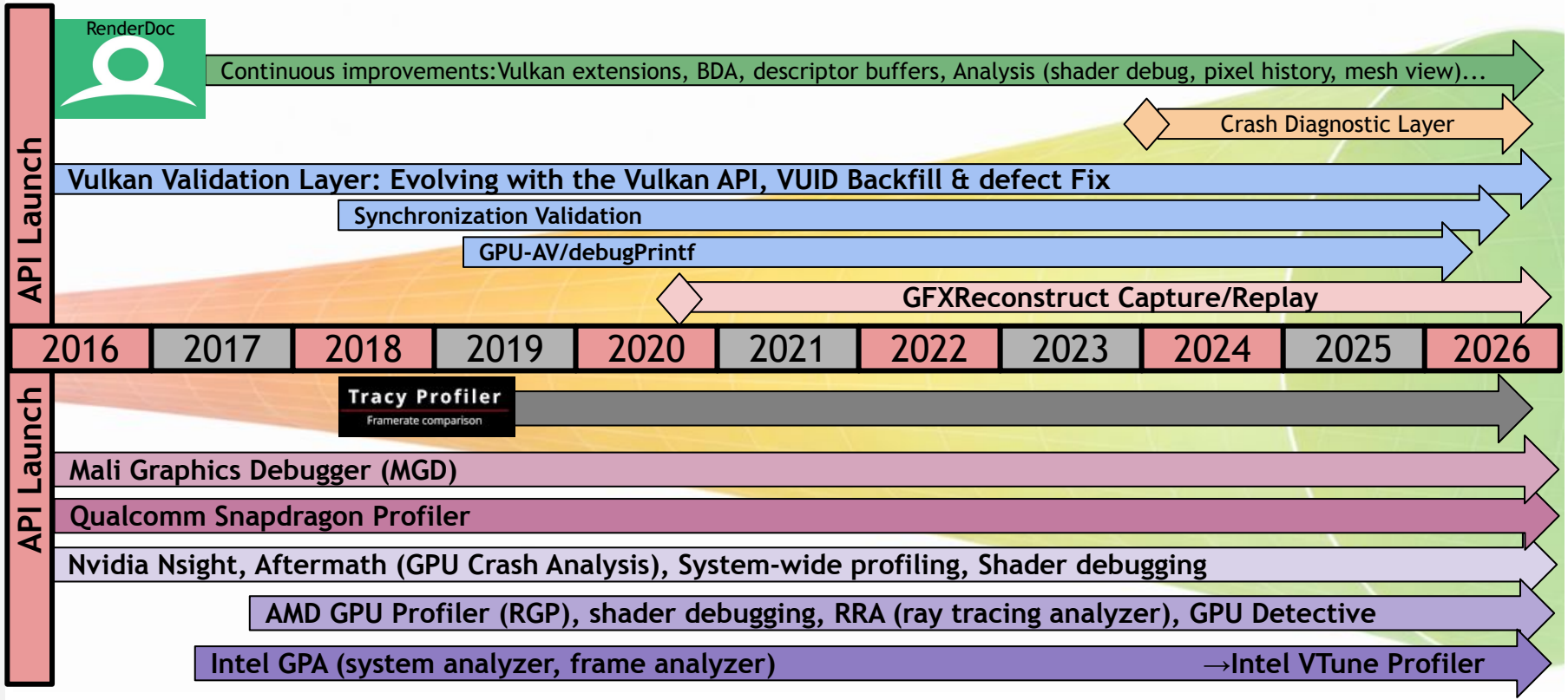
Resources required to support a broader range of platforms

It Takes an Ecosystem!

