Vulkan, Forging Ahead

Ralph Potter, Samsung
Vulkan Working Group Chair

Today’s Presentations

- **Vulkan 1.4**
  - Ralph Potter (Vulkan Working Group Chair / Samsung)
- **Vulkan SDK Update**
  - Spencer Fricke (LunarG)
- **HDR support in Vulkan**
  - Kangying Cai (Huawei)
- **GPU-driven Rendering in Vulkan**
  - Ken Shanyi Zhang (AMD)
- **Slang**
  - Shannon Woods (NVIDIA)
- **Vulkan Safety Critical**
  - Shannon Woods (NVIDIA)
- **Questions & Answers**

KHRONOS GROUP

Vulkan 1.4 Launch

Strengthening the Vulkan Ecosystem

5 December 2024

Vulkan

An explicit API for graphics and compute on GPUs

- Radically cross-platform, from embedded to desktop
- Focus on high performance and user control

Driving the future evolution of graphics hardware

- Setting requirements for new hardware
- Ensuring compatibility with current hardware
- Focus on solving issues raised by industry experts

Developed collaboratively by industry experts

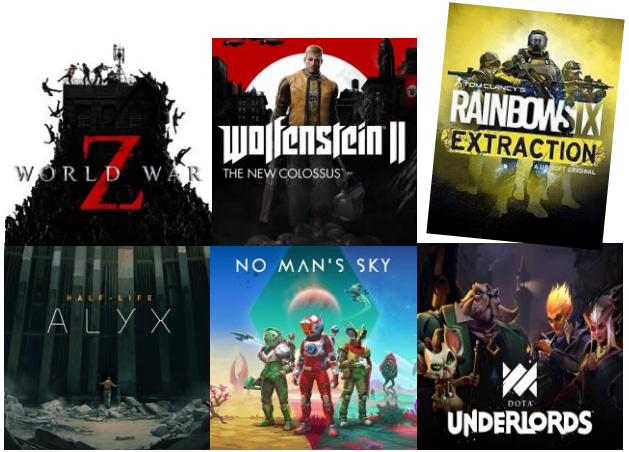
- Input considered from a wide range of sources

Vulkan is Everywhere

Desktop and Mobile GPUs and SOCs

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

http://vulkan.gpuinfo.org/

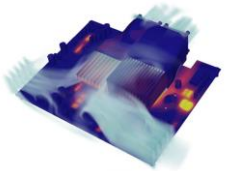


Desktop Games

Mobile Games

AUTODESK® FUSION 360®

Cross-platform post-processing and display of simulation results



Adobe, Substance

Substance 3D Stager
Cross-platform ray tracing

Applications

Engines

CIDER, CINDER, XENKO, GODOT, CRYENGINE, Unity, source, id, UNREAL ENGINE, Croteam Serious Engine, UX3D, NetEase Games

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

Vulkan is Unique

Vulkan is the *only open standard* modern GPU API

Under multi-company governance

Supported by all major GPU vendors

Cross-platform support reduces developer porting effort and costs

Used extensively by games and applications

Windows and Linux
Desktops and Cloud

Mobile

Game Streaming
Platforms

Gaming
Platforms

Apple Platforms
(via translation layer)

Vulkan Roadmap: Setting industry direction

The roadmap sets forward-looking targets for new hardware, with milestones established for new mid-to-high-end GPUs in the year of their release

Roadmap **2022**

Roadmap **2024**

Roadmap **2026**

Core Specifications: Support for Current GPUs

Features that can be supported on current GPUs are brought into core

Maintenance Updates

Vulkan **Core 1.0**

Vulkan **Core 1.1**

Vulkan **Core 1.2**

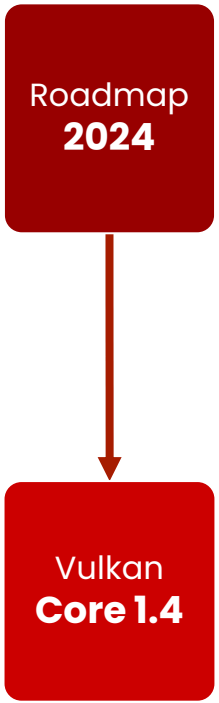
Vulkan **Core 1.3**

Vulkan **Core 1.4**

Vulkan **Core 1.5**

2016 2018 2020 2022 2024

Roadmap Sets Direction, Core Solidifies It



- **Vulkan 1.4 is the first core version derived from the roadmap**
  - Notable benefits both in design and development
  - Enabled huge increase in supported features
- **Most of the tough questions for 1.4 largely already answered**
  - Future direction already set with the roadmap
  - Features already designed, shipped, and implemented
  - Vendors already knew which hardware could support what
- **“Just” had to put the pieces together**
  - Much easier development cycle than previous cores
  - Allowed us to focus on future roadmap items

Vulkan 1.4 Core Specification

Integrates significant requested functionality proven as extensions

Mandated support for new functionality ensures availability on all Vulkan 1.4 implementations

Dynamic rendering local read bringing subpass support to the dynamic rendering API

Streaming transfers via host image copy or mandatory async transfer queue support

Fine-grained control of floating point optimization behavior

Mandating previously optional features such as scalar block layout and 8/16 integer support

Maintenance extensions up to VK_KHR_maintenance6

Several limit increases, including 8K rendering with up to eight separate render targets

And more…

16 extensions in total

Raising the Bar

Vulkan 1.0 was designed to run on GLES 3.1-class GPUs (circa 2014)

Core versions need to run on the broadest set of devices

Vulkan 1.4 raises minimum hardware requirements

Optional functionality and artificially low limits increase complexity for developers

Makes 28 previously optional features mandatory, including scalar block layout and 8/16 bit integer support in shaders

Raises minimum limits on 31 properties

Provides reliable access to functionality across all supported platforms

Streaming Transfers

Streaming image resources without interrupting rendering

Previously required copies on GPU timeline

VK_EXT_host_image_copy (optionally) promoted to core

Enables CPU-side image copies

If host copy is not supported, then a dedicated asynchronous transfer queue is mandatory

https://www.khronos.org/blog/copying-images-on-the-host-in-vulkan

Dynamic Rendering Local Read

Vulkan 1.3 promoted dynamic rendering to core…

- Removed the need for render pass and framebuffer objects
- Greatly simplified the programming model

…but the original extension didn't address input attachments or subpasses

- Critical for performance on tile-based GPUs

Vulkan 1.4 includes local reads for color attachments/storage resources

- Closes the gap versus legacy render passes
- Local reads for depth/stencil/multisampled attachments are optional

https://www.khronos.org/blog/streamlining-subpasses

Vulkan 1.4 Release Schedule

- **Specification - Available Now**
  - https://docs.vulkan.org
- **API Headers - Available Now**
  - https://github.com/KhronosGroup/Vulkan-Headers
- **Conformance Tests - Available Now**
  - https://github.com/KhronosGroup/VK-GL-CTS
- **Vulkan Loader - Available Now**
  - https://github.com/KhronosGroup/Vulkan-Loader
- **Vulkan Validation Layers - MR in review**
  - https://github.com/KhronosGroup/Vulkan-ValidationLayers/pull/8955
- **Complete Vulkan SDK release - January 2025**
  - https://www.lunarg.com/vulkan-sdk
- **Implementations**
  - AMD, Arm, Imagination, Intel, NVIDIA, Qualcomm, and Samsung passing conformance today
  - NVIDIA and Mesa drivers publicly available today
  - Other vendors coming soon

Vulkanised 2025

Vulkanised 2025

The 7th Vulkan Conference | Cambridge, UK | Feb 11-13, 2025

The Premier Vulkan Developer Conference

To be hosted by Arm in Cambridge, UK

Registration and program: https://vulkan.org/events/vulkanised-2025

More Information

- **Vulkan: https://www.vulkan.org/**
  - Press Release: https://khr.io/vulkan14
  - Final specification: https://docs.vulkan.org/spec/latest
  - Spec GitHub Repo: https://github.com/KhronosGroup/Vulkan-Docs/
  - Discord Link for community discussion: https://discord.com/invite/vulkan

Thank You!

Vulkan Working Group, Khronos F2F Meeting, Seattle, September 2024

Vulkan SDK Update

Spencer Fricke, LunarG

日本語版スライド (Japanese version of slides)

このプレゼンテーションをフォローするために

About Me

- Been at LunarG for 2+ years
- Vulkan Validation Layer tech lead
- Previously lived in Japan for 2 years
  - Now located permanently back in the USA
- Active member of Vulkan Working Group
- Help maintain various other parts of ecosystem
  - Vulkan-Guide
  - SPIRV-Visualizer
  - SPIRV-Guide
  - SPIRV-Reflect

What drives the Vulkan ecosystem?

What drives the Vulkan ecosystem?

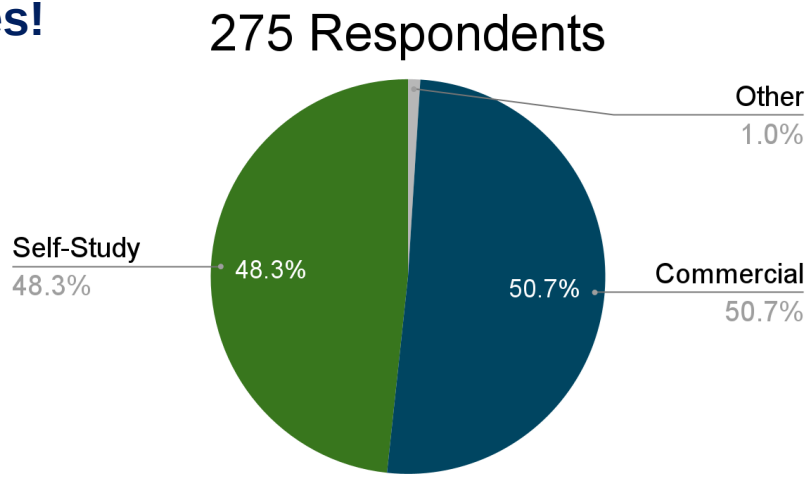
All of you!

Vulkan Developer Ecosystem Survey

Helps drives LunarG Ecosystem Priorities!

What we heard -

- Shader tool chain - needs dev & maintenance
  - 60%+ glsl -> SPIR-V
  - 20% use DXC
- Validation Layers
  - Increase coverage
  - Readability / interpretation of error messages
  - Improve performance
- MoltenVK
  - Move forward faster
- Difficult Tasks
  - Identifying driver defects
  - Debugging layer issues
  - Debugging install & configuration issues



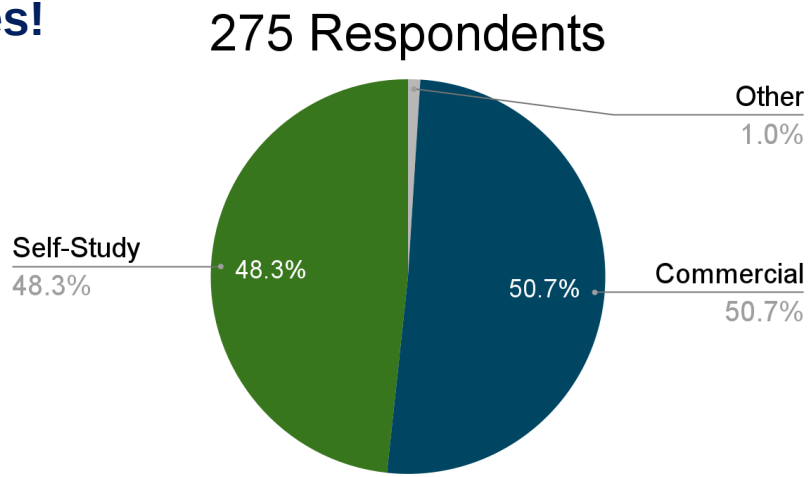
LUNARG

Vulkan Developer Ecosystem Survey

Helps drives LunarG Ecosystem Priorities!

