
Vulkanised
OSAKA 2023
MAY 11th
2023
KHRONOS
GROUP
Khronos Developer Day


Vulkan Ecosystem Developer Tools
Spencer Fricke
LunarG, Inc.
Presented at the May Khronos DevDay in Osaka Japan
LUNARG

日本語のスライド

https://www.lunarg.com/wp-content/uploads/2023/05/J-Vulkan-Ecosystem-Developer-Tools-Osaka.pdf


Vulkan Ecosystem Developer Tools
Spencer Fricke
LunarG, Inc.
Presented at the May Khronos DevDay in Osaka Japan
LUNARG

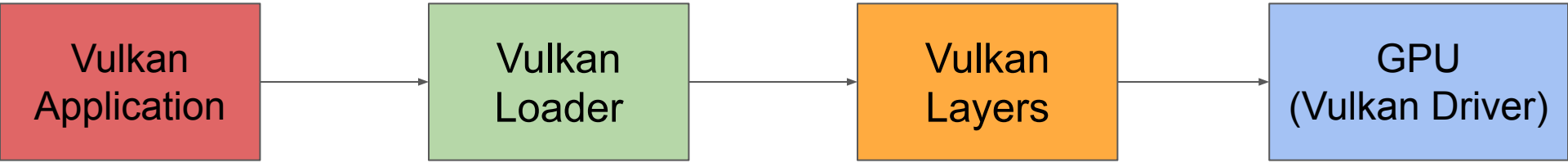
Who is Spencer

- Been working with Vulkan since it came out
  - As a college student
  - With a Hardware vendor
  - Independently
  - With LunarG
- Have seen the ecosystem grow over time

Goal of this talk

- Lots of tools!
- You will not need them all
- Not teaching you details how to use them
  - Tools change all the time, see their documentation
- Being aware they exists is important
  - Prevents reinventing the wheel

Quick Vocabulary Recap

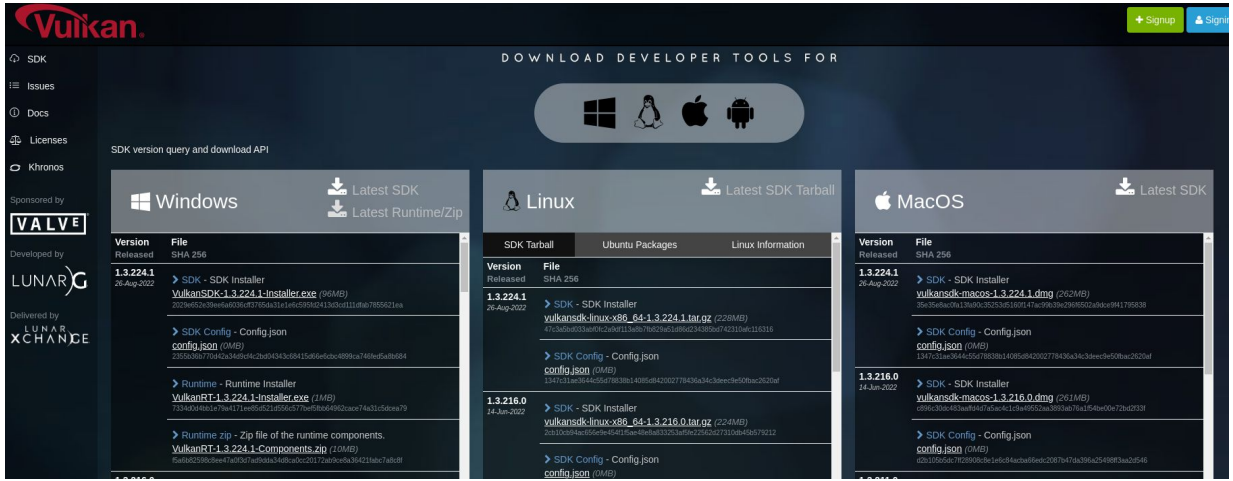


Quick Vocabulary Recap

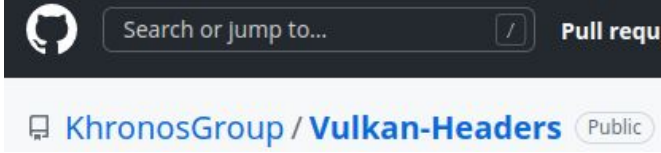
- Offline vs Online
  - Offline = Before execution
  - Online = During execution

Vulkan SDK Tools

- These are included in the SDK as indicated by
  - As of 1.3.243
- Already built and ready to go!
- Download from vulkan.lunarg.com



SDK



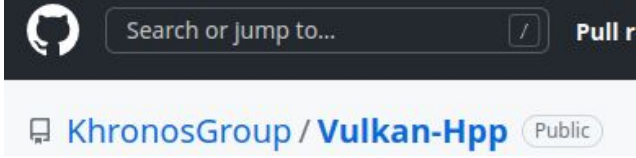
Vulkan-Headers

- C headers to include in program
- Define all structs/functions/etc

SDK

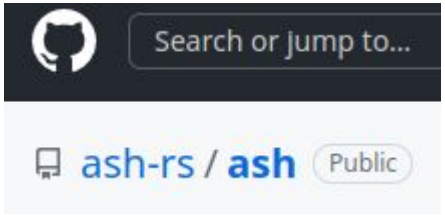
Vulkan-Hpp

- C++ version of headers

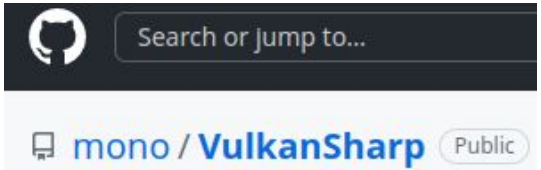


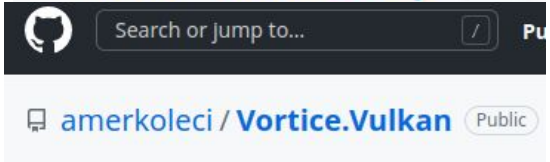
Binding for Language of your choice

Rust

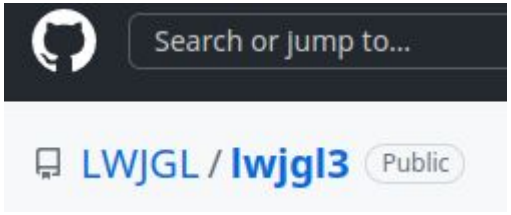


C#

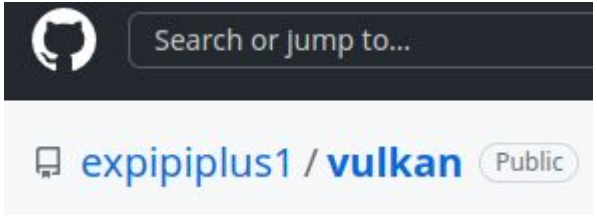




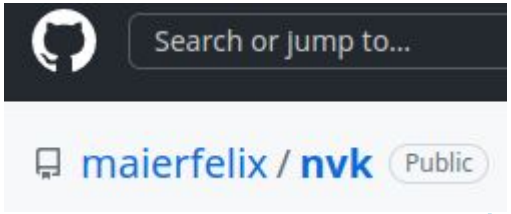
Java



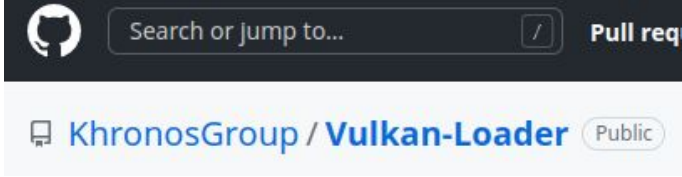
Haskell



Javascript/Typescript

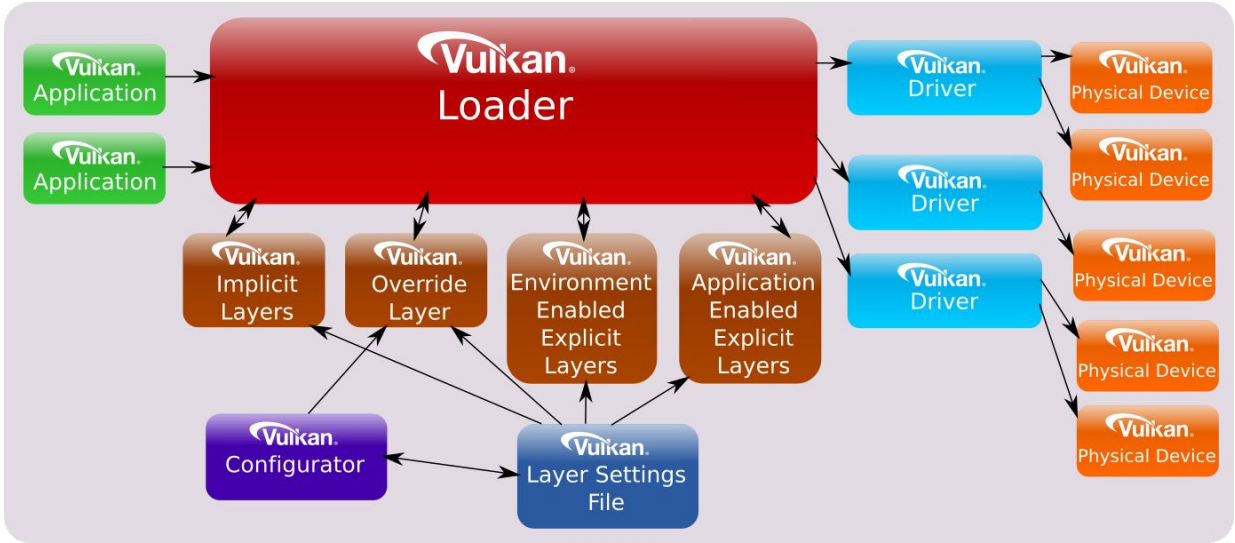


SDK*

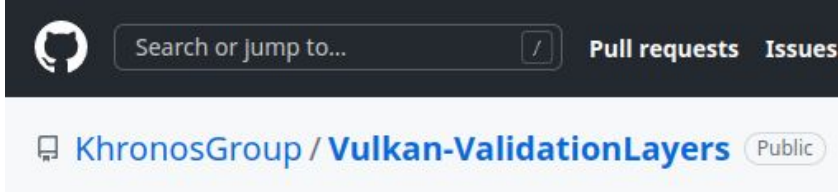


Vulkan-Loader

- Khronos official Vulkan ICD desktop loader for Windows, Linux, and MacOS
  - Android has own implementation of a Vulkan Loader
  - Not included in Windows SDK*
    - Windows driver packages install the Vulkan Loader