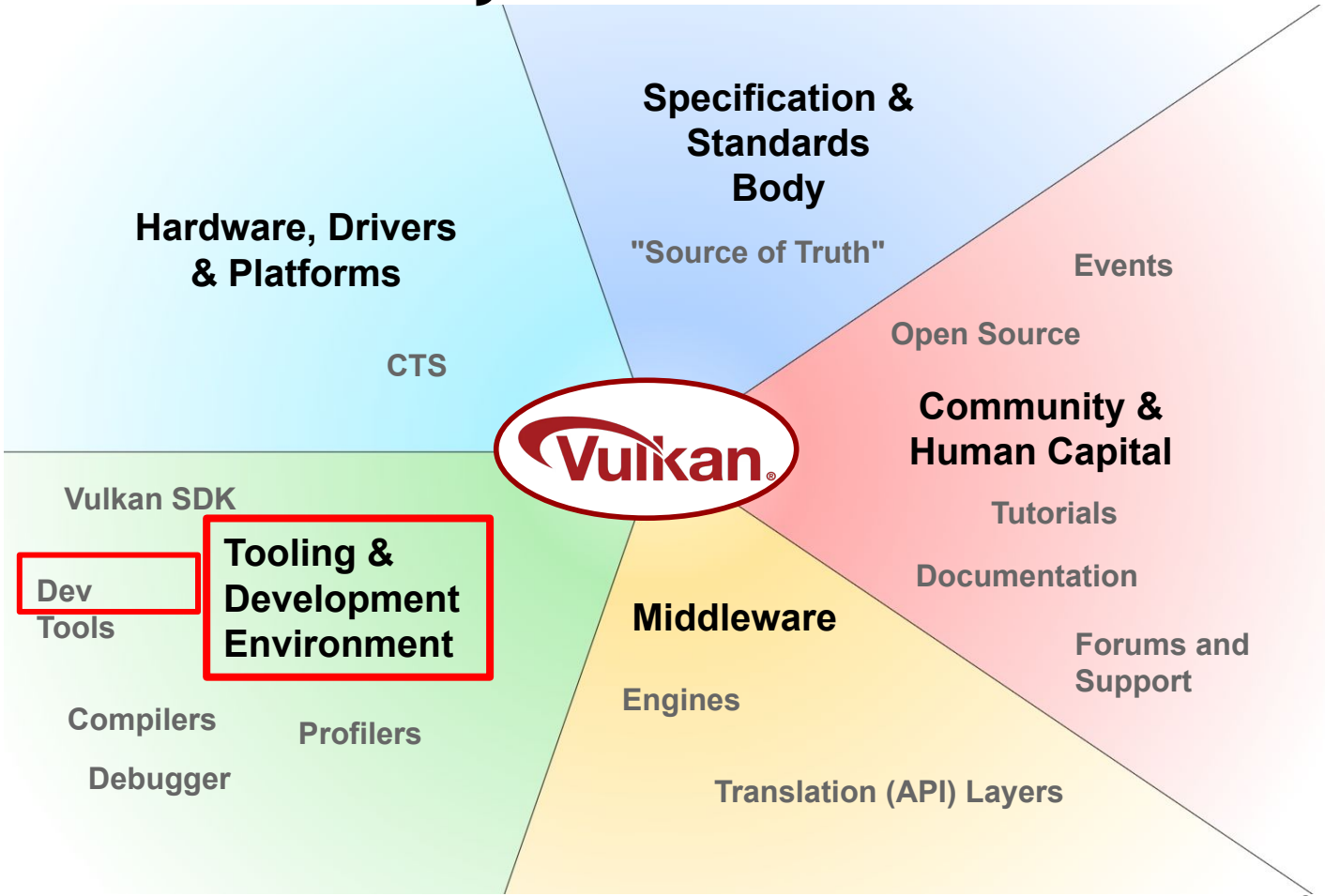


Developer Tooling - Debugging & Profiling

The expansion of the Vulkan tooling universe!