What we heard -

- Shader tool chain - needs dev & maintenance
  - 60%+ glsl -> SPIR-V
  - 20% use DXC
- Validation Layers
  - Increase coverage
  - Readability / interpretation of error messages
  - Improve performance
- MoltenVK
  - Move forward faster
- Difficult Tasks
  - Identifying driver defects
  - Debugging layer issues
  - Debugging install & configuration issues



2025 Ecosystem Survey Coming in February!

LUNARG

Windows

Windows On arm

The Vulkan SDK

Vulkan Loader	vkconfig	Validation Layer	vulkaninfo	
SPIR-V Optimizer	SPIR-V Tools	slang	Crash Diagnostic Layer	Emulation Layers
shaderc	SPIR-V Validator	Profiles Toolset	GPUInfo	VOLK
DXC	SPIR-V Reflect	apidump	GFX Reconstruct	VKVIA
SPIR-V Cross	glslang	Vulkan-HPP	Screenshot	VMA
MoltenVK	SPIR-V Visualizer	SDL	Monitor	GLM

Benefits

- Pre-built
- Curated
- Integrated
- System Installation
- vkconfig ready for use
- License Registry

Delivered by LunarG in close coordination with the Khronos Vulkan working group

SDK Enhancements in 2024

2024

- Addition of Slang compiler
- Crash Diagnostic Layer
- Windows 11 on ARM Vulkan SDK
- Synchronization validation for Timeline Semaphore
- iOS support added to macOS Vulkan SDK
- Profiles enhancements
- VP_KHR_roadmap_2024
- VK_EXT_layer_settings API

SDK Enhancements in 2024

2024

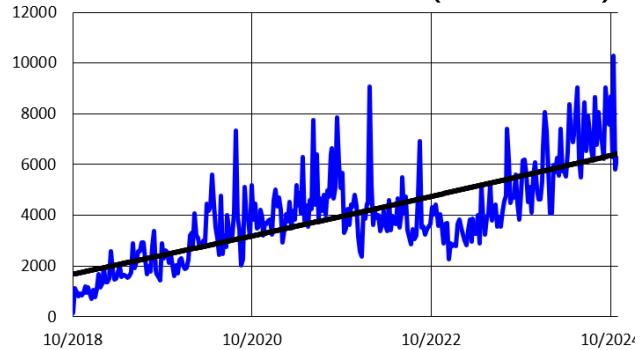
- Addition of Slang compiler
- Crash Diagnostic Layer
- Windows 11 on ARM Vulkan SDK
- Synchronization validation for Timeline Semaphore
- iOS support added to macOS Vulkan SDK
- Profiles enhancements
- VP_KHR_roadmap_2024
- VK_EXT_layer_settings API

Coming in January

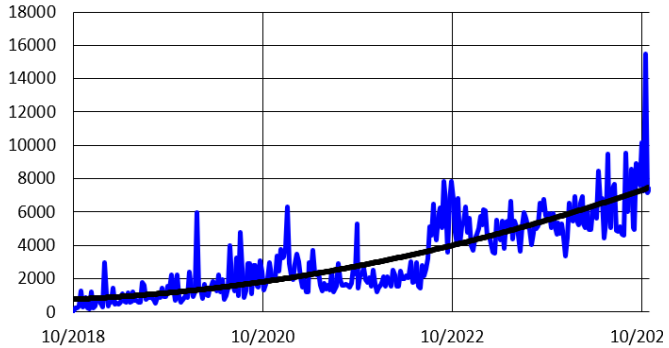
- Vulkan 1.4 support
- vkconfig3
- Automatic update of the Vulkan Runtime (Loader) with the Windows Vulkan SDK installation

SDK Downloads

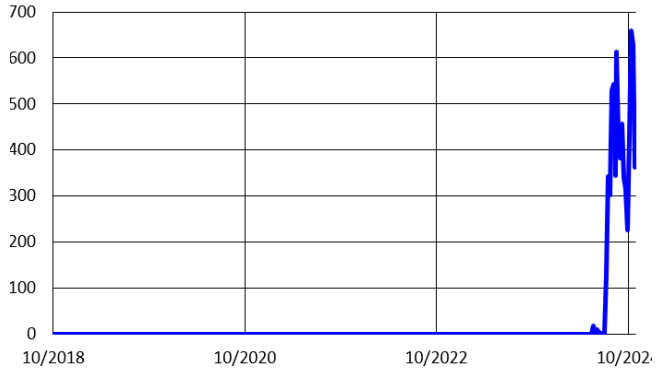
Linux SDK (~6000/wk)



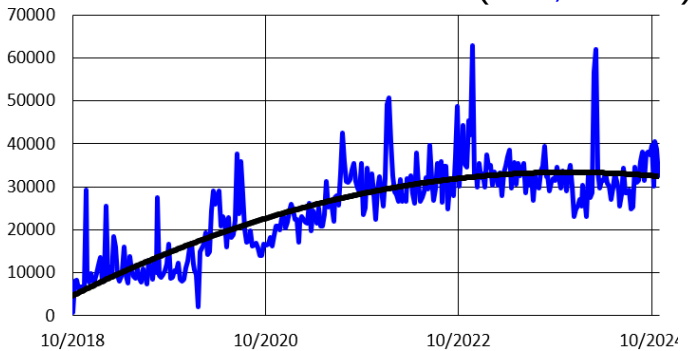
Mac SDK (~6000/wk)



Windows 11 on ARM SDK



Windows x86 SDK (~31,000/wk)



Trends -

- Win x86 - Plateau
- Linux - Slowing
- Mac - Growing
- Win ARM - Too early

★ Polynomial trendlines due to large fluctuations

LUNARG

Updates for some of the Vulkan SDK components

GFXReconstruct

Key Results

- Can now get draw resources for showing intermediate results
- Can now share initialization data among multiple captures in one session to reduce file sizes
- Experimental OpenXR branch

Needs Work

- Capture overhead improvement
- Android functionality and stability
- Ray-tracing support

Validation Layers

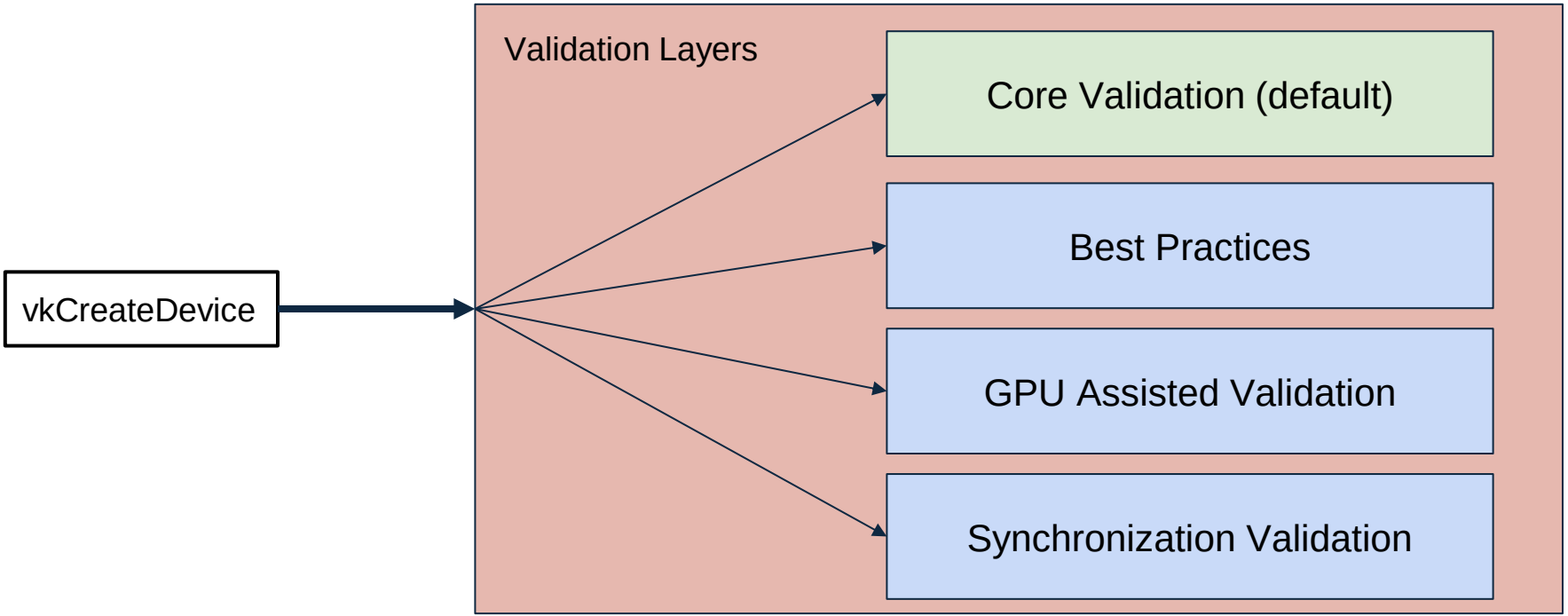
Key Results

- Improved error messages
- More extensions added
- Many issues closed
- **Zero crashing focus**
- **Zero False Positives focused**

Needs Work

- Many Open Issues
- Always new extensions
- Continue performance tuning
- Best practices
  - Lack dedicated engineer
- Error reporting
  - Consistency, completeness, format
  - **Please raise an issue if you find a confusing error message!**

Validation Layers



LUNARG

GPU-AV (GPU Assisted Validation)

Key Results

- Dedicated engineer!
- Performance improvements
- More accurate results
- New functionality added
- Better error messages

Needs Work

- Still can be painfully slow
- Majority of missing validation checks require GPU-AV
- Errors can still be hard to know where they came from

Synchronization Validation

Key Results

- Timeline Semaphore support (added in 1.3.296)
- Many Github issues closed!
- Improved error reporting in the January 1.4 SDK

Needs Work

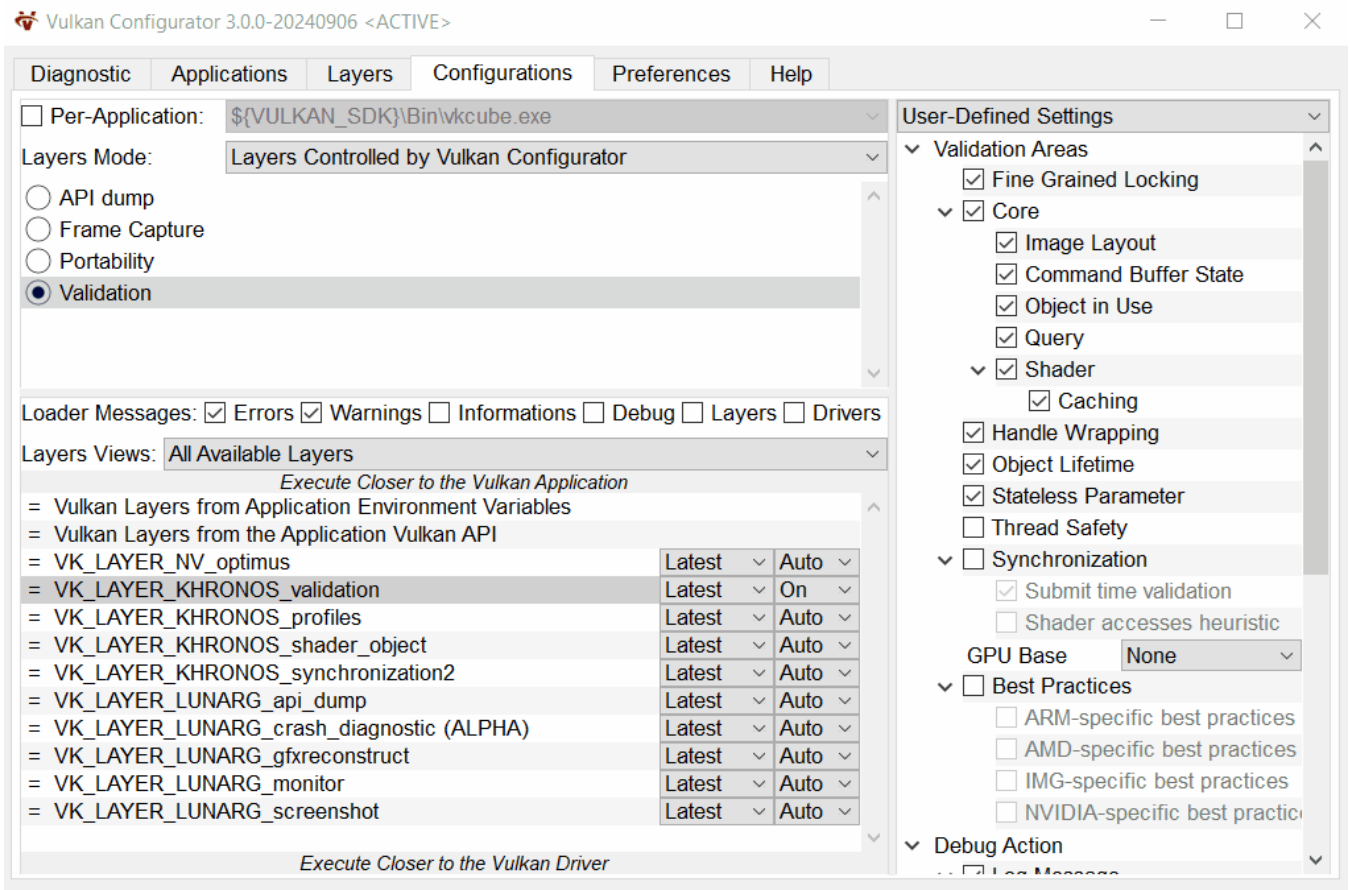
- Performance tuning
- Continue improving error reporting
- Support for memory aliasing
- Better testing
- **GPU-AV integration to track accesses inside shaders**