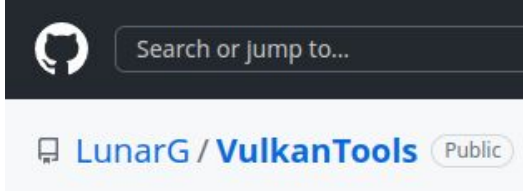
SDK



Validation Layers

- Validates incorrect usage of the Vulkan API
- Will slow down application
- Only for development
- See other talk for more details

SDK



API Dump

- Prints out all the API calls being made
- Best way to see what actually was sent to the GPU
- Also great for bug reports
- Can get large!
  - Options to help reduce size

SDK

API Dump

```
Thread 0, Frame 2:
vkAcquireNextImageKHR(device, swapchain, timeout, semaphore, fence, pImageIndex) returns VkResult VK_SUCCESS (0):
    device:                VkDevice = 0x55b4d33099e0
    swapchain:        VkSwapchainKHR = 0x55b4d331f300
    timeout:               uint64_t = 18446744073709551615
    semaphore:          VkSemaphore = 0x55b4d331f0d0
    fence:                  VkFence = 0
    pImageIndex:           uint32_t* = 2

Thread 0, Frame 2:
vkQueueSubmit(queue, submitCount, pSubmits, fence) returns VkResult VK_SUCCESS (0):
    queue:                  VkQueue = 0x55b4d3219280
    submitCount:           uint32_t = 1
    pSubmits:    const VkSubmitInfo* = 0x7fffeec8c010
        pSubmits[0]:  const VkSubmitInfo = 0x7fffeec8c010:
            sType:           VkStructureType = VK_STRUCTURE_TYPE_SUBMIT_INFO (4)
            pNext:                const void* = NULL
            waitSemaphoreCount:     uint32_t = 1
            pWaitSemaphores: const VkSemaphore* = 0x7fffeec8c500
                pWaitSemaphores[0]: const VkSemaphore = 0x55b4d331f0d0
            pWaitDstStageMask: const VkPipelineStageFlags* = 0x7fffeec8bf2c
                pWaitDstStageMask[0]: const VkPipelineStageFlags = 1024 (VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT)
            commandBufferCount:     uint32_t = 1
            pCommandBuffers: const VkCommandBuffer* = 0x55b4d30191b8
                pCommandBuffers[0]: const VkCommandBuffer = 0x55b4d33ae6f0
            signalSemaphoreCount:   uint32_t = 1
            pSignalSemaphores: const VkSemaphore* = 0x7fffeec8c510
                pSignalSemaphores[0]: const VkSemaphore = 0x55b4d331f140
    fence:                  VkFence = 0x55b4d331f060
```

SDK

API Dump

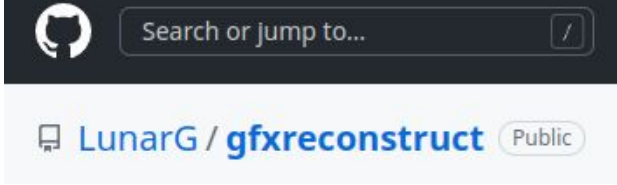
- Configurable

Layer Settings Overview

Setting	Type	Default Value
Output Range	STRING	0-0
Output Format	ENUM	text
Output to File	BOOL	false
Log Filename	SAVE_FILE	stdout
Log Flush After Write	BOOL	true
Name Size	INT	32
Show Types	BOOL	true
Type Size	INT	0
Show Timestamp	BOOL	false
Show Shader	BOOL	false
Show Parameter Details	BOOL	true
Hide Addresses	BOOL	false
Use Spaces	BOOL	true
Indent Size	INT	4
Show Thread and Frame	BOOL	true

SDK

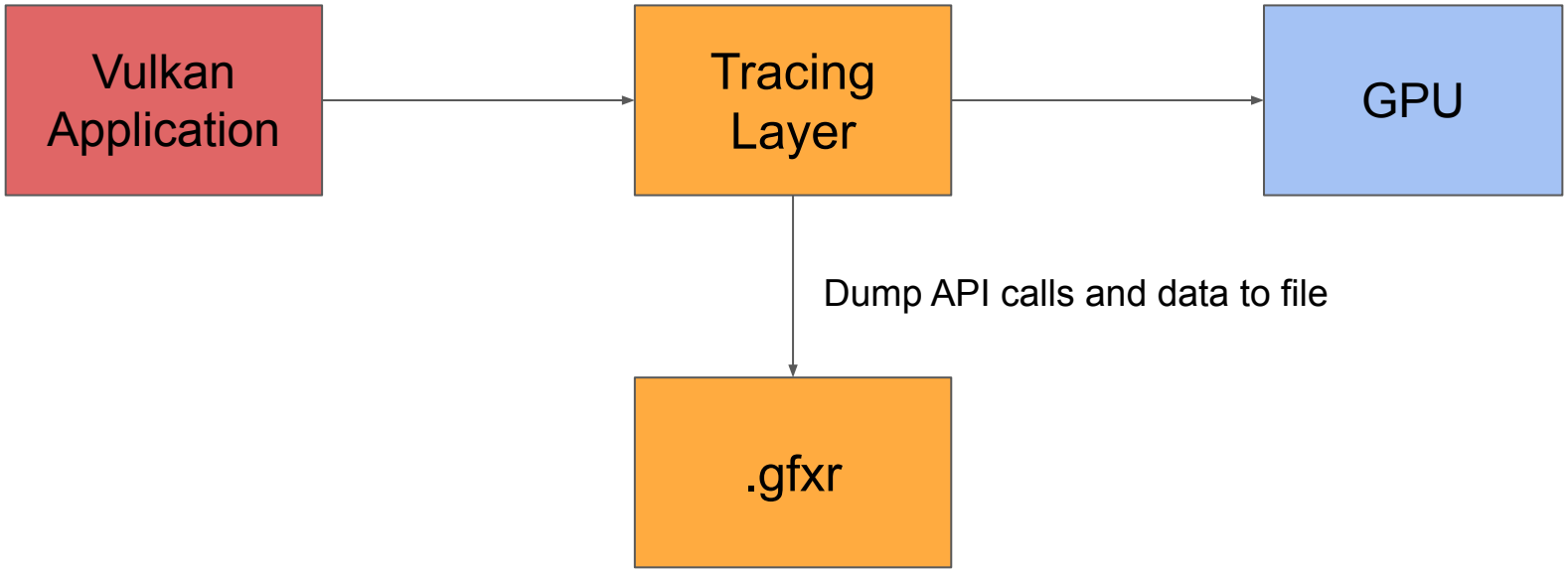
GFXReconstruct



- vktrace / vkreplay successor
- Captures commands to a file (aka “a capture”)
- Replays captures
- Cross-platform support
  - Linux, Android, Windows
- API-agnostic
  - Vulkan and Direct3D 12

SDK

GFXReconstruct - Tracing



SDK

GFXReconstruct - Replay

GFXReconstruct - Use Cases

Save an app's Vulkan commands and replay them consistently

- Driver regression testing
- Architecture simulation
- Silicon bringup
- Debugging
- Bug reporting

Currently in use by several GPU, chipset, platform vendors

SDK

GFXReconstruct

- Additional tools
  - Python wrapper recommend to use
- gfxrecon.py optimize
  - Finds resources not being used to reduce size of capture
  - Useful for trimmed captures
- gfxrecon.py compress
  - Change compression format or decompress
- gfxrecon.py extract
  - extract shader binaries
- gfxrecon.py info
  - Provides info about .gfxr file

SDK

GFXReconstruct

- Additional tools
  - Python wrapper recommend to use
- **gfxrecon.py optimize**
  - Finds resources not being used to reduce size of capture
  - Useful for trimmed captures
- gfxrecon.py compress
  - Change compression format or decompress
- gfxrecon.py extract
  - extract shader binaries
- gfxrecon.py info
  - Provides info about .gfxr file

SDK

GFXReconstruct

- Additional tools
  - Python wrapper recommend to use
- gfxrecon.py optimize
  - Finds resources not being used to reduce size of capture
  - Useful for trimmed captures
- **gfxrecon.py compress**
  - Change compression format or decompress
- gfxrecon.py extract
  - extract shader binaries
- gfxrecon.py info
  - Provides info about .gfxr file

SDK

GFXReconstruct

- Additional tools
  - Python wrapper recommend to use
- gfxrecon.py optimize
  - Finds resources not being used to reduce size of capture
  - Useful for trimmed captures
- gfxrecon.py compress
  - Change compression format or decompress
- **gfxrecon.py extract**
  - extract shader binaries
- gfxrecon.py info
  - Provides info about .gfxr file

SDK

GFXReconstruct

- Additional tools
  - Python wrapper recommend to use
- gfxrecon.py optimize
  - Finds resources not being used to reduce size
  - Useful for trimmed captures
- gfxrecon.py compress
  - Change compression format or decompress
- gfxrecon.py extract
  - extract shader binaries
- **gfxrecon.py info**
  - Provides info about .gfxr file