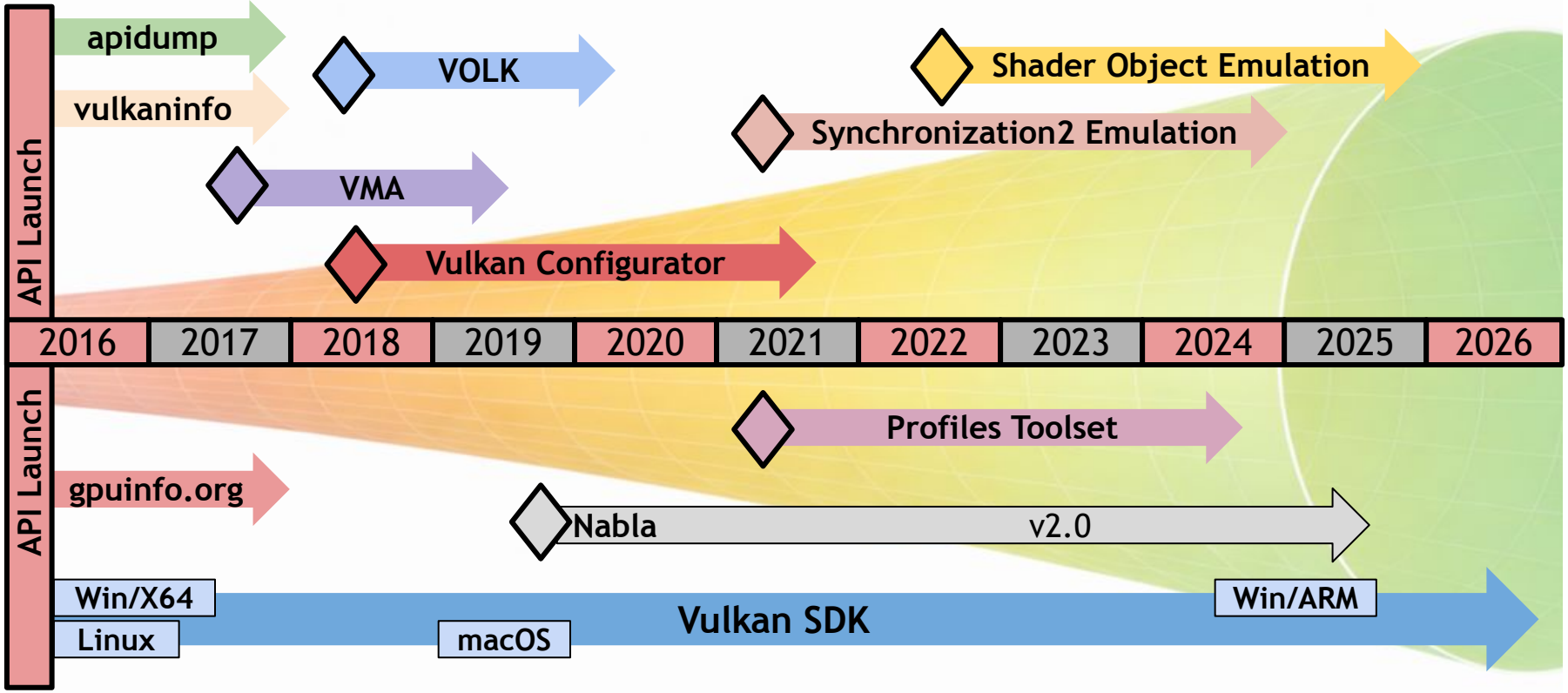


It Takes an Ecosystem!