[Generated from ChatGPT]

Vulkan Configurator 3 (vkconfig3)

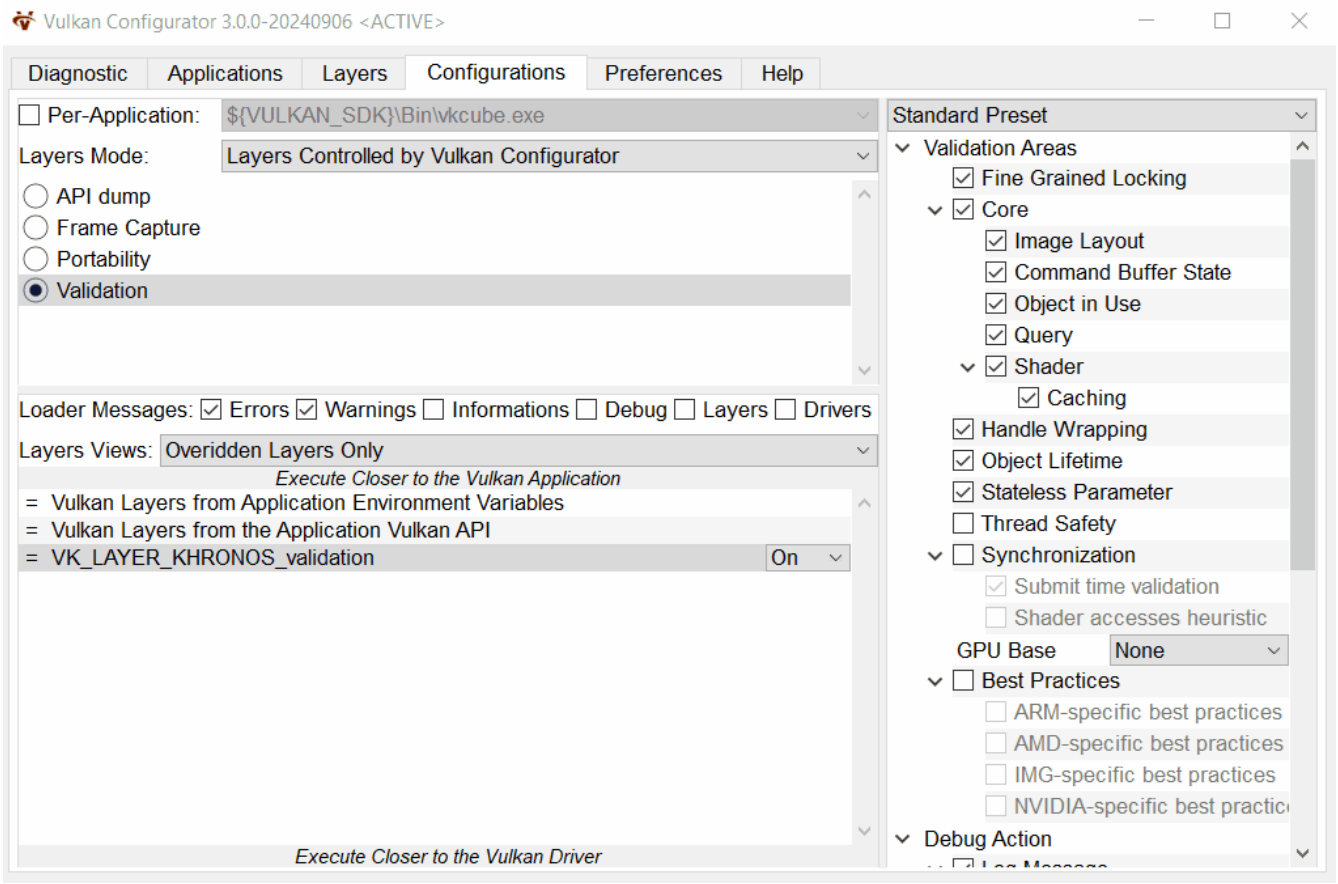
- **Vulkan Configurator 2** - in maintenance mode since May 2024 SDK
- **Vulkan Configurator 3** - Targeting January 2025 SDK
  - Implementing Vulkan Loader settings file:
    - Vulkan developer control of all layers using the UI
    - Vulkan Loader logging using the UI
    - Per-executable layer configurations
    - Easier to select a specific layer version
  - Redesigning UI using a tab per use case
    - Improve application launcher with environment variables and multiple options
    - Add a diagnostic tab (checking Vulkan installation)
  - Adding a layer setting type to run and control command lines

Vulkan Configurator 3 - UI Example 1



LUNARG

Vulkan Configurator 3 - UI Example 2

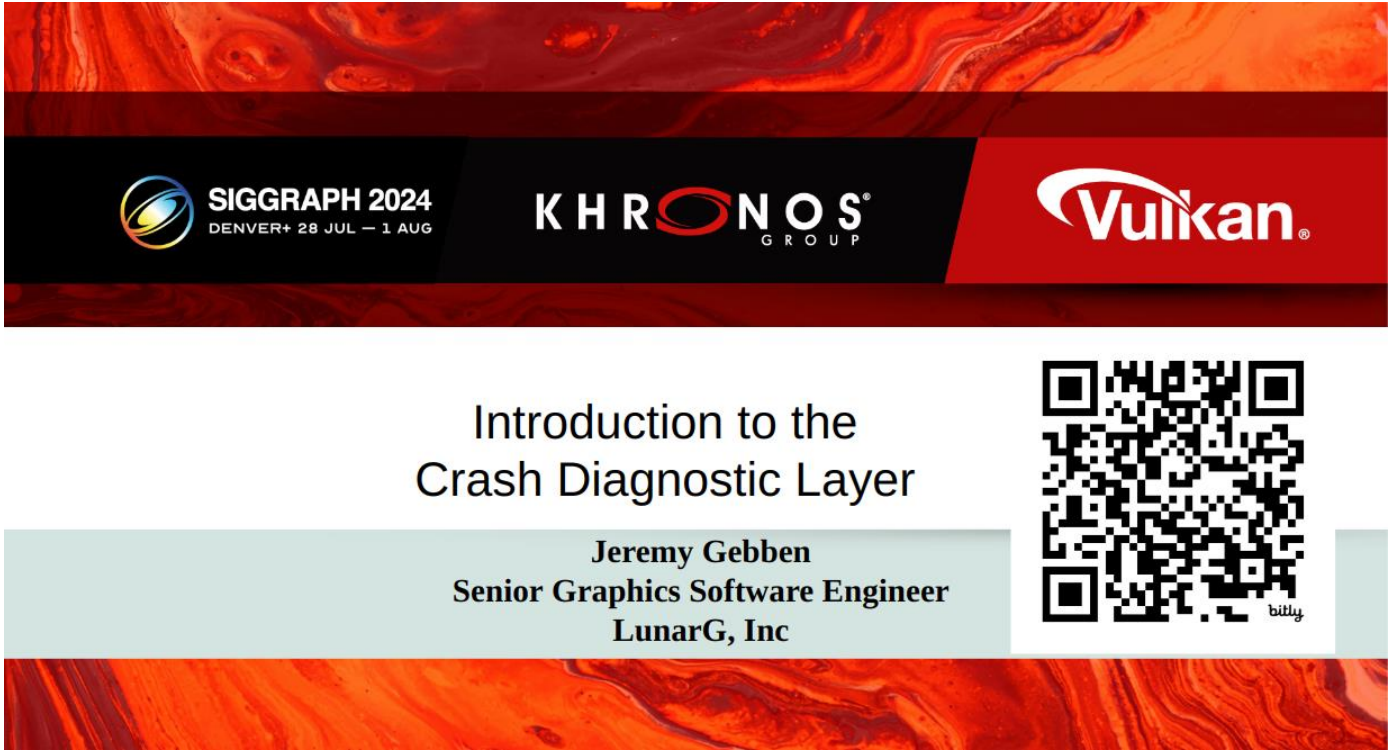


Crash Diagnostic Layer

- Track down and identify the cause of GPU hangs and crashes
  - aka VK_ERROR_DEVICE_LOST
- Instruments command buffers with completion checkpoints
- Generates a dump file
- Strong user demand
  - Debugging Device Lost errors very difficult!
- Still in early development
  - **We would love to get your feedback!**

For more information

https://www.youtube.com/watch?v=h5Ty-o8_pWE



LUNARG

Vulkan Profiles - Use Cases

- **Roadmap profiles**: express guidance on the future direction of Vulkan devices.
  - Eg: Khronos Roadmap 2024
- **Platform profiles**: express the Vulkan support available on a platform.
  - Eg: Android Baseline 2021
- **Engine profiles**: express some rendering code paths requirements of an engine.
  - Eg: VP_UE_Vulkan_SM6_RT in Unreal Engine.
- **Device profiles**: express the Vulkan support of a single Vulkan driver for a Vulkan device.
  - Eg: GPUinfo.org reports
- **Architecture profiles**: express the Vulkan support of a class of GPUs.
  - Eg: D3D12 Feature Level 12.1