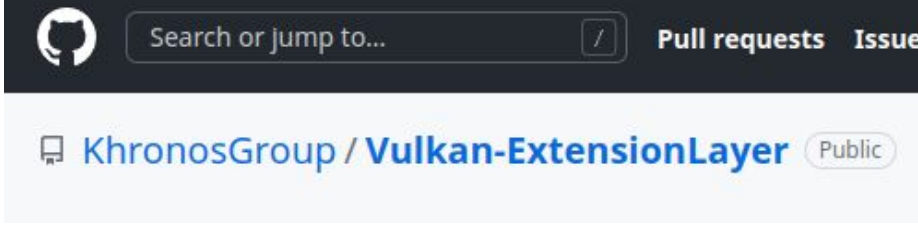
```
$ gfxrecon.py info ~/gfxrecon_capture_20220412T075011.gfxr
File info:
        Compression format: LZ4
        Total frames: 50

Application info:
        Application name: vkcube
        Application version: 0
        Engine name: vkcube
        Engine version: 0
        Target API version: 4198400 (1.1.0)

Physical device info:
        Device name: AMD Radeon RX 6700 XT
        Device ID: 0x73df
        Vendor ID: 0x1002
        Driver version: 8388821 (0x8000d5)
        API version: 4206795 (1.3.203)

Device memory allocation info:
[...]
```

SDK

GFXReconstruct - **gfxrecon.py convert**

- Coverts .gfxr file to JSON
- Useful debugging tool
- API Dump, but on a .gfxr file
  - Data can be easily separated
  - Can be run offline
  - Includes writes to mapped VkMemory objects
  - Additional information

SDK*



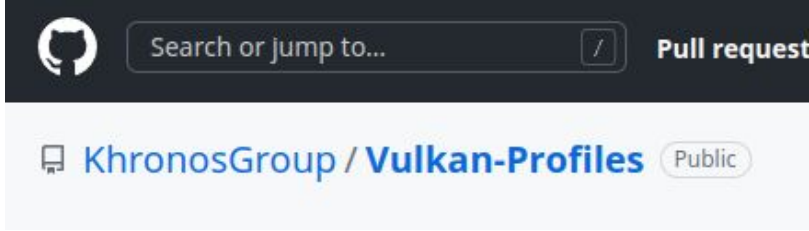
Extension Layer

- Layers to emulate the extension if the driver doesn't support it
  - Will be slower than a native implementation
- Designed to be shipped with application
- Currently support for:
  - VK_KHR_timeline_semaphore
  - VK_KHR_synchronization2 (*only one in SDK)
  - VK_EXT_shader_object
  - VK_NV_memory_decompression

SDK

Vulkan Profiles

- VK_LAYER_KHRONOS_profiles
- Tooling around the Profile JSON schema
- Represents what an application supports



SDK

Vulkan Profiles

```
"extensions": {
    "VK_KHR_swapchain": 70,
    "VK_KHR_sampler_mirror_clamp_to_edge": 3,
```

SDK

Vulkan Profiles

```
"extensions": {
    "VK_KHR_swapchain": 70,
    "VK_KHR_sampler_mirror_clamp_to_edge": 3,
```

```
"VkPhysicalDeviceShaderSubgroupExtendedTypesFeaturesKHR": {
    "shaderSubgroupExtendedTypes": true
},
"VkPhysicalDevice8BitStorageFeaturesKHR": {
    "storageBuffer8BitAccess": true,
    "uniformAndStorageBuffer8BitAccess": true,
    "storagePushConstant8": false
},
```

SDK

Vulkan Profiles

```
 "extensions": {
     "VK_KHR_swapchain": 70,
     "VK_KHR_sampler_mirror_clamp_to_edge": 3,
```

```
"VK_FORMAT_B8G8R8A8_SSCALED": {
    "VkFormatProperties": {
        "linearTilingFeatures": [
            "VK_FORMAT_FEATURE_BLIT_SRC_BIT",
            "VK_FORMAT_FEATURE_SAMPLED_IMAGE_FILTER_LINEAR_BIT"
        ],
        "optimalTilingFeatures": [
            "VK_FORMAT_FEATURE_BLIT_SRC_BIT",
            "VK_FORMAT_FEATURE_SAMPLED_IMAGE_FILTER_LINEAR_BIT"
        ],
        "bufferFeatures": [
            "VK_FORMAT_FEATURE_UNIFORM_TEXEL_BUFFER_BIT",
            "VK_FORMAT_FEATURE_STORAGE_TEXEL_BUFFER_ATOMIC_BIT",
            "VK_FORMAT_FEATURE_VERTEX_BUFFER_BIT"
        ]
    }
},
```

```
"VkPhysicalDeviceShaderSubgroupExtendedTypesFeaturesKHR": {
    "shaderSubgroupExtendedTypes": true
},
"VkPhysicalDevice8BitStorageFeaturesKHR": {
    "storageBuffer8BitAccess": true,
    "uniformAndStorageBuffer8BitAccess": true,
    "storagePushConstant8": false
},
```

SDK

Vulkan Profiles

```
 "extensions": {
     "VK_KHR_swapchain": 70,
     "VK_KHR_sampler_mirror_clamp_to_edge": 3,
```

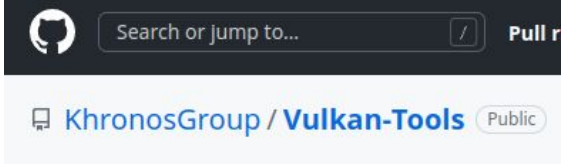
```
"VK_FORMAT_B8G8R8A8_SSCALED": {
    "VkFormatProperties": {
        "linearTilingFeatures": [
            "VK_FORMAT_FEATURE_BLIT_SRC_BIT",
            "VK_FORMAT_FEATURE_SAMPLED_IMAGE_FILTER_LINEAR_BIT"
        ],
        "optimalTilingFeatures": [
            "VK_FORMAT_FEATURE_BLIT_SRC_BIT",
            "VK_FORMAT_FEATURE_SAMPLED_IMAGE_FILTER_LINEAR_BIT"
        ],
        "bufferFeatures": [
            "VK_FORMAT_FEATURE_UNIFORM_TEXEL_BUFFER_BIT",
            "VK_FORMAT_FEATURE_STORAGE_TEXEL_BUFFER_ATOMIC_BIT",
            "VK_FORMAT_FEATURE_VERTEX_BUFFER_BIT"
        ]
    }
},
```

```
"VkPhysicalDeviceShaderSubgroupExtendedTypesFeaturesKHR": {
    "shaderSubgroupExtendedTypes": true
},
"VkPhysicalDevice8BitStorageFeaturesKHR": {
    "storageBuffer8BitAccess": true,
    "uniformAndStorageBuffer8BitAccess": true,
    "storagePushConstant8": false
},
```

```
"maxUniformBufferRange": 4294967295,
"maxVertexInputAttributeOffset": 4294967295,
"maxVertexInputAttributes": 64,
"maxVertexInputBindingStride": 16383,
"maxVertexInputBindings": 32,
"maxVertexOutputComponents": 128,
"maxViewports": 16,
```

Vulkan Profiles - Use Case

- Validation Layers need to test extensions when we don’t have HW that supports it
- Use MockICD as our driver
- Use Profile Layers to make it seem we support the extension



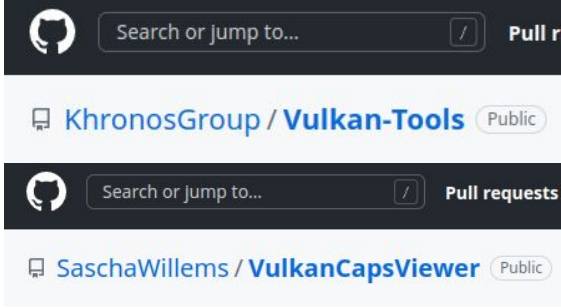
MockICD

- Null driver
- Will pretend to be a driver, but will do no work
- Great if you need to test a layer and don’t care about the GPU

SDK

Vulkan Info and Vulkan Caps Viewer

- Shows what is supported on your device
- **Vulkan Info** == command line tool
- **Vulkan Caps View** == GUI tool
  - Results can be viewed on https://vulkan.gpuinfo.org/



SDK

Vulkan Info

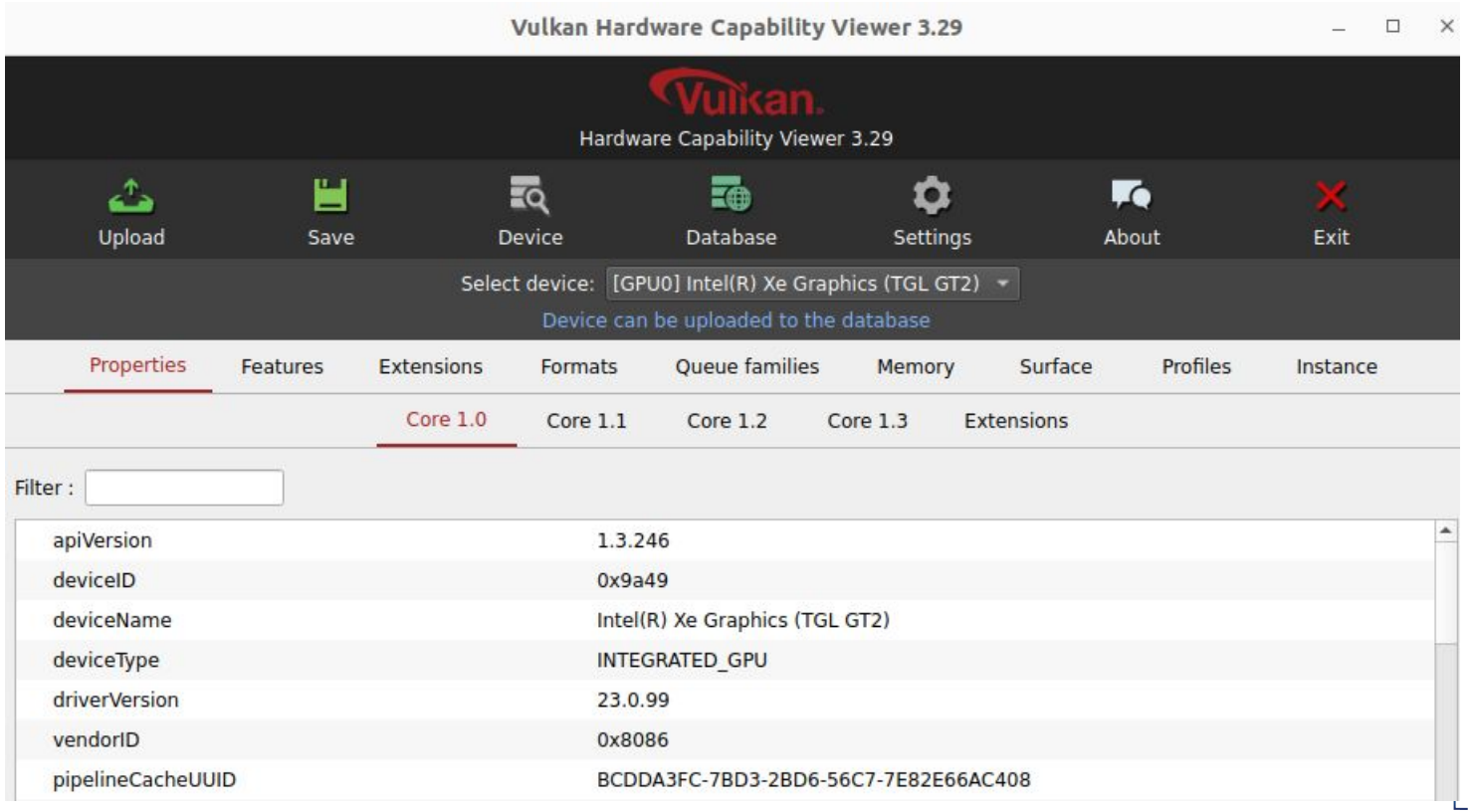
```
$ vulkaninfo
```

```
$ vulkaninfo --summary
```