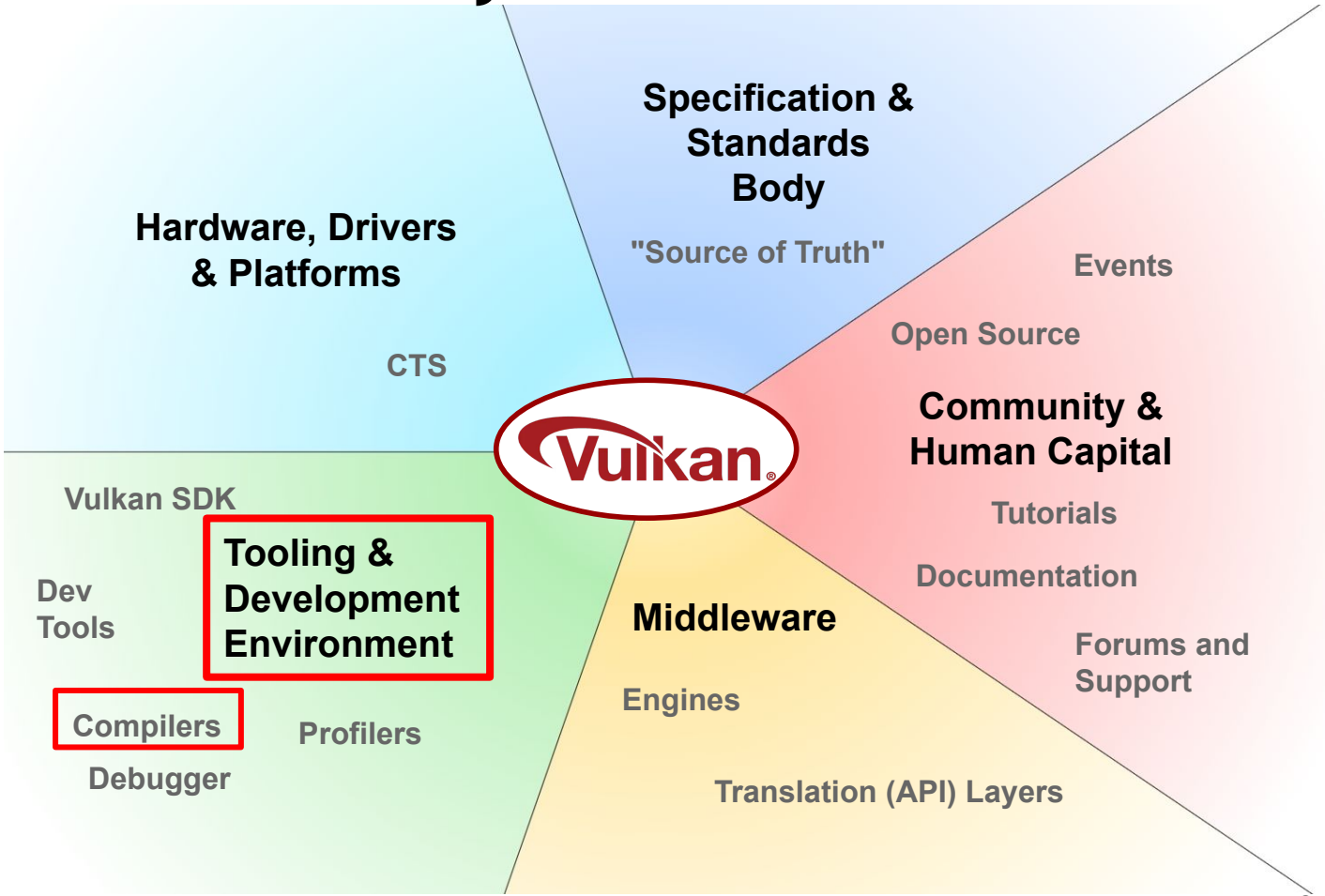


Developer Tooling - Layers & Tools

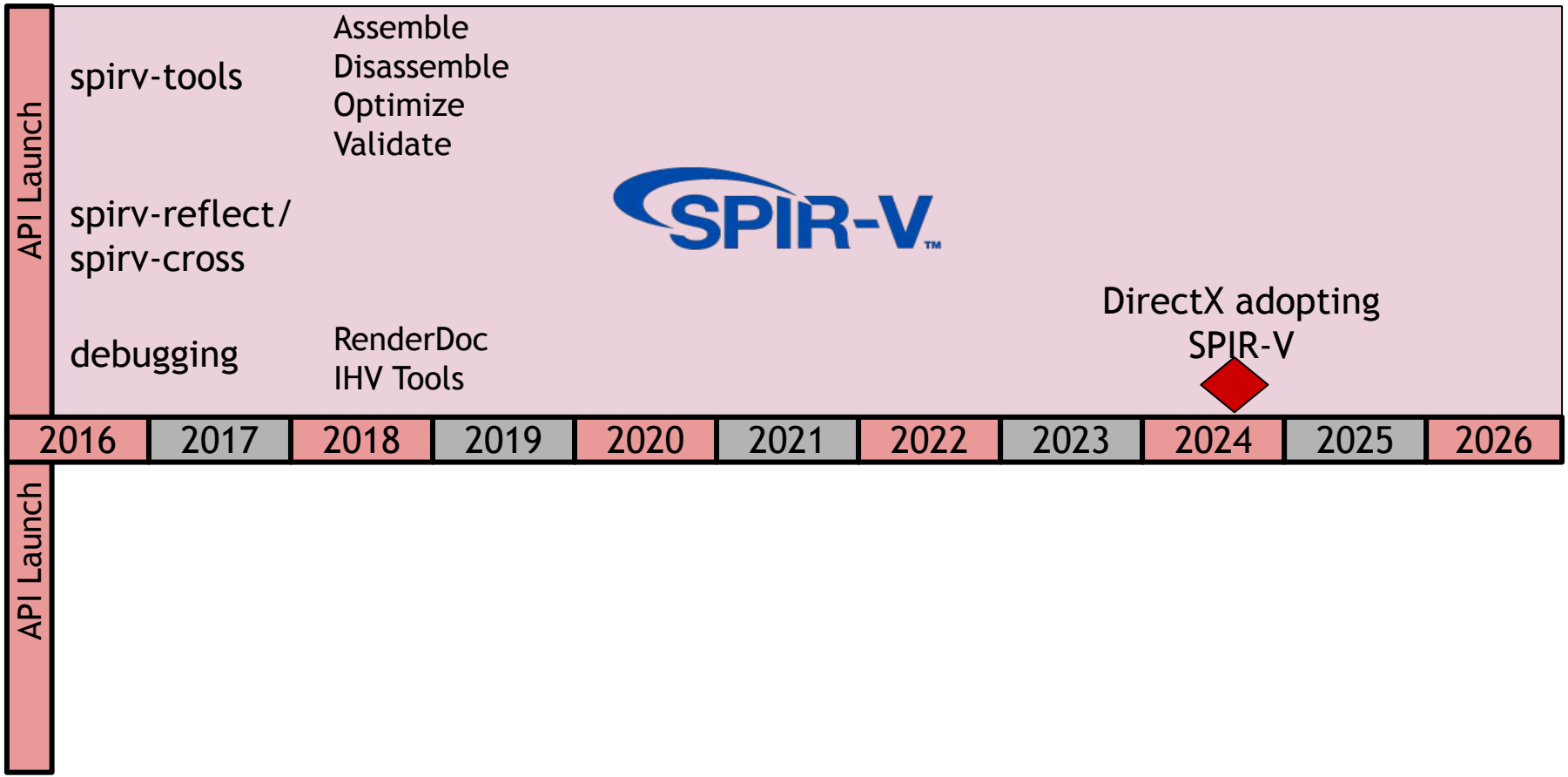