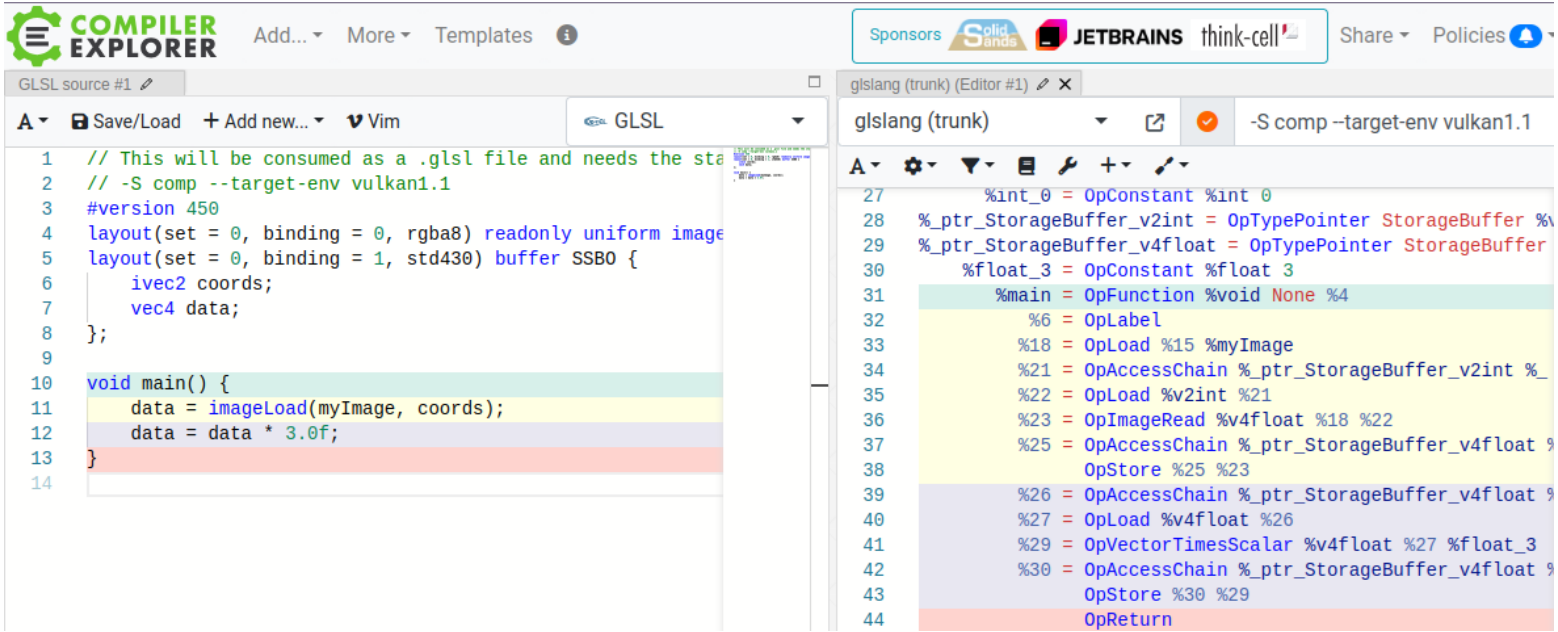
Vulkan Profiles - Tool Set for Developers

- For more information about
  - How to create a Vulkan Profile
  - Vulkan Profiles Layer
  - Vulkan Profiles API Library

"Better Vulkan Application Deployment thanks to Vulkan Profiles"

glslang

- Can now use GLSL, HLSL, and SPIR-V on Compiler Explorer (godbolt)
- https://www.lunarg.com/lunarg-3d-graphics-engineer-adds-glsl-spir-v-as-inputs-to-compiler-explorer/



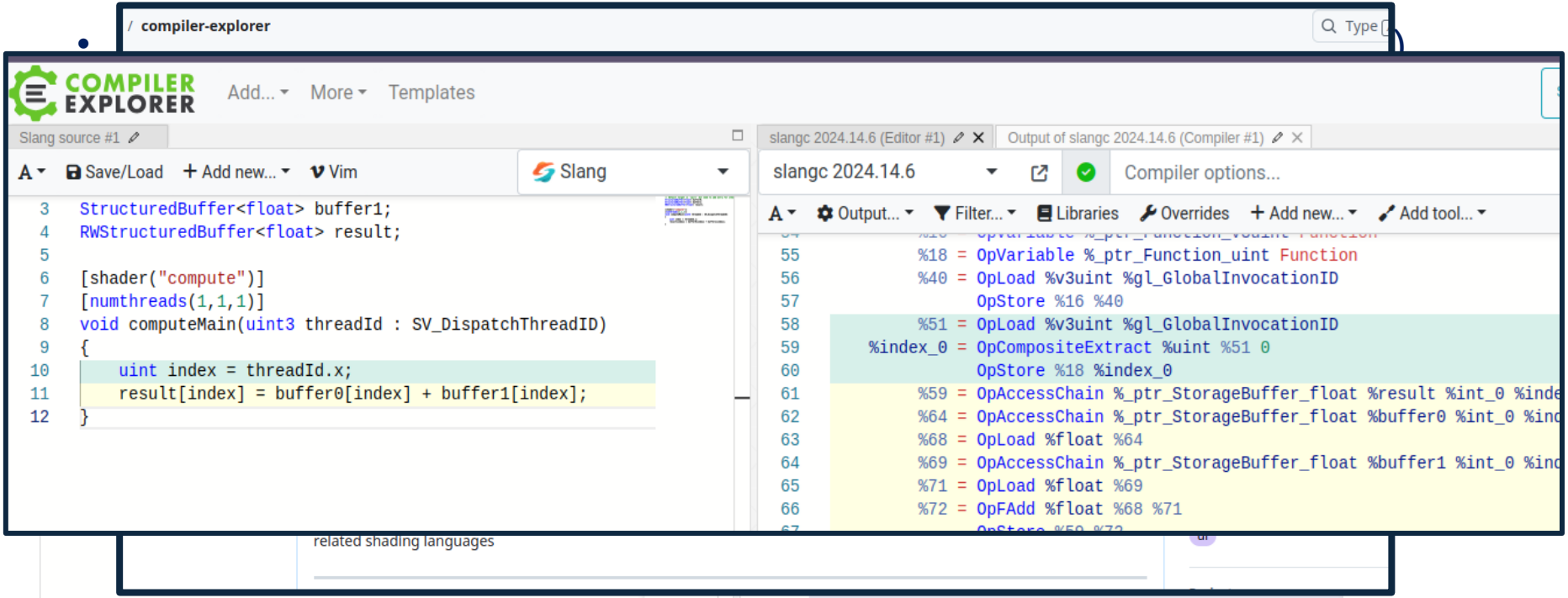
LUNARG

glslang

BREAKING NEWS!

glslang

MORE BREAKING NEWS!



LUNARG

We need your help!

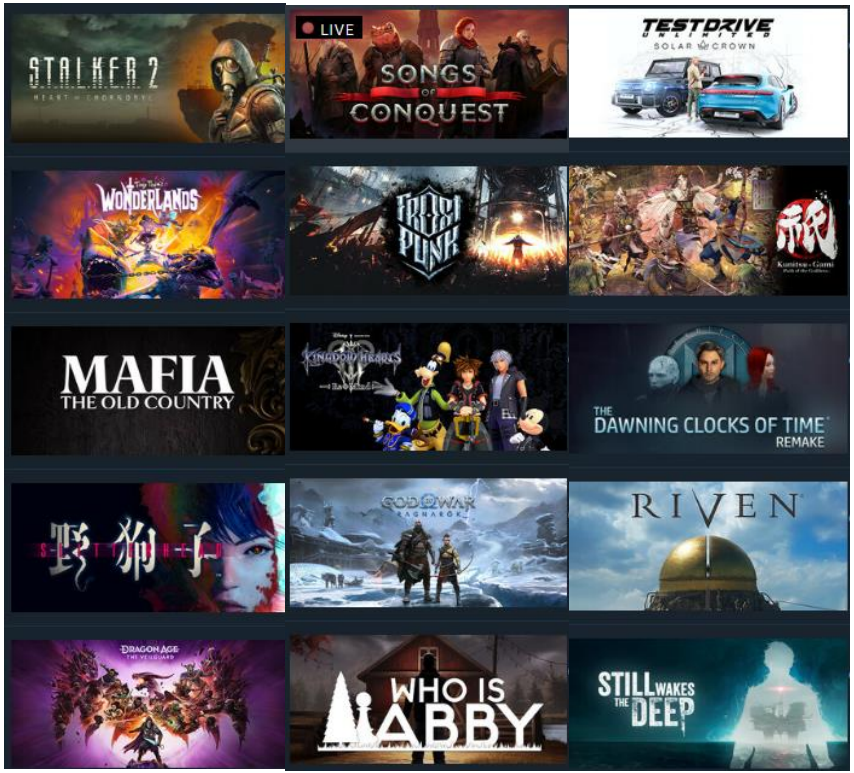

LUNARG®

HDR support in Vulkan

Zehui LIN, Kangying CAI, Pan GAO, Xiufeng ZHANG Huawei

More and More HDR in Industry

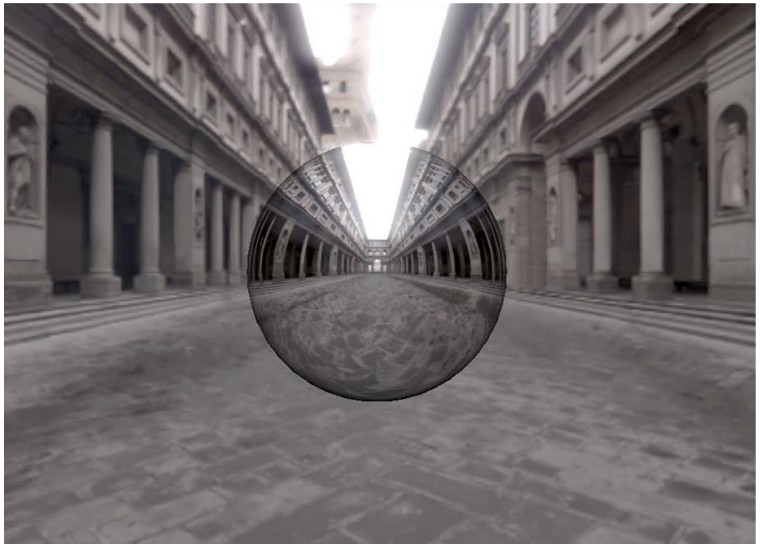
- Most high-end mobiles can shoot HDR images.
- Most high-end laptops, TVs and mobile phones have over 1000 nit HDR displays.
- Many over-the-top platforms have provided HDR videos.
- HDR Vivid (a standard by UWA) has over 50,000 hours of video content.
- More and more games have native HDR support.



A list of HDR games on Steam

From HDR Rendering to HDR Display

- Comparison between sRGB and HDR
- It's time to get the best visual effect out of your HDR display!