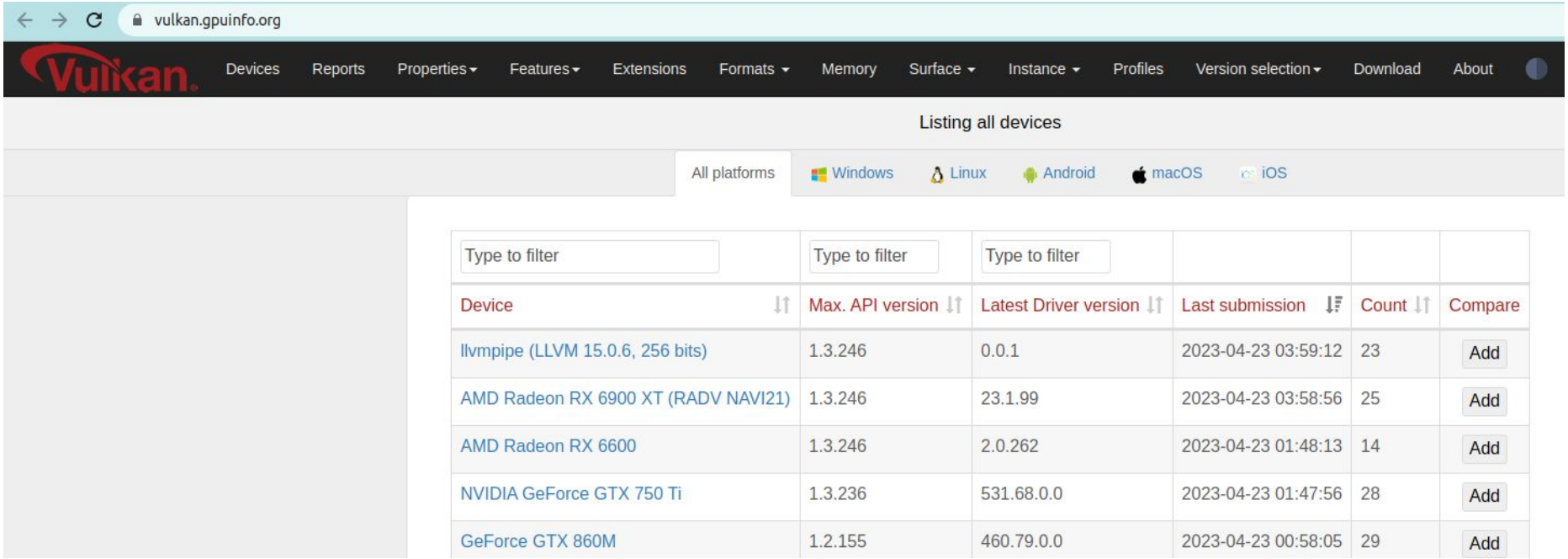
```
Device Properties and Extensions:
=================================
GPU0:
VkPhysicalDeviceProperties:
---------------------------
        apiVersion        = 1.3.246 (4206838)
        driverVersion     = 23.0.99 (96469091)
        vendorID          = 0x8086
        deviceID          = 0x9a49
        deviceType        = PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU
        deviceName        = Intel(R) Xe Graphics (TGL GT2)
        pipelineCacheUUID = bcdda3fc-7bd3-2bd6-56c7-7e82e66ac408

VkPhysicalDeviceLimits:
-----------------------
        maxImageDimension1D                               = 16384
        maxImageDimension2D                               = 16384
        maxImageDimension3D                               = 2048
        maxImageDimensionCube                             = 16384
        maxImageArrayLayers                               = 2048
```