The expansion of the Vulkan tooling universe!



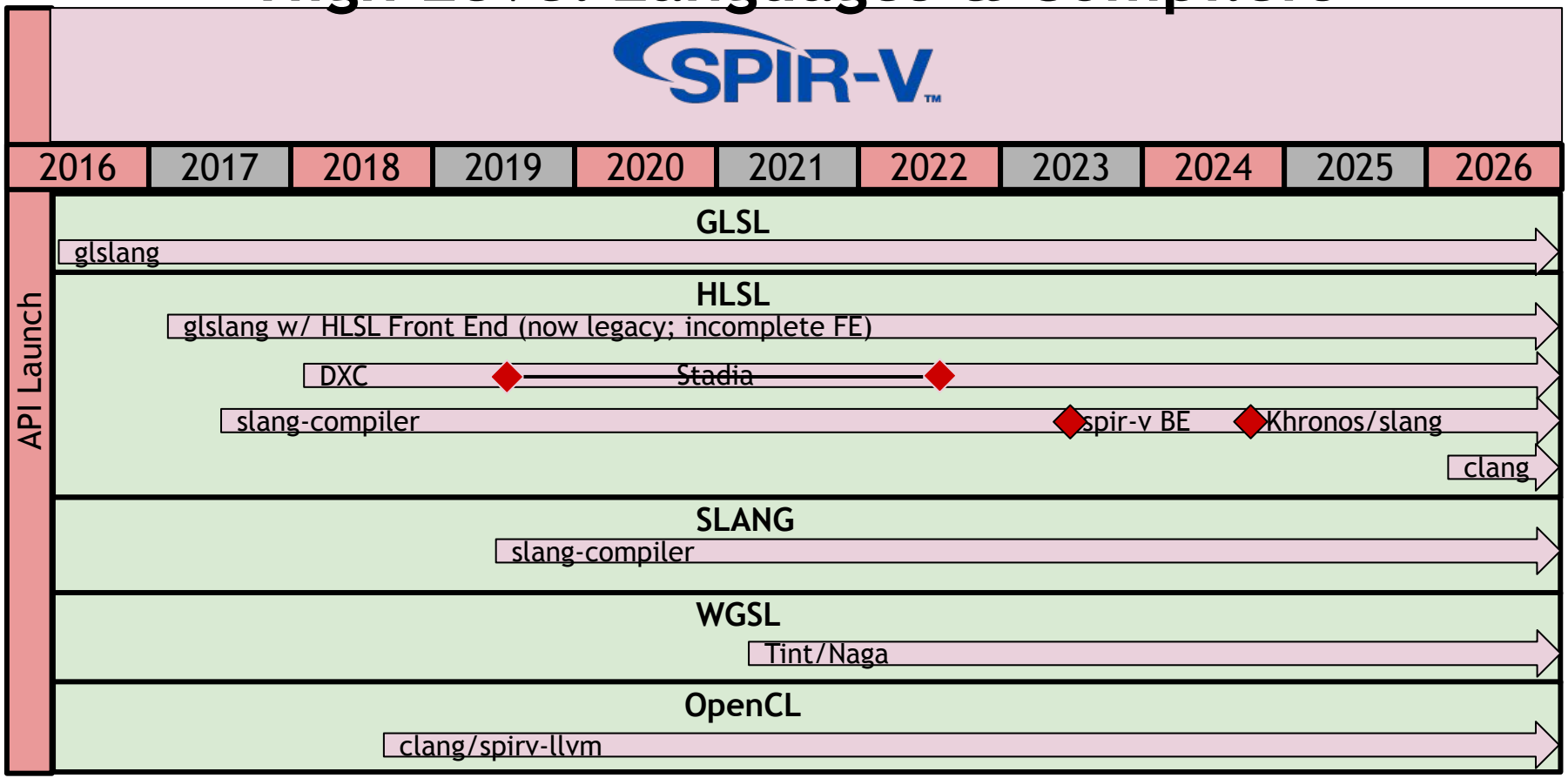
It Takes an Ecosystem!