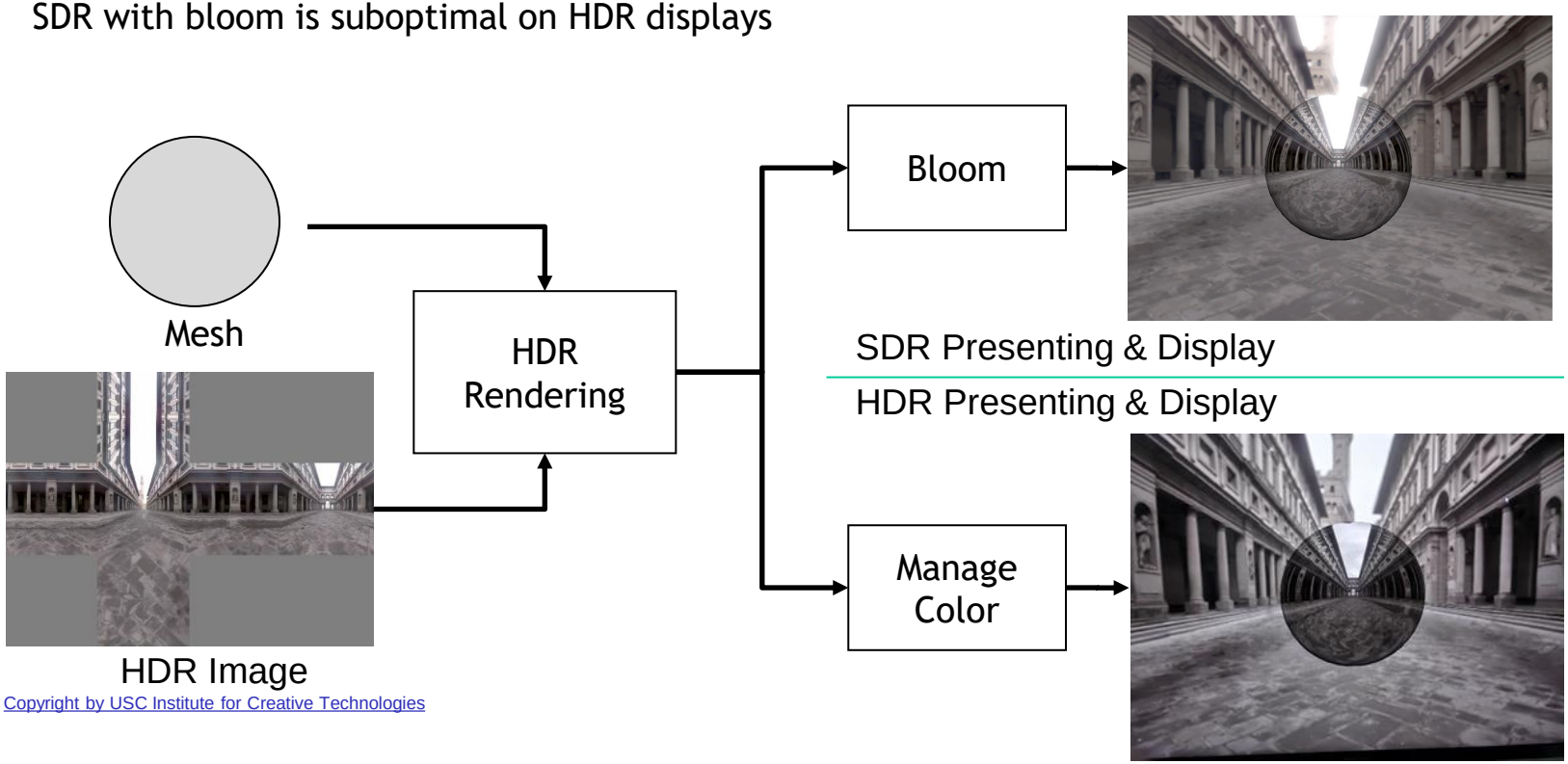


sRGB Output

HDR Output (Simulated)

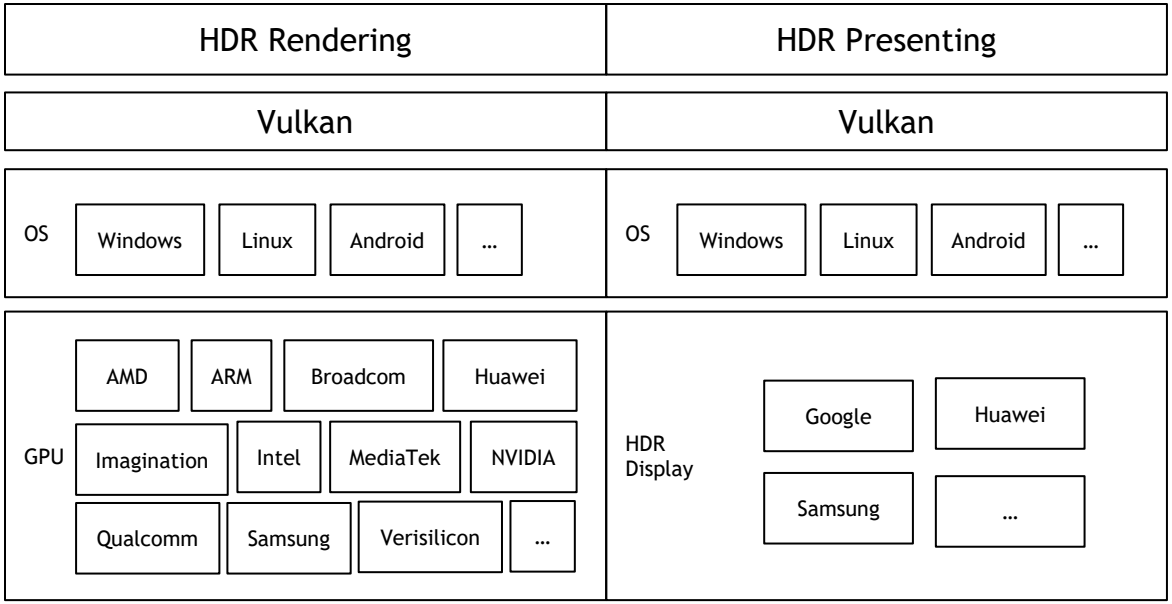
From HDR Rendering to HDR Display

- Using an HDR environment map can produce a sRGB image (with bloom effect)
- SDR with bloom is suboptimal on HDR displays



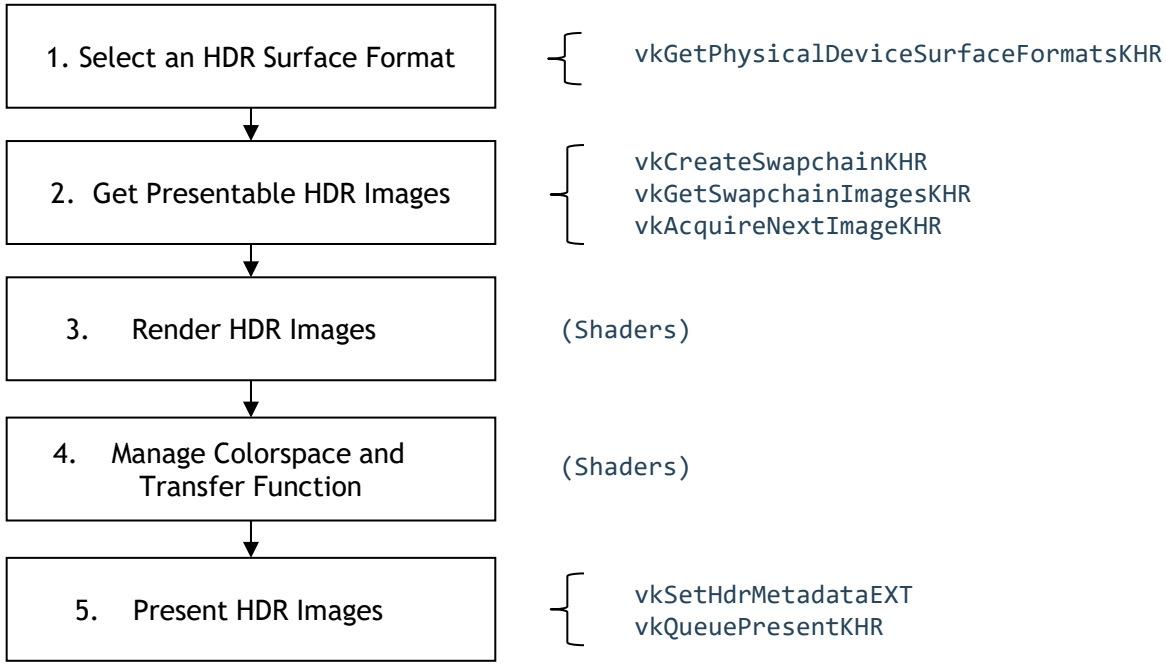
Vulkan – Cross Platform API for HDR

- HDR Rendering and HDR Presenting on Vulkan
- Seamless connection between GPUs and displays



Steps to display HDR content in Vulkan

- Steps and related commands



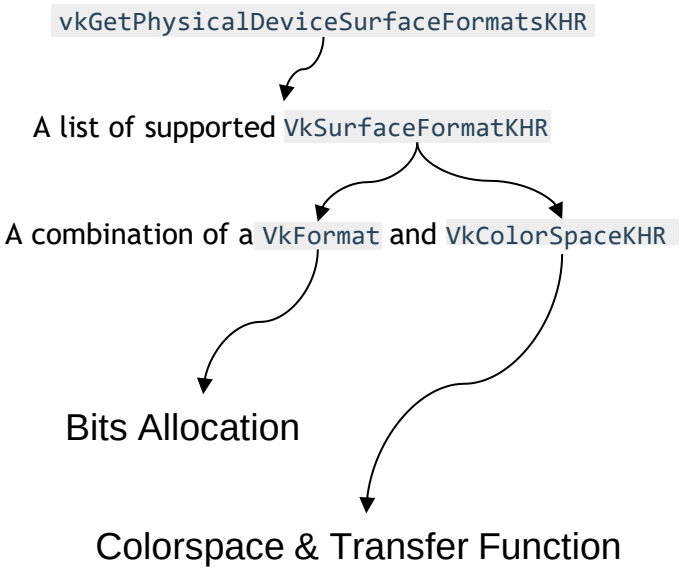
Step 1. Select an HDR Surface Format

```
uint32_t surface_format_count{0U};
// Get the number of supported formats
vkGetPhysicalDeviceSurfaceFormatsKHR(
    device,
    surface,
    &surface_format_count,
    nullptr);

// Get the supported format list
surface_formats.resize(surface_format_count);
vkGetPhysicalDeviceSurfaceFormatsKHR(
    device,
    surface,
    &surface_format_count,
    surface_formats.data());

auto surface_format = select_surface_format(surface_formats);
```

```
VkSurfaceFormatKHR select_surface_format(
    const std::vector< VkSurfaceFormatKHR> &surface_formats) {
    for (auto surface_format : surface_formats) {
        if (surface_format.colorSpace == VK_COLOR_SPACE_HDR10_ST2084_EXT &&
                surface_format.format == VK_FORMAT_A2R10G10B10_UNORM_PACK32)
            return surface_format;
    }
    // fall back to sRGB
    …
}
```



Step 2. Get Presentable HDR Images

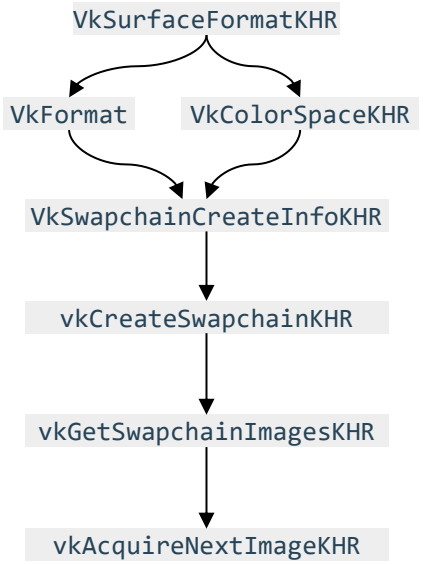
```
// Set the desired format and colorspace
VkSwapchainCreateInfoKHR create_info;
create_info.imageFormat      = surface_format.format;
create_info.imageColorSpace = surface_format.colorSpace;
… // Other configs

// Create a swapchain with the format and colorspace
VkSwapchainKHR swapchain;
vkCreateSwapchainKHR(device, &create_info, nullptr, &swapchain);

// Get swapchain images
uint32_t image_available{0u};
vkGetSwapchainImagesKHR(device, swapchain, &image_available, nullptr);
images.resize(image_available);
vkGetSwapchainImagesKHR(device, swapchain, &image_available,
images.data());

// Acquire a presentable image for rendering from swapchain
uint32_t image_index;
vkAcquireNextImageKHR(
    device,
    swapchain,
    timeout,
    semaphore,
    fence,
    &image_index);
```

- Setup swapchain and images with selected surface format



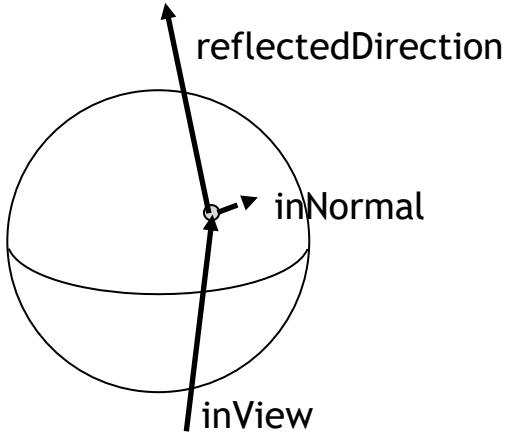
Step 3. Render HDR Images

```
// vertex shader
…
layout (location = 0) out vec3 outNormal;
layout (location = 1) out vec3 outView;
…
```

```
// fragment shader

layout (location = 0) in vec3 inNormal;
layout (location = 1) in vec3 inView;
…
layout (binding = 0) uniform samplerCube HdrEnvMap;
…

void main() {
    …
    // Sample the HDR environment map
    vec4 color = texture(HdrEnvMap, reflect(-inView, inNormal));
    …
}
```