SDK

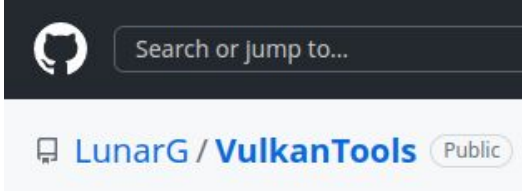
Vulkan Caps Viewer



Vulkan Caps Viewer

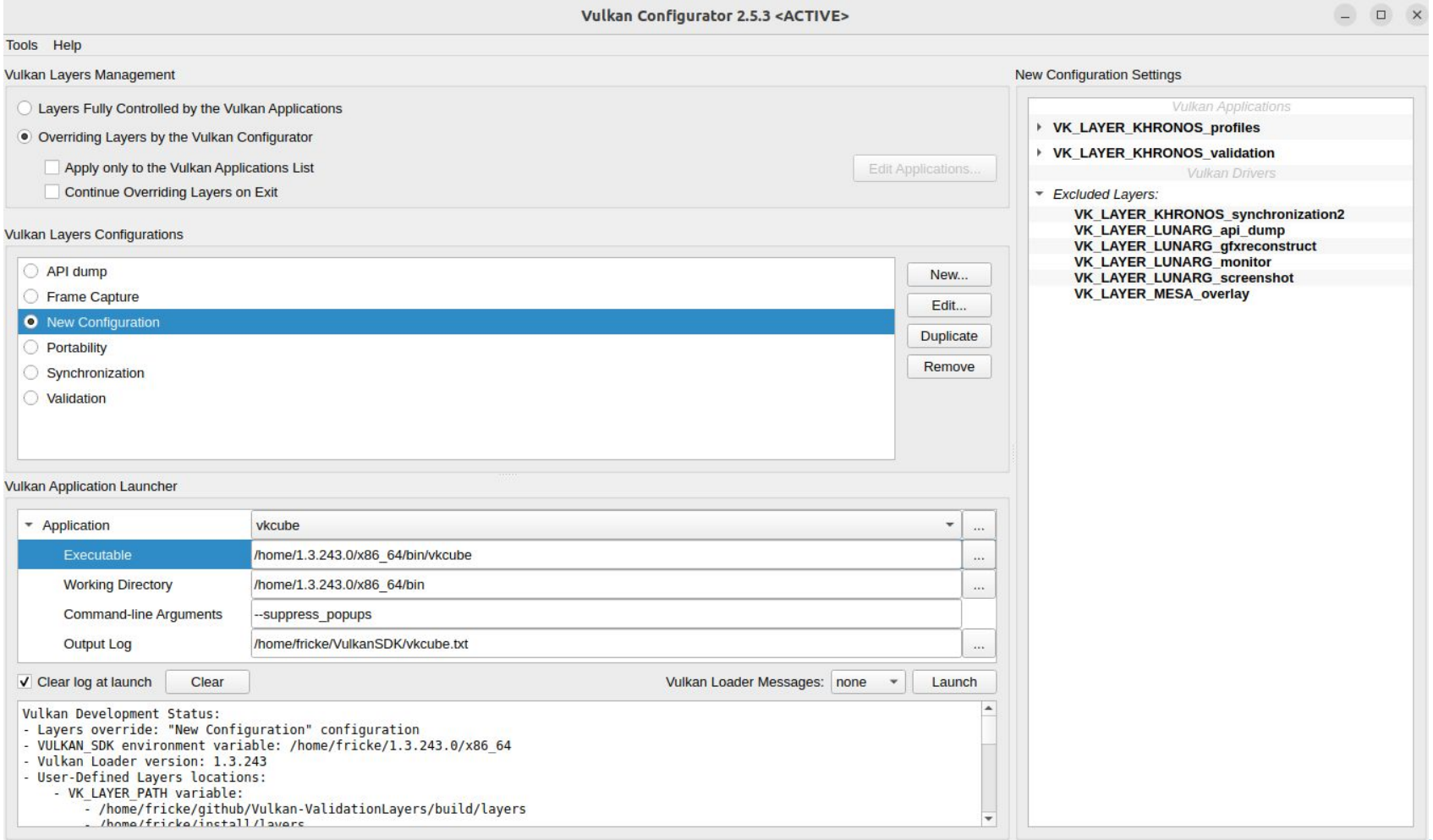


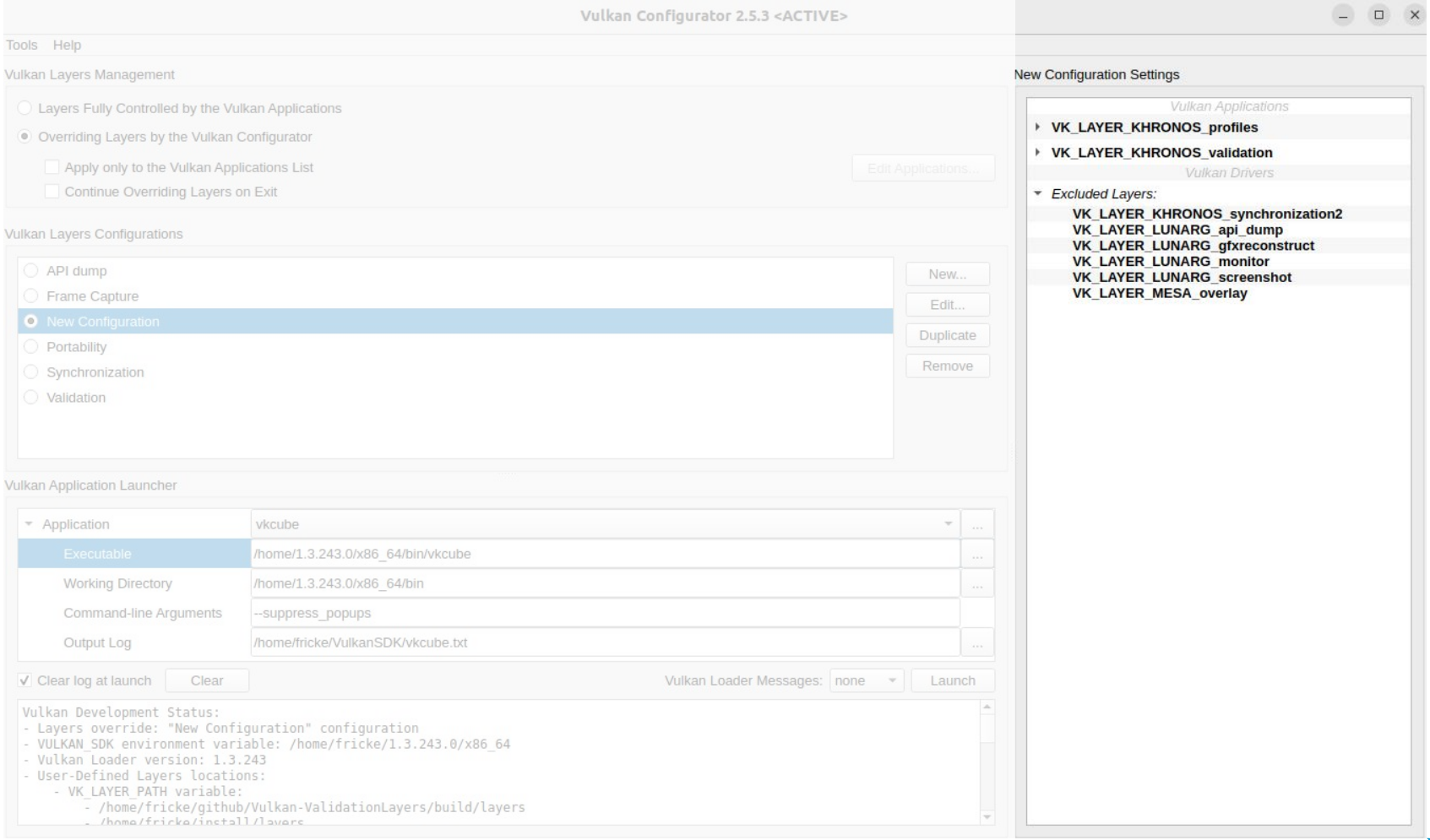
SDK

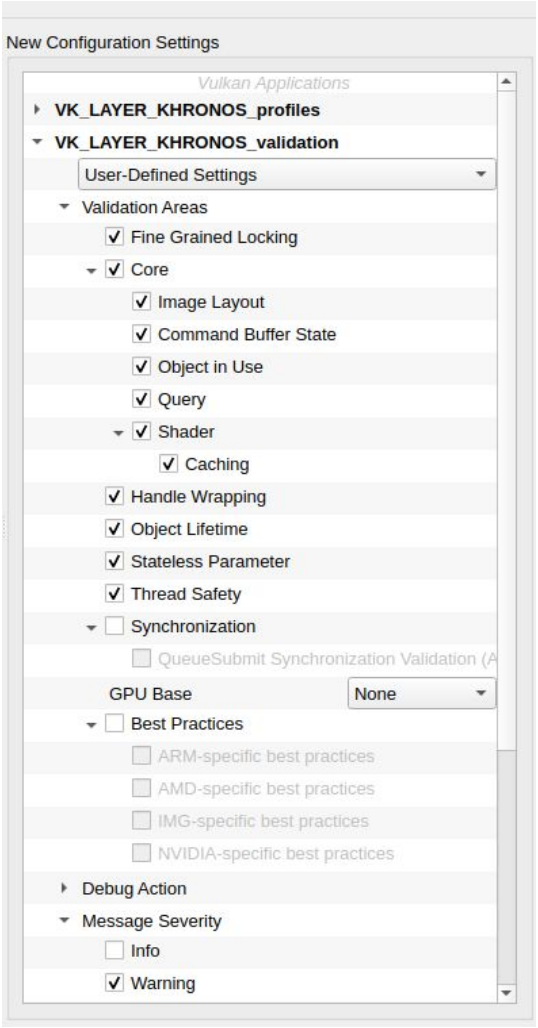


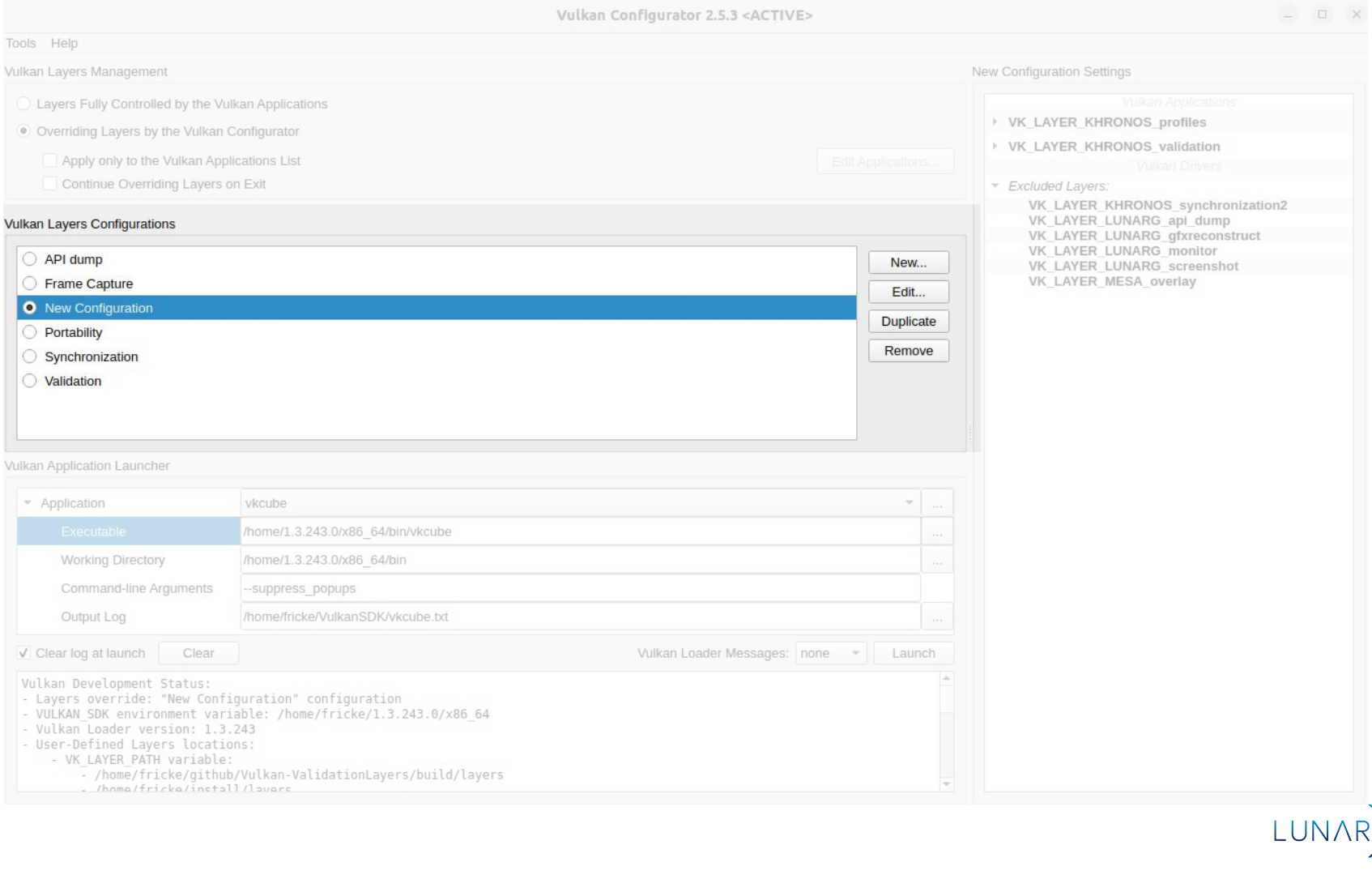
Vulkan Configurator (VkConfig)