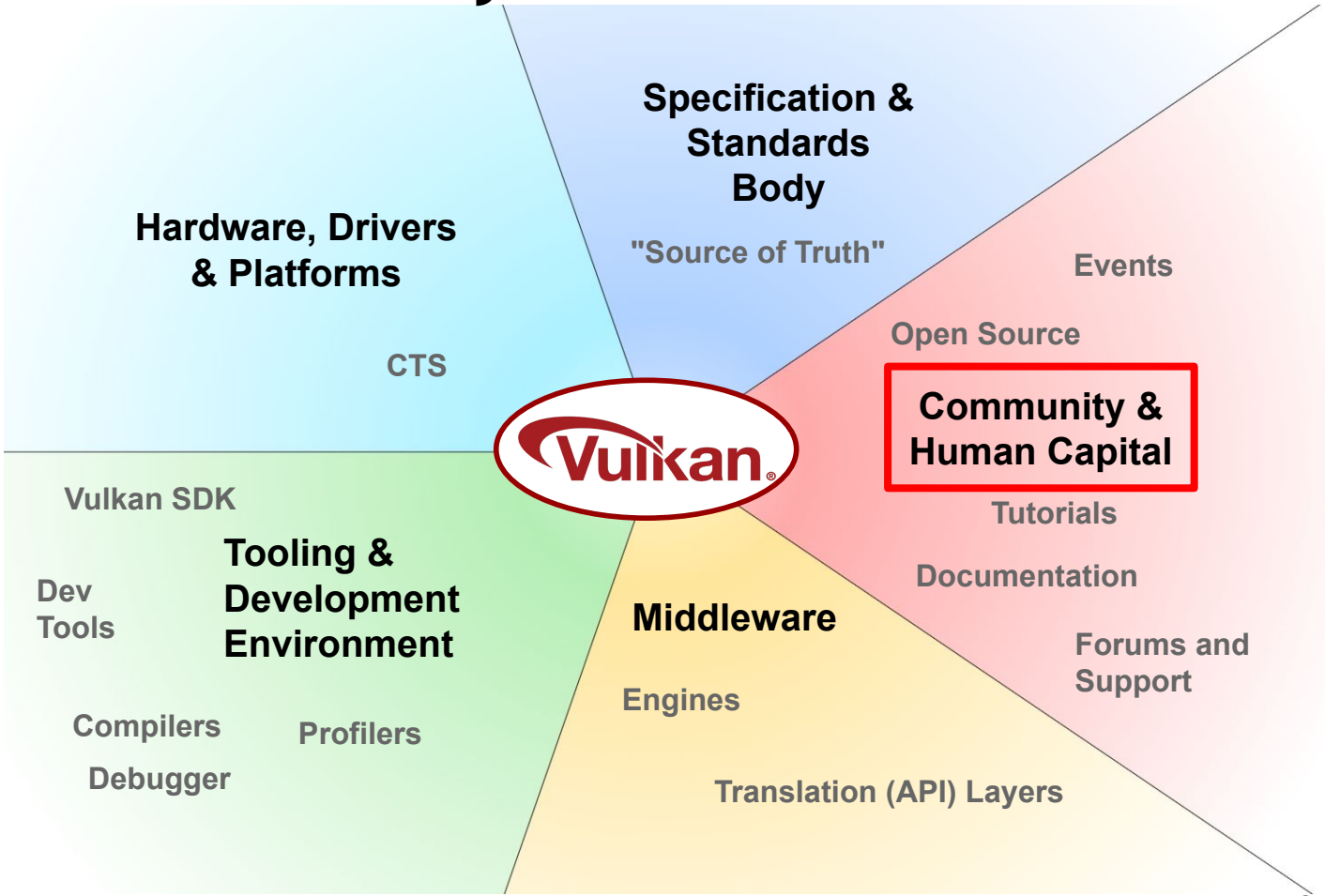
Tooling and Development Environment
Vulkan Standardizes on SPIR-V IR