- In some applications, HDR image comes from HDR video decoding

Step 4. Manage Colorspace and Transfer Function

```
// fragment shader
layout (constant_id = 0) const int colorSpace = 0;
layout (constant_id = 1) const int transferFunction =
0;
...

void main()
{
    vec4 color = texture(samplerColor, inUV);

    // convert to BT.2020 if necessary
    if (colorspace == BT2020)
        color.xyz = sRGBToBT2020(color.xyz);

    // encode according to ST2084 or HLG if necessary
    if (transferFunction == ST2084) {
        color.xyz = linearToST2084(color.xyz);
    } else if (transferFunction == HLG) {
        color.xyz = linearToHLG(color.xyz);
    }

    outColor = color;
}
```

VkColorSpaceKHR	Colorspace	Transfer Function
VK_COLOR_SPACE_HDR10_ST2084_EXT	BT.2020	ST2084
VK_COLOR_SPACE_HDR10_HLG_EXT	BT.2020	HLG
VK_COLOR_SPACE_EXTENDED_SRGB_LINEAR_EXT	sRGB	linear

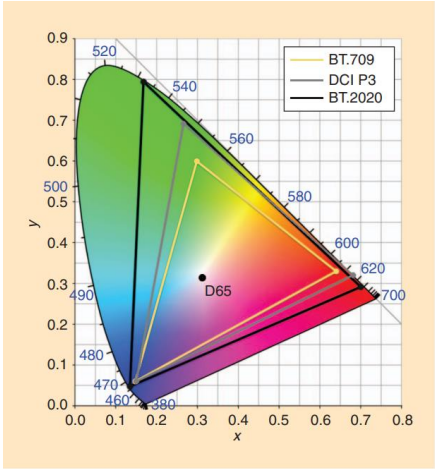
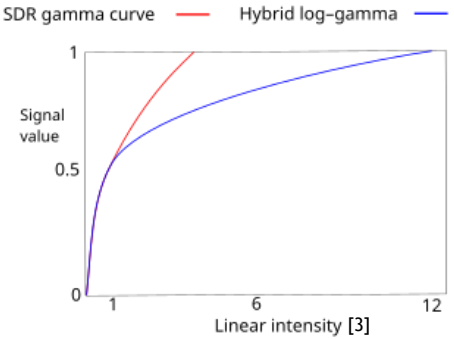


FIGURE 14. The CIE 1931 *xy* chromaticity diagram with the BT.709 [21], DCI P3 [22], and BT.2020 [23] color gamut. [2]

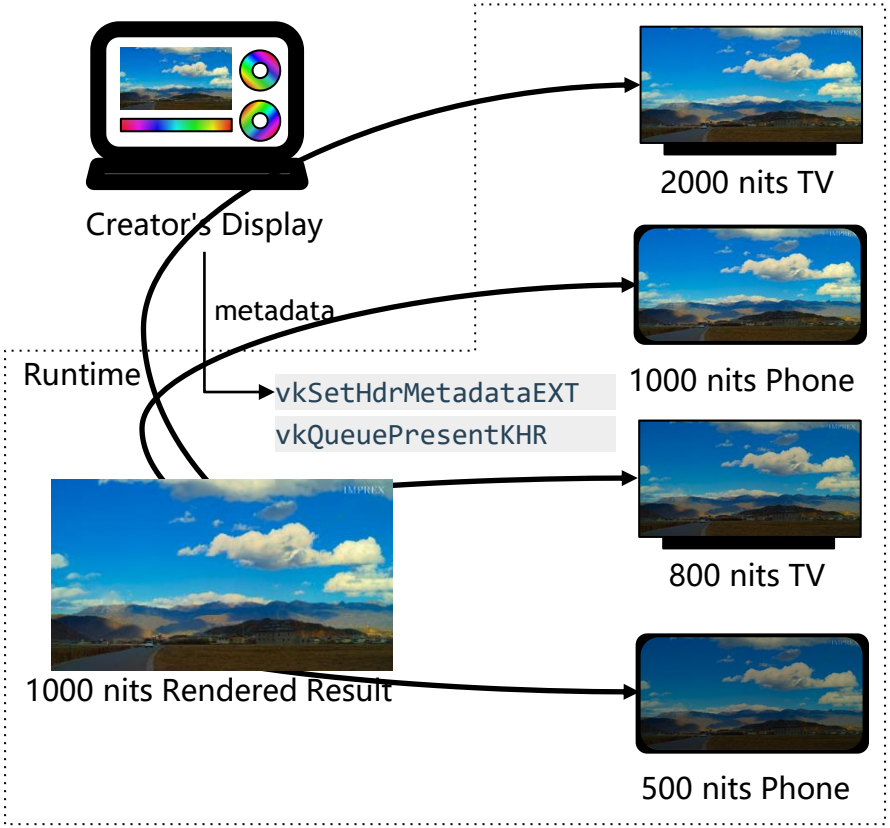


Step 5. Present HDR Images

```
// Set metadata as a hint to the display
vkSetHdrMetadataEXT(
    device,
    1, // swapchain count
    &swapchain,
    &metadata);

// Fill in present info
VkPresentInfoKHR present_info = {
    VK_STRUCTURE_TYPE_PRESENT_INFO_KHR
};
present_info.pNext              = NULL;
present_info.swapchainCount     = 1;
present_info.pSwapchains        = &swapchain;
present_info.pImageIndices      = &image_index;
… // Other settings

// Send the image to present
vkQueuePresentKHR(queue, &present_info);
```



HDR Display related extensions

- HDR texture compression

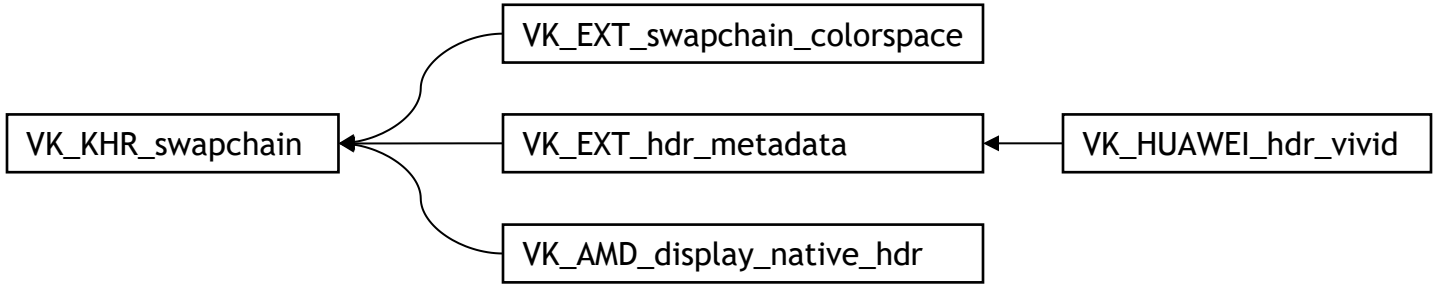
VK_EXT_astc_decode_mode

VK_EXT_texture_compression_astc_hdr

……

glTF HDR image compression

- HDR Display



Reference

1. “High-Resolution Light Probe Image Gallery.” USC Institute for Creative Technologies, https://vgl.ict.usc.edu/Data/HighResProbes/ . Accessed 20 Nov 2024.
2. Boitard, Ronan et al. “Demystifying High-Dynamic-Range Technology: A new evolution in digital media.” *IEEE Consumer Electronics Magazine* 4 (2015): 72-86.
3. “Hybrid log–gamma.” Wikipedia, https://en.wikipedia.org/wiki/Hybrid_log%E2%80%93gamma . Accessed 20 Nov 2024.
4. Khronos Group. “Vulkan-Samples”, GitHub repository, https://github.com/KhronosGroup/Vulkan-Samples . Accessed 20 Nov 2024.
5. HDR Games. https://store.steampowered.com/curator/33286359-HDR-Games/. Accessed 20 Nov 2024.

GPU-driven Rendering in Vulkan

Ken Shanyi Zhang, AMD

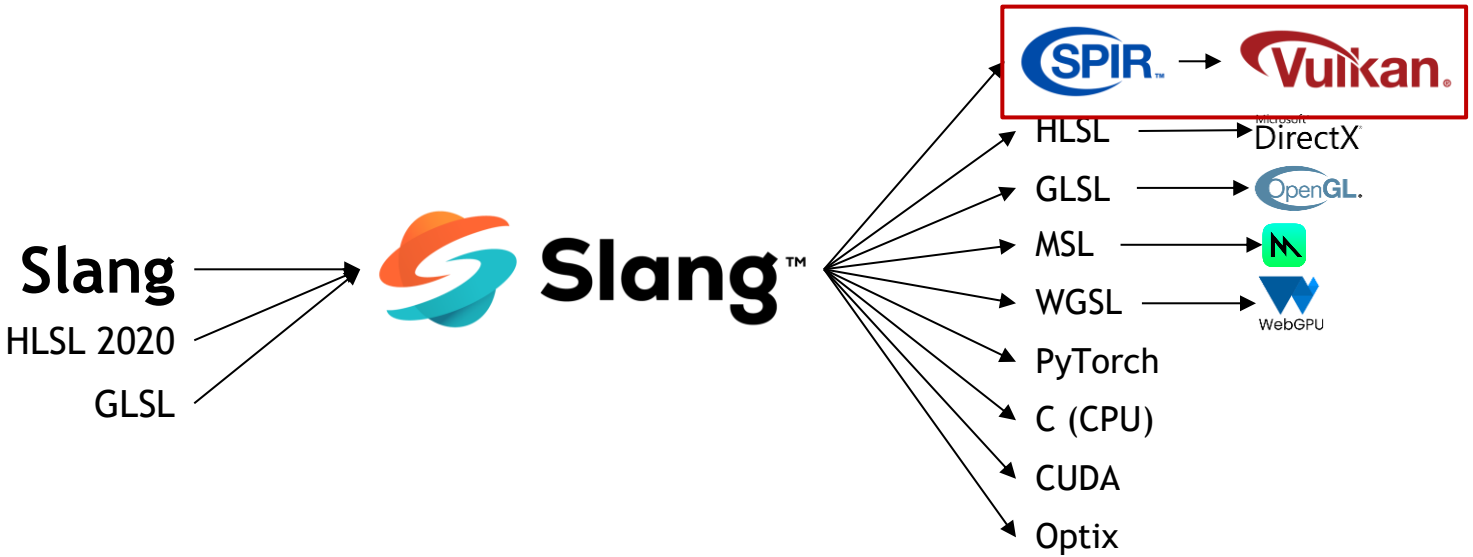
KHRONOS GROUP

Vulkan

Slang

Shannon Woods, NVIDIA

Open-Source, Cross-Platform Compiler



Why Another Shading Language?