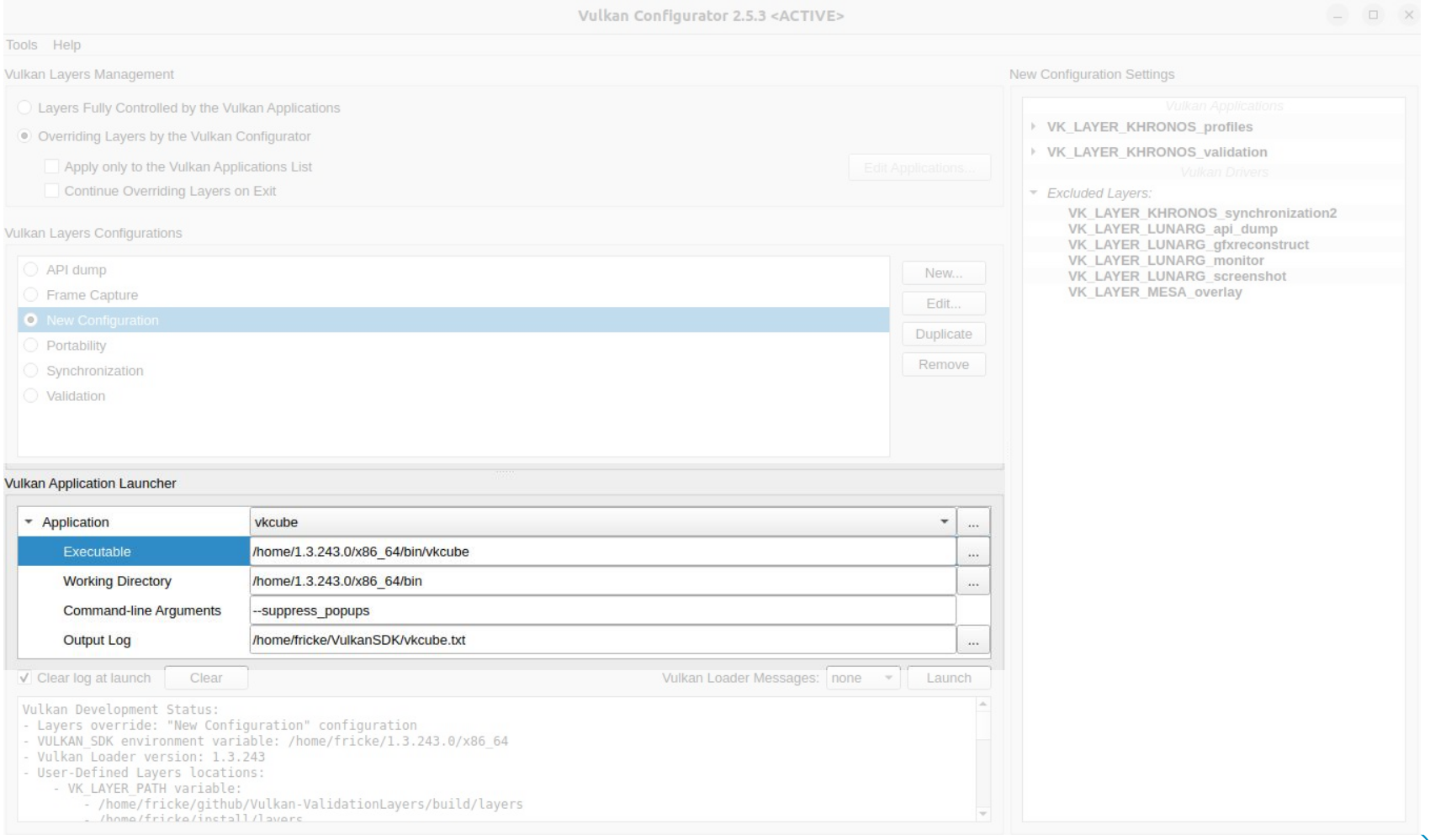
- Enabling and configuring layers can be hard
- VkConfig makes it easy
  - GUI tool
  - Lists available options visible for selection


Vulkan Configurator 2.5.3 <ACTIVE>
Tools Help
Vulkan Layers Management
Layers Fully Controlled by the Vulkan Applications
Overriding Layers by the Vulkan Configurator
Apply only to the Vulkan Applications List
Continue Overriding Layers on Exit
Edit Applications...
Vulkan Layers Configurations
API dump
Frame Capture
New Configuration
Portability
Synchronization
Validation
New...
Edit...
Duplicate
Remove
Vulkan Application Launcher
Application
vkcube
Executable
/home/1.3.243.0/x86_64/bin/vkcube
Working Directory
/home/1.3.243.0/x86_64/bin
Command-line Arguments
--suppress_popups
Output Log
/home/fricke/VulkanSDK/vkcube.txt
Clear log at launch
Clear
Vulkan Loader Messages:
none
Launch
Vulkan Development Status:
- Layers override: "New Configuration" configuration
- VULKAN_SDK environment variable: /home/fricke/1.3.243.0/x86_64
- Vulkan Loader version: 1.3.243
- User-Defined Layers locations:
- VK_LAYER_PATH variable:
- /home/fricke/github/Vulkan-ValidationLayers/build/layers
New Configuration Settings
Vulkan Applications
VK_LAYER_KHRONOS_profiles
VK_LAYER_KHRONOS_validation
Vulkan Drivers
Excluded Layers:
VK_LAYER_KHRONOS_synchronization2
VK_LAYER_LUNARG_api_dump
VK_LAYER_LUNARG_gfxreconstruct
VK_LAYER_LUNARG_monitor
VK_LAYER_LUNARG_screenshot
VK_LAYER_MESA_overlay

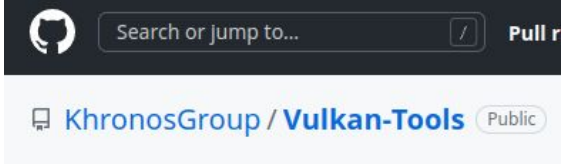

Vulkan Configurator 2.5.3 <ACTIVE>
Tools Help
Vulkan Layers Management
Layers Fully Controlled by the Vulkan Applications
Overriding Layers by the Vulkan Configurator
Apply only to the Vulkan Applications List
Continue Overriding Layers on Exit
Edit Applications...
Vulkan Layers Configurations
API dump
Frame Capture
New Configuration
Portability
Synchronization
Validation
New...
Edit...
Duplicate
Remove
Vulkan Application Launcher
Application
vkcube
Executable
/home/1.3.243.0/x86_64/bin/vkcube
Working Directory
/home/1.3.243.0/x86_64/bin
Command-line Arguments
--suppress_popups
Output Log
/home/fricke/VulkanSDK/vkcube.txt
Clear log at launch
Clear
Vulkan Loader Messages:
none
Launch
Vulkan Development Status:
- Layers override: "New Configuration" configuration
- VULKAN_SDK environment variable: /home/fricke/1.3.243.0/x86_64
- Vulkan Loader version: 1.3.243
- User-Defined Layers locations:
- VK_LAYER_PATH variable:
- /home/fricke/github/Vulkan-ValidationLayers/build/layers
New Configuration Settings
Vulkan Applications
VK_LAYER_KHRONOS_profiles
VK_LAYER_KHRONOS_validation
Vulkan Drivers
Excluded Layers:
VK_LAYER_KHRONOS_synchronization2
VK_LAYER_LUNARG_api_dump
VK_LAYER_LUNARG_gfxreconstruct
VK_LAYER_LUNARG_monitor
VK_LAYER_LUNARG_screenshot
VK_LAYER_MESA_overlay


New Configuration Settings
Vulkan Applications
VK_LAYER_KHRONOS_profiles
VK_LAYER_KHRONOS_validation
User-Defined Settings
Validation Areas
Fine Grained Locking
Core
Image Layout
Command Buffer State
Object in Use
Query
Shader
Caching
Handle Wrapping
Object Lifetime
Stateless Parameter
Thread Safety
Synchronization
QueueSubmit Synchronization Validation (A
GPU Base
None
Best Practices
ARM-specific best practices
AMD-specific best practices
IMG-specific best practices
NVIDIA-specific best practices
Debug Action
Message Severity
Info
Warning


Vulkan Configurator 2.5.3 <ACTIVE>
Tools Help
Vulkan Layers Management
Layers Fully Controlled by the Vulkan Applications
Overriding Layers by the Vulkan Configurator
Apply only to the Vulkan Applications List
Continue Overriding Layers on Exit
Edit Applications...
New Configuration Settings
Vulkan Applications
VK_LAYER_KHRONOS_profiles
VK_LAYER_KHRONOS_validation
Vulkan Drivers
Excluded Layers:
VK_LAYER_KHRONOS_synchronization2
VK_LAYER_LUNARG_api_dump
VK_LAYER_LUNARG_gfxreconstruct
VK_LAYER_LUNARG_monitor
VK_LAYER_LUNARG_screenshot
VK_LAYER_MESA_overlay
Vulkan Layers Configurations
API dump
Frame Capture
New Configuration
Portability
Synchronization
Validation
New...
Edit...
Duplicate
Remove
Vulkan Application Launcher
Application
vkcube
Executable
/home/1.3.243.0/x86_64/bin/vkcube
Working Directory
/home/1.3.243.0/x86_64/bin
Command-line Arguments
--suppress_popups
Output Log
/home/fricke/VulkanSDK/vkcube.txt
Clear log at launch
Clear
Vulkan Loader Messages:
none
Launch
Vulkan Development Status:
- Layers override: "New Configuration" configuration
- VULKAN_SDK environment variable: /home/fricke/1.3.243.0/x86_64
- Vulkan Loader version: 1.3.243
- User-Defined Layers locations:
- VK_LAYER_PATH variable:
- /home/fricke/github/Vulkan-ValidationLayers/build/layers

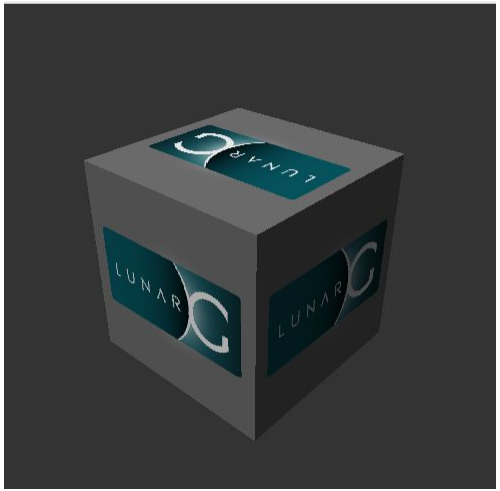

Vulkan Configurator 2.5.3 <ACTIVE>
Tools Help
Vulkan Layers Management
Layers Fully Controlled by the Vulkan Applications
Overriding Layers by the Vulkan Configurator
Apply only to the Vulkan Applications List
Continue Overriding Layers on Exit
Edit Applications...
Vulkan Layers Configurations
API dump
Frame Capture
New Configuration
Portability
Synchronization
Validation
New...
Edit...
Duplicate
Remove
New Configuration Settings
Vulkan Applications
VK_LAYER_KHRONOS_profiles
VK_LAYER_KHRONOS_validation
Vulkan Drivers
Excluded Layers:
VK_LAYER_KHRONOS_synchronization2
VK_LAYER_LUNARG_api_dump
VK_LAYER_LUNARG_gfxreconstruct
VK_LAYER_LUNARG_monitor
VK_LAYER_LUNARG_screenshot
VK_LAYER_MESA_overlay
Vulkan Application Launcher
Application
vkcube
Executable
/home/1.3.243.0/x86_64/bin/vkcube
Working Directory
/home/1.3.243.0/x86_64/bin
Command-line Arguments
--suppress_popups
Output Log
/home/fricke/VulkanSDK/vkcube.txt
Clear log at launch
Clear
Vulkan Loader Messages:
none
Launch
Vulkan Development Status:
- Layers override: "New Configuration" configuration
- VULKAN_SDK environment variable: /home/fricke/1.3.243.0/x86_64
- Vulkan Loader version: 1.3.243
- User-Defined Layers locations:
- VK_LAYER_PATH variable:
- /home/fricke/github/Vulkan-ValidationLayers/build/layers