Tooling and Development Environment
High Level Languages & Compilers



It Takes an Ecosystem!



Community & Human Capital - Docs/Tutorials

2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026

Khronos Documentation Project

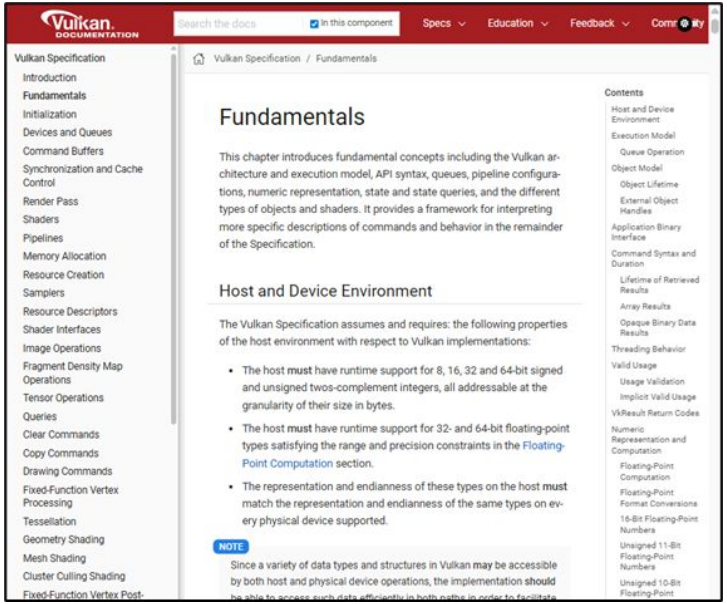
docs.vulkan.org
powered by
Antora
HOLOCHIP

https://docs.vulkan.org

At launch:
PDFs, Single Page HTML
Difficult to navigate
Huge - load and hang issues

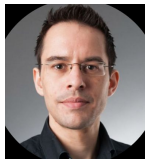
Today: Unified documentation in a single, searchable location:

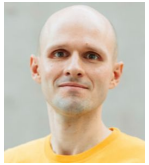
- **Vulkan specification**
- **Vulkan Guide**
- **API proposal docs**
- **Vulkan Samples documentation**
- **Khronos Vulkan Tutorial**
- **Shader language documentation**
- **…**