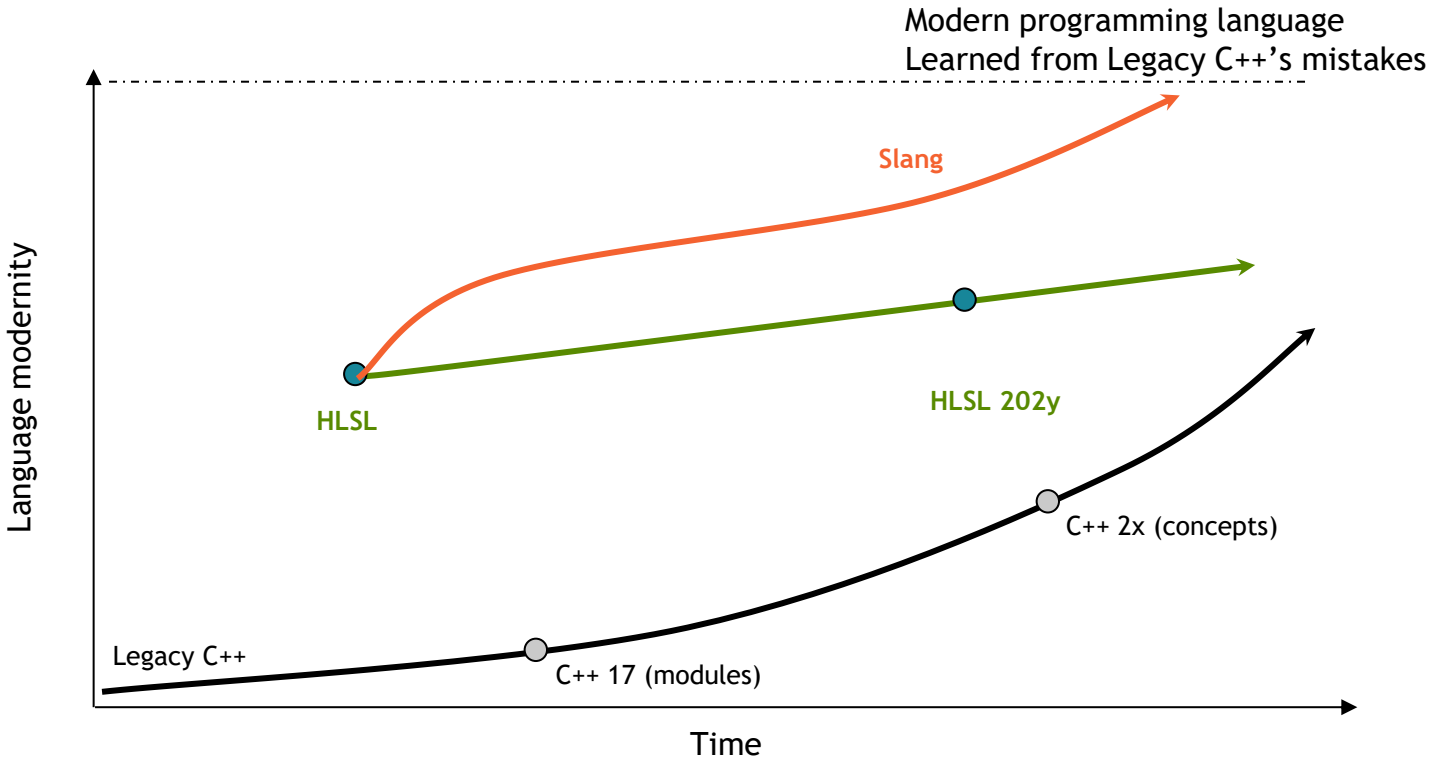
	GLSL	MSL	WGSL	HLSL	Slang
Actively Evolving	NO	YES	YES	YES	YES
Modular Code Management	NO	NO	NO	NO	YES
Converging with C++	NO	YES	NO	YES	NO*
Auto-diff / Neural Shading	NO	NO	NO	NO	YES
Diverse Backend Targets	NO	NO	NO	DXIL and SPIR-V	YES
Open-Source Compiler(s)	YES	NO	YES	YES	YES
Open Governance	YES	NO	YES	NO	YES

*** Slang and HLSL are taking complementary evolutionary paths**

HLSL will remain and evolve as a critically important shading language for many developers

Language diversity and choice is good for the graphics ecosystem!

Language Evolution

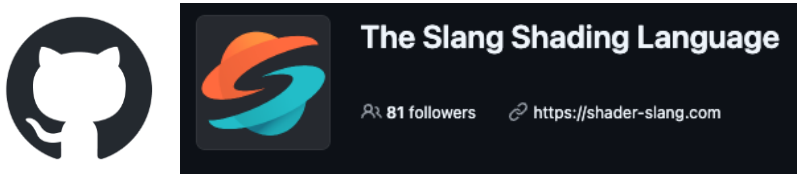


Developers Win with Slang

- Shading language diversity means more competition & innovation
- No single company controls the language, so it can evolve as developers need

For developers, by developers

- Community structure built from OSS best practices
- **Any company or individual** is welcome to become a contributor, not just Khronos members
- Decision-making and development in the open – you can join technical conversations today on Discord, or propose features directly to the repository.
- Slang developers make the decisions about what goes into the language, and you can become one



Slang Tooling

You can already use Slang with existing toolchains

- Step-through debugging in Renderdoc
- Shader inspection in Nsight
- Slang in Vulkan SDK 1.3.296.0 and above

Step-through debugging in RenderDoc

Amazing autocomplete support

- Extensions for Visual Studio & VSCode provide IntelliSense support
- Language server module available for integration into other IDEs
- No other shading language offers something this cool

IntelliSense / Language Server support in action

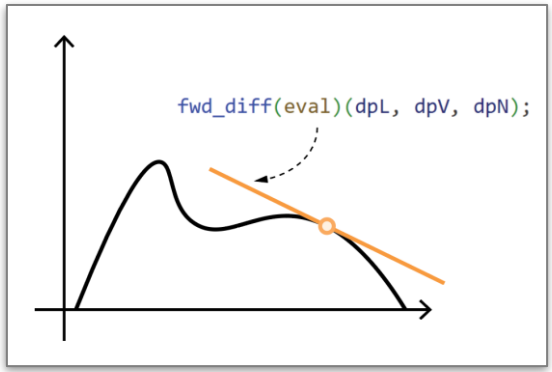
Easily write & maintain differentiable code

- **Differentiable functions power gradient descent solution approaches**
  - Slang brings automatic differentiation to languages optimized for GPU usage
  - Developers can optionally provide custom derivatives for just the portions of a shader where it's necessary – flexibility & control
  - Autodiff support includes arbitrary control flow & dynamic dispatch

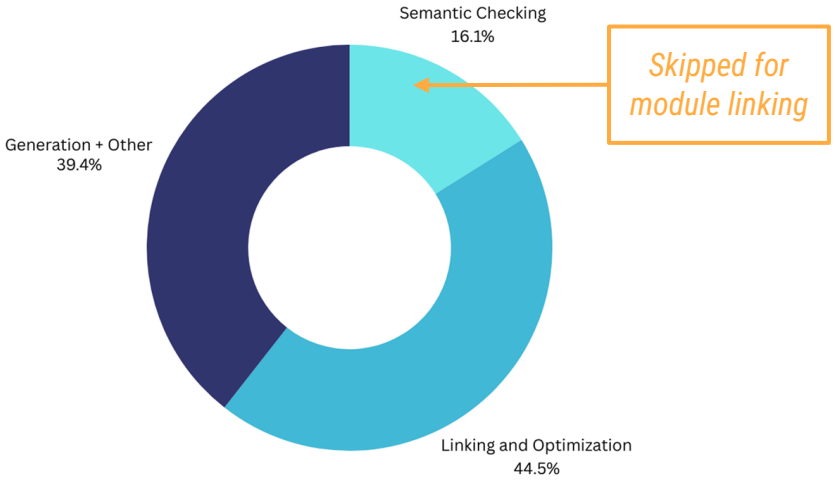
```
interface IBRDF : IDifferentiable
{
    [Differentiable] float3 eval(float3 L, float3 V, float3 N);
}

struct GGXBRDF : IBRDF
{
    float3 baseColor;
    float roughness;
    float metallic;
    float specular;

    [Differentiable] float3 eval(float3 L, float3 V, float3 N)
    {
        float NdotL = dot(N, L);
        float NdotV = dot(N, V);
        if (NdotL < 0 || NdotV < 0)
```



Modules + Interfaces + Generics = Faster Compiles



- **Compilation time**
  - Within 10% of glslc/dxc compilation times with monolithic code
  - Modularized code can reduce time spent in front-end compilation

Modules

Provide separation of compilation and control over visibility

material.slang

material.slang > Material > somePrivateMethod

```
1   module material;
2
3   public struct Material
4   {
5       public float4 evalBRDF(float3 wi, float3 wo)
6       {
7           // ...
8       }
9       internal float4 somePrivateMethod()
10      {
11          // ...
12      }
13  }
14
```

scene.slang 1

scene.slang > Scene > compute

```
1   module scene;
2
3   import material;
4
5   struct Scene
6   {
7       StructuredBuffer<Material> materials;
8
9       void compute(float3 wi, float3 wo)
10      {
11          float4 result = materials[0].evalBRDF(wi, wo);
12          materials[0].somePrivateMethod();
13      }
14  }
15
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

scene.slang 1

'somePrivateMethod' is not accessible from the current context. (30600) [Ln 12, Col 22]

Generics & Interfaces

Generics improve code maintainability

Allows Intellisense to provide accurate assistance

Faster front-end compilation time from reusing type checking results

Interfaces make requirements explicit

Similar to Rust traits, Swift protocols, Haskell typeclasses …