SDK



VkCube

- “Is everything set up correctly” app
- "Lightswitch test": Is my Vulkan installation working?
  - (Loader, layers, driver, etc)

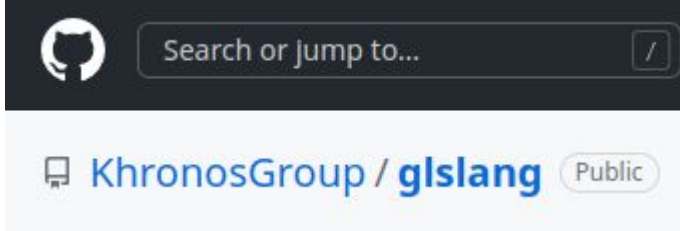


Shaders

- See SPIR-V talk if you want to make your own SPIR-V Tool
- Lots of tools focused on shaders

SDK

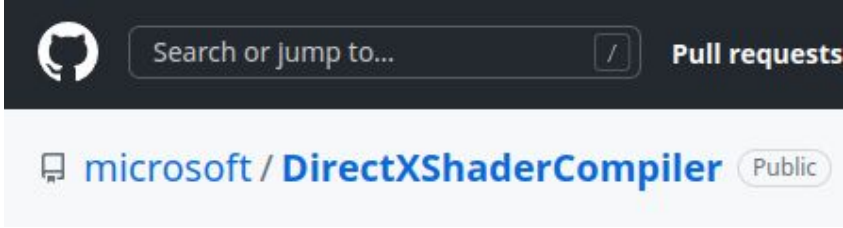
glslang



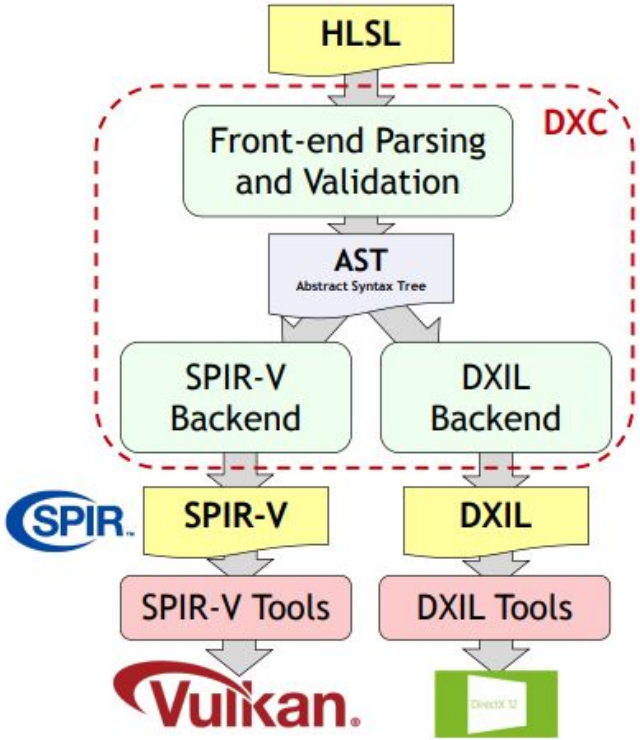
- Khronos reference GLSL compiler
- Most common way to bring GLSL to SPIR-V
- Can also compile HLSL to SPIR-V
  - Up to Shader Model 5

SDK

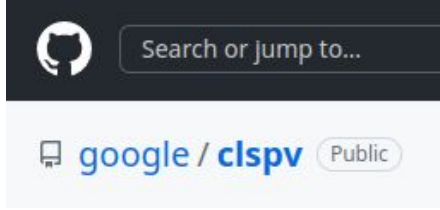
DXC



- Tool for taking HLSL to SPIR-V



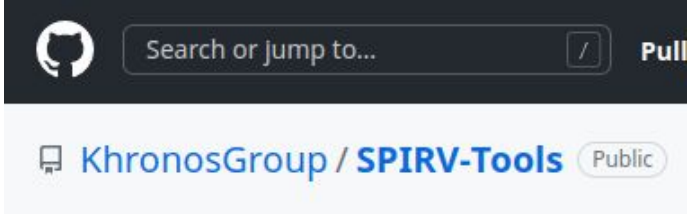
clspv



- Turns OpenCL kernels into Vulkan compatible SPIR-V
- Works well, requires a lot of work arounds
  - https://github.com/google/clspv/blob/main/docs/OpenCLCOnVulkan.md

SDK

SPIR-V Tools



- Collections of Tools maintained by the Khronos Groups

SDK

spirv-as and spirv-dis

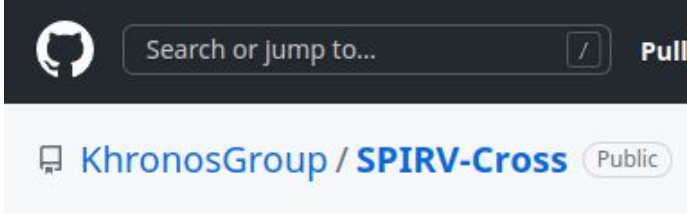
- as == assembler
- dis == disassembler
- Go between SPIR-V binary and readable

spirv-opt

- Set of passes that can be used to optimized SPIR-V
- Designed to be run offline

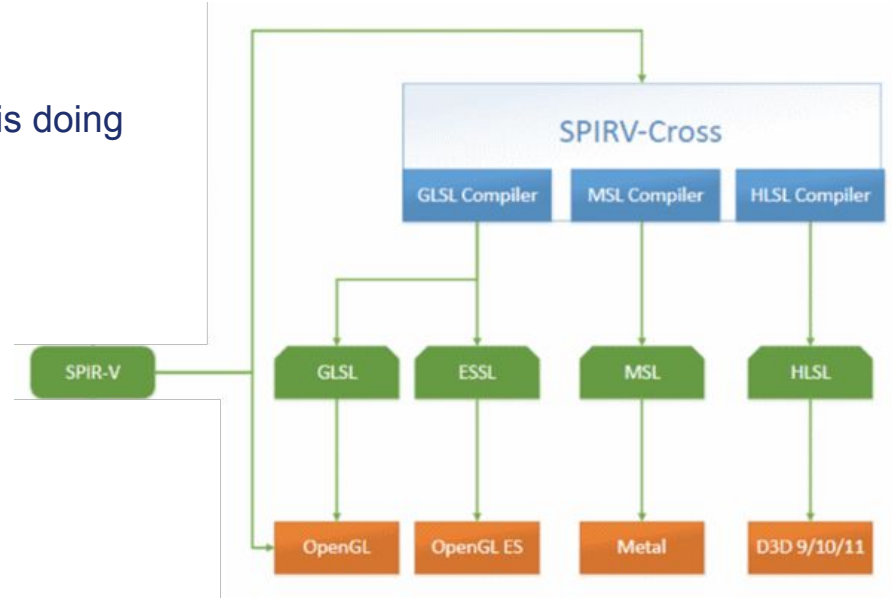
spirv-diff, spirv-reduce, spirv-fuzz

- **spirv-diff**
  - Shows a diff between 2 SPIR-V modules
- **spirv-reduce**
  - Tries to simplify a SPIR-V shader as much as possible
- **spirv-fuzz**
  - Applies semantics-preserving transformations

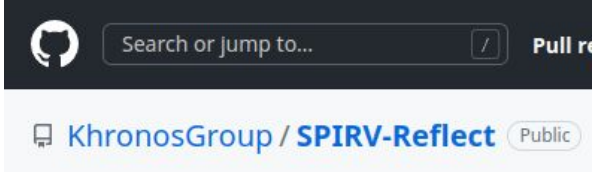


SPIRV-Cross

- Takes SPIR-V and tries to turn it into a human readable language (GLSL, HLSL, etc)
- Main two use cases
  - Better understand what the shader is doing
  - Portability



SDK



SPIRV-Reflect

- Runtime library to help parse what is in the SPIR-V file
  - Descriptor locations
  - Getting interface variable info
  - etc

SDK

SPIRV-Reflect

```
// Generate reflection data for a shader
SpvReflectShaderModule module;
SpvReflectResult result = spvReflectCreateShaderModule(spirv_nbytes, spirv_code, &module);
assert(result == SPV_REFLECT_RESULT_SUCCESS);

// Enumerate and extract shader's input variables
uint32_t var_count = 0;
result = spvReflectEnumerateInputVariables(&module, &var_count, NULL);
assert(result == SPV_REFLECT_RESULT_SUCCESS);
SpvReflectInterfaceVariable** input_vars =
  (SpvReflectInterfaceVariable**)malloc(var_count * sizeof(SpvReflectInterfaceVariable*));
result = spvReflectEnumerateInputVariables(&module, &var_count, input_vars);
assert(result == SPV_REFLECT_RESULT_SUCCESS);
```

SDK

SPIRV-Reflect

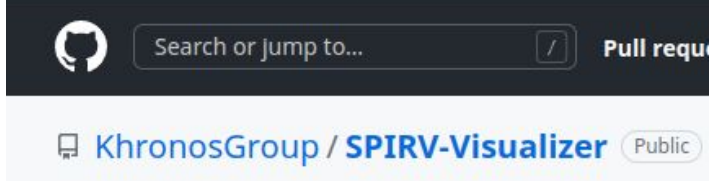
```
// Generate reflection data for a shader
SpvReflectShaderModule module;
SpvReflectResult result = spvReflectCreateShaderModule(spirv_nbytes, spirv_code, &module);
assert(result == SPV_REFLECT_RESULT_SUCCESS);

// Enumerate and extract shader's input variables
uint32_t var_count = 0;
result = spvReflectEnumerateInputVariables(&module, &var_count, NULL);
assert(result == SPV_REFLECT_RESULT_SUCCESS);
SpvReflectInterfaceVariable** input_vars =
  (SpvReflectInterfaceVariable**)malloc(var_count * sizeof(SpvReflectInterfaceVariable*));
result = spvReflectEnumerateInputVariables(&module, &var_count, input_vars);
assert(result == SPV_REFLECT_RESULT_SUCCESS);
```