Community and Human Capital - Samples

2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026





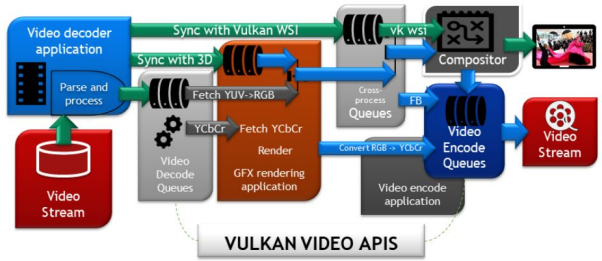
arm

Khronos Unified Samples Project

Khronos Vulkan Samples launch

Vulkan-Samples (https://github.com/KhronosGroup/Vulkan-Samples)

Vulkan-Video-Samples (https://github.com/KhronosGroup/Vulkan-Video-Samples)



Community & Human Capital

- **Forums/support and Events**
  - Reddit, Vulkan Discord, Stack Overflow,...
  - Vulkanised!
- **Open Source!**
  - KhronosGroup/VK-GL-CTS
  - KhronosGroup/Vulkan-Docs
  - KhronosGroup/Vulkan-Headers
  - KhronosGroup/Vulkan-HPP
  - KhronosGroup/Vulkan-Loader
  - KhronosGroup/Vulkan-ValidationLayers
  - KhronosGroup/Vulkan-Utility-Libraries
  - KhronosGroup/Vulkan-Tools
  - KhronosGroup/Vulkan-Profiles
  - KhronosGroup/Vulkan-Samples
  - KhronosGroup/Vulkan-Tutorial
  - baldurk/renderdoc
  - LunarG/GFXReconstruct
  - LunarG/VulkanTools
  - ...

Strong Ecosystem -> Vulkan is Everywhere

Desktop and Mobile GPUs and SOCs

AMD, arm, Broadcom, Google, Huawei, Imagination, intel, NVIDIA, MediaTek, Qualcomm, Samsung, VeriSilicon

http://vulkan.gpuinfo.org/

Desktop Games

Mobile Games

AUTODESK FUSION

aiMotive

aiSim AIR engine

AUTODESK VRED

blender

Adobe Substance 3D

Applications

Engines

source 2, GODOT, CINDER, NetEase Games, Unity, id TECH, FLAX ENGINE, UNIGINE, STRIDE, CRYENGINE, O3DE, UNREAL ENGINE

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

Lessons Learned

- **The Vulkan Ecosystem is in a good state**
- **However, looking back over the last 10 years, what might we have done differently?**
  - Assuming there were no constraints on resources…

- **Note: The Lessons Learned will be focused on the validation layers, samples, and documentation projects "close" to the Vulkan WG**
  - I am unable to represent lessons learned for engines, middleware, drivers, and all the many other parts of the ecosystem

Lessons Learned - Sanctioned Tutorial/Samples/Guide

- **The Vulkan API is not for the "faint of heart"**
- **A Khronos-sanctioned Tutorial and Samples from day one**
  - Avoid out-of-date tutorials
  - Avoid "chasing" of tutorials, which one is good?
- **A Programmer's guide (Vulkan Guide) from day one**
  - The specification is great for implementers
  - Not ideal for many developers

Lessons Learned
Validation Layers/API Sync

- **Today, the Vulkan Validation Layer is in a good state**
  - It took 10 years from launch to catch up!

- **Beginning with Vulkan Launch - Validation Layer support + new API**
  - Probably a "pipe dream". Would have significantly delayed the Vulkan launch
  - Without this, there was a lot of missing validation for multiple years

- **This policy was put in place in the Vulkan WG beginning with 1.1**
  - We still needed better discipline to ensure completeness and quality of validation

Lessons Learned
Validation Layers/API Sync

- **If that policy had existed initially…**
  - More timely validation support for application developers
  - An improved Vulkan specification (developing validation is a good review of the spec for accuracy and clarity)

- **Success story - Descriptor Heap**
  - All CPU based validation completed with good quality. 16K LoC!
  - Launched with extension release
  - GPU-AV validation coming in next SDK

Lessons Learned
Validation Layers/Specification VUIDs

- **The initial launch of the specification did NOT have VUIDs (Valid Usage statements)!**
  - We should have had those on "day one"
- **This really made it tricky to generate a good error message**
- **We had no idea how much validation coverage we were providing**
- **And even harder to point to relevant portions of the Vulkan specification!**

On Vulkan's 10th Birthday
Ecosystem Summary

- The Vulkan Ecosystem is not perfect, but it is robust

Thank You! Questions?