```
1  interface IMaterial
2  {
3      associatedtype BRDF : IBRDF;
4      BRDF sampleAt(SurfacePoint p);
5  }
6
7  interface IBRDF
8  {
9      float4 evaluate(float3 lightDir, float3 eyeDir);
10 }
11
12 interface IGeometry { /*...*/ }
13 interface ILighting { /*...*/ }
```

Switching to Slang isn't Hard!

- Valve migrated entire Source 2 HLSL codebase
- Slang in use in production
- Minimal changes (~10 lines) needed to compile existing shaders with Slang

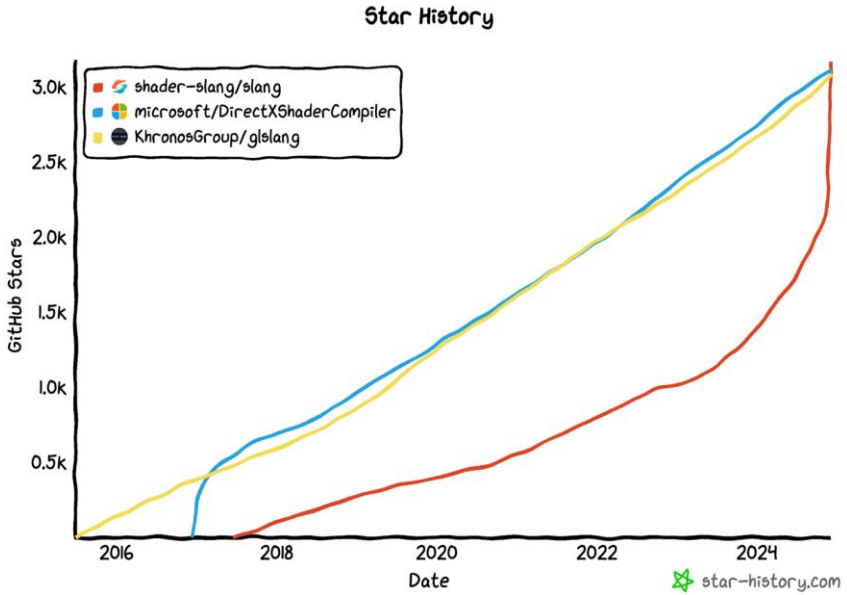
- Slang is used by Aurora path tracing renderer, enables single-source ray tracing codebase
- Ray tracing support just worked!
- Slang shaders are open source & available to check out

- **Binary Size**: 8mb uncompressed
  - No LLVM - we generate C++
  - Includes all backends!

- **Runtime performance**
  - Meet or beat handwritten code
  - Even when using advanced features such as generics

How you can get involved today

- Join the Discord!
  - 200 members and counting!
- File an issue or feature request
- Start a GitHub discussion
- Submit a pull request
- Become a committer!



KHRONOS GROUP

Vulkan®

Vulkan Safety Critical

Shannon Woods, NVIDIA

What is Vulkan Safety Critical

With GPUs at the heart of the modern machinery - it’s even more critical that all GPU workloads are fault tolerant.

Vulkan SC is a streamlined, deterministic and robust API based on Vulkan 1.2 that enables state-of-the-art GPU-accelerated graphics and computation to be deployed in safety-critical systems that are certified to meet industry functional safety standards.

Agenda

- How do we share AI and Graphics on GPU
- How Vulkan SC is the right solve
- Comparing Safety Critical APIs
- Porting Vulkan to Vulkan SC
- Camera -> AI -> Visualization is everywhere
- Platform Availability
- Try it out!

Ex: Visualizing what your self driving car sees

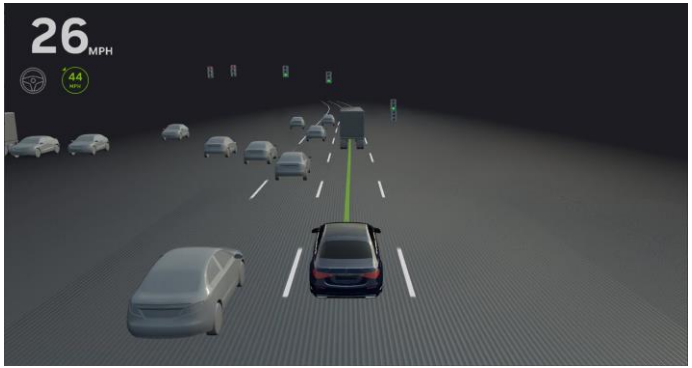
A modern car is handling and visualizing huge amounts of data in realtime:

- Backup camera
- Surround view
- Confidence View

This data needs visualizing!

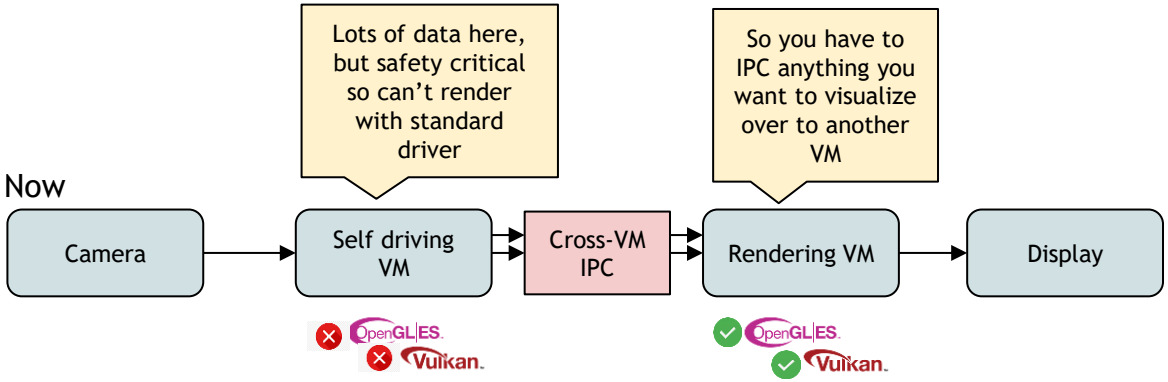
All at 60fps+.

And, the trend is toward ever-bigger data.



How sensor visualization is done today…

Today, self driving VMs are safety certified, so anything in that VM needs to also be provably safe. Thus:

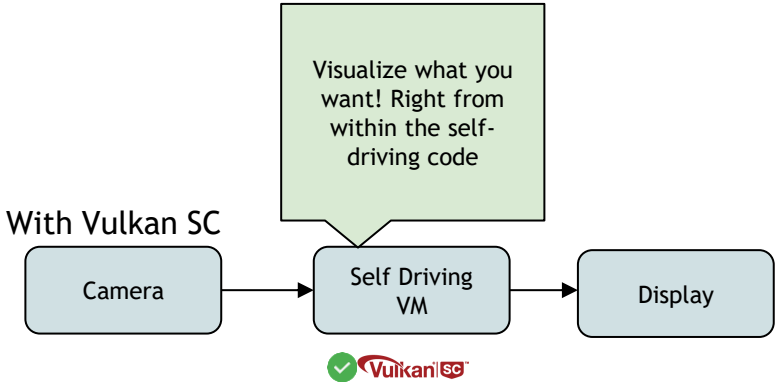


Example:

- Surround view requires streaming real time camera data across VM boundaries
- Need to decide which side of this boundary to do stitching

Vulkan SC: The right way

With Vulkan SC, you can access graphics right inside the safe VM:



Rich graphics that won't interfere with AI workloads on the same VM - magic!

Why Vulkan SC versus GL SC?

Vulkan has some killer features

	Zero copy by default	Predictable command execution	Multithreading	Modern Shading Language and Features	AI/GPU Compute
GL SC	If careful	No	No	No	No
Vulkan SC	Yes	Yes	Yes	Yes	Yes

… and Vulkan SC stays current:

API	Last spec update
GL SC	2019-07
Vulkan	2024-12
Vulkan SC	2024-10

Porting Vulkan to Vulkan SC is easy

Develop in Vulkan! Then:

1. Shift your pipeline and shader compilations to be done offline (~= pipeline cache)
2. Shift from dynamic object and command pool reservation to static reservation during initialization

That's it! If you want more tech details,

see the Khronos website!

It doesn't take much:

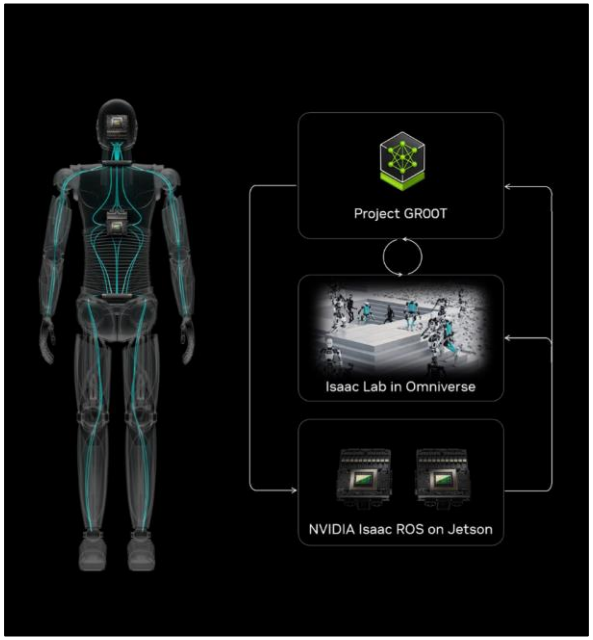
Example	Purpose	# lines changed
vk_triangle	Draw simple triangle	50
Lime SC	NVIDIA's reference confidence view	1000

Cameras -> AI -> Visualization is everywhere you look

… and Vulkan SC is well positioned to help

Surround View System

NVIDIA DRIVE IX



Robotic environments

Avionics Graphics and Compute

CoreAVI VkCoreSC

Vulkan SC is available on these production platforms

Core AVI

Avionics

Automotive

Industrial

NVIDIA

Discrete GPU

Jetson

DriveOS

If you have an NVIDIA GPU, you can use Vulkan SC NOW

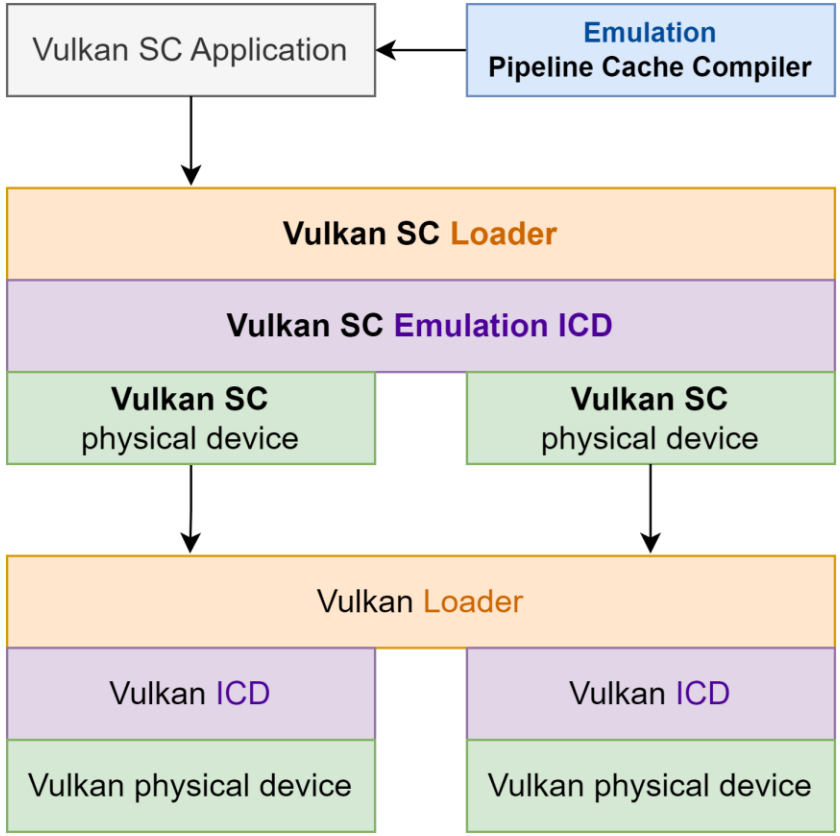
3 Easy Steps

1. Download SDK at NVIDIA: https://developer.nvidia.com/vulkan-sc/
2. Build
3. Run it!

* GPU RTX 20 Series and newer

If you have a Vulkan Driver, you can try Vulkan SC NOW

- **Vulkan SC Emulation**
  - enables running Vulkan SC apps on top of Vulkan driver stacks
- **Source code and instructions available on Github**
  - https://github.com/KhronosGroup/VulkanSC-Emulation



Vulkan SC Tools and Resources

Specification	Vulkan SC 1.0 Specification and Extensions (PDF, HTML)
VulkanSC-Headers	Official API headers
VulkanSC-Loader	ICD loader, documentation and tests (Linux, Windows and QNX)
VulkanSC-ValidationLayers	Validation layers to verify applications are making correct use of the API
VulkanSC-Tools	Tools and utilities: vulkanscinfo command line tool, the mock ICD, and the device simulation layer.
Emulation Layer	Develop Vulkan SC application on top of Vulkan Implementations

Questions?

For more info:

Khronos
https://www.khronos.org/vulkansc/

CoreAVI
https://coreavi.com/product_category/safety-critical-graphics-and-compute/

NVIDIA
https://developer.nvidia.com/vulkan-sc/

Khronos BOFs at SIGGRAPH Asia

Day	Time / Room	Session Title	Standards and Projects
Tuesday 3rd	1:00-2:00PM, G408	Khronos Fast Forward	Vulkan, OpenXR, Slang, ANARI, glTF
Wednesday 4th	1:00-2:00PM, G407	Slang Shading Language	Slang
Wednesday 4th	3:30-4:30PM, G407	Immersive Web with Khronos and W3C	WebGL, WebXR, WebGPU, three.js
Thursday 5th	2:15-3:15PM, G407	OpenXR Update and Roadmap	OpenXR
Thursday 5th	3:30-5:30PM, G407	Vulkan Update and Ecosystem	Vulkan, Vulkan SC, Slang
Friday 6th	1:00-2:00PM, G408	glTF 3D Transmission Format	glTF, VRM Avatar Format

All BOF slides and videos will be uploaded to the
Khronos SIGGRAPH event page

SIGGRAPH ASIA 東京
2024 TOKYO 3–6 DEC

Khronos BOFs

KHRONOS GROUP

Khronos Information
www.khronos.org
memberservices@khronosgroup.org

Presentation Title

Name, Company/Org
Business Title