SDK

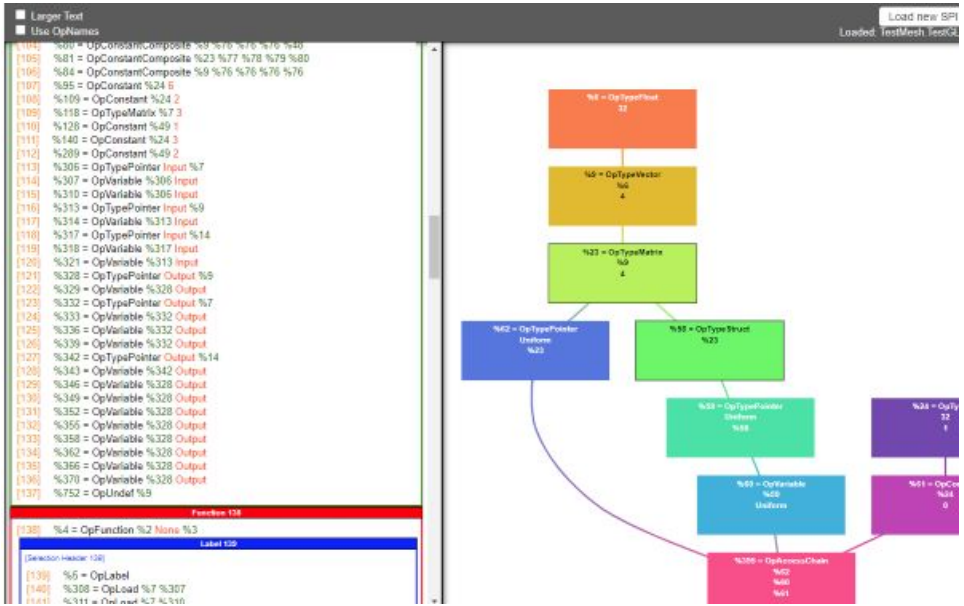
SPIRV-Reflect

```
// Generate reflection data for a shader
SpvReflectShaderModule module;
SpvReflectResult result = spvReflectCreateShaderModule(spirv_nbytes, spirv_code, &module);
assert(result == SPV_REFLECT_RESULT_SUCCESS);

// Enumerate and extract shader's input variables
result = spvReflectEnumerateInputVariables(&module, &var_count, NULL);
SpvReflectInterfaceVariable** input_vars =
result = spvReflectEnumerateInputVariables(&module, &var_count, input_vars);
```

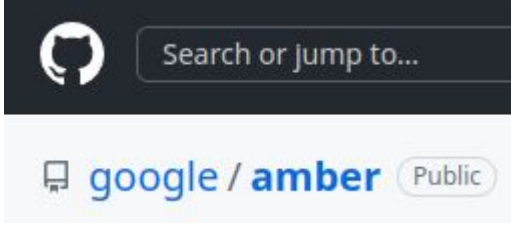
SPIRV-Visualizer



- Web GUI tool to view a SPIR-V
- Nicer than using spirv-dis
- www.khronos.org/spir/visualizer/



Amber



- Provide a shader and info that describes the intended action
- Will take it, generate the correct Vulkan, then run it
- Designed for isolating bugs

Amber

```
SHADER vertex vtex_shader PASSTHROUGH
SHADER fragment frag_shader GLSL
#version 430

layout(location = 0) in vec4 color_in;
layout(location = 0) out vec4 color_out;

void main() {
  color_out = color_in;
}
END

BUFFER img_buf FORMAT B8G8R8A8_UNORM

PIPELINE graphics my_pipeline
  ATTACH vtex_shader
  ATTACH frag_shader

  FRAMEBUFFER_SIZE 256 256
  BIND BUFFER img_buf AS color LOCATION 0
END

CLEAR my_pipeline
EXPECT img_buf IDX 0 0 SIZE 256 256 EQ_RGBA 0 0 0 0
```

Amber

```
SHADER vertex vtex_shader PASSTHROUGH
SHADER fragment frag_shader GLSL
#version 430

layout(location = 0) in vec4 color_in;
layout(location = 0) out vec4 color_out;

void main() {
  color_out = color_in;
}
END

BUFFER img_buf FORMAT B8G8R8A8_UNORM

PIPELINE graphics my_pipeline
  ATTACH vtex_shader
  ATTACH frag_shader

  FRAMEBUFFER_SIZE 256 256
  BIND BUFFER img_buf AS color LOCATION 0
END

CLEAR my_pipeline
EXPECT img_buf IDX 0 0 SIZE 256 256 EQ_RGBA 0 0 0 0
```

Amber

```
SHADER vertex vtex_shader PASSTHROUGH
SHADER fragment frag_shader GLSL
#version 430

layout(location = 0) in vec4 color_in;
layout(location = 0) out vec4 color_out;

void main() {
  color_out = color_in;
}
END

BUFFER img_buf FORMAT B8G8R8A8_UNORM

PIPELINE graphics my_pipeline
  ATTACH vtex_shader
  ATTACH frag_shader

  FRAMEBUFFER_SIZE 256 256
  BIND BUFFER img_buf AS color LOCATION 0
END

CLEAR my_pipeline
EXPECT img_buf IDX 0 0 SIZE 256 256 EQ_RGBA 0 0 0 0
```

Amber

```
SHADER vertex vtex_shader PASSTHROUGH
SHADER fragment frag_shader GLSL
#version 430

layout(location = 0) in vec4 color_in;
layout(location = 0) out vec4 color_out;

void main() {
  color_out = color_in;
}
END

BUFFER img_buf FORMAT B8G8R8A8_UNORM

PIPELINE graphics my_pipeline
  ATTACH vtex_shader
  ATTACH frag_shader

  FRAMEBUFFER_SIZE 256 256
  BIND BUFFER img_buf AS color LOCATION 0
END

CLEAR my_pipeline
EXPECT img_buf IDX 0 0 SIZE 256 256 EQ_RGBA 0 0 0 0
```

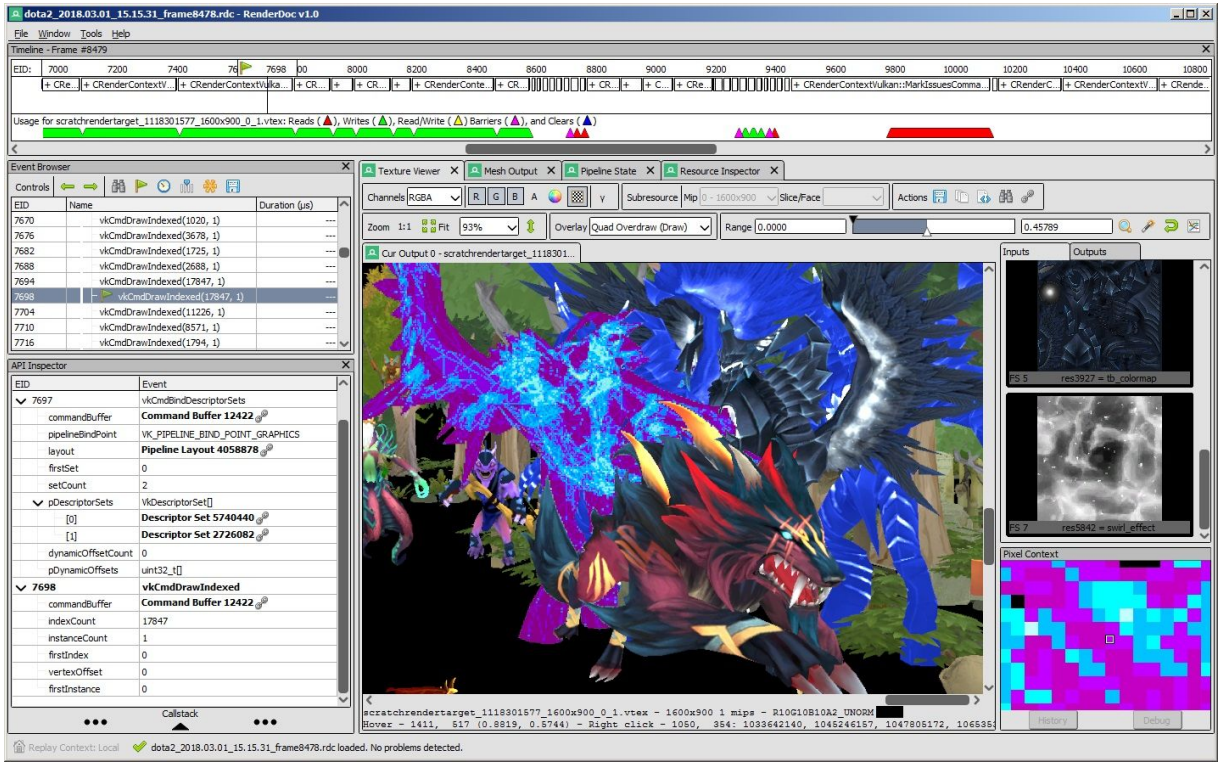
Amber

```
SHADER vertex vtex_shader PASSTHROUGH
SHADER fragment frag_shader GLSL
#version 430

layout(location = 0) in vec4 color_in;
layout(location = 0) out vec4 color_out;

void main() {
  color_out = color_in;
}
END

BUFFER img_buf FORMAT B8G8R8A8_UNORM

PIPELINE graphics my_pipeline
  ATTACH vtex_shader
  ATTACH frag_shader

  FRAMEBUFFER_SIZE 256 256
  BIND BUFFER img_buf AS color LOCATION 0
END

CLEAR my_pipeline
EXPECT img_buf IDX 0 0 SIZE 256 256 EQ_RGBA 0 0 0 0
```

RenderDoc

Search or jump to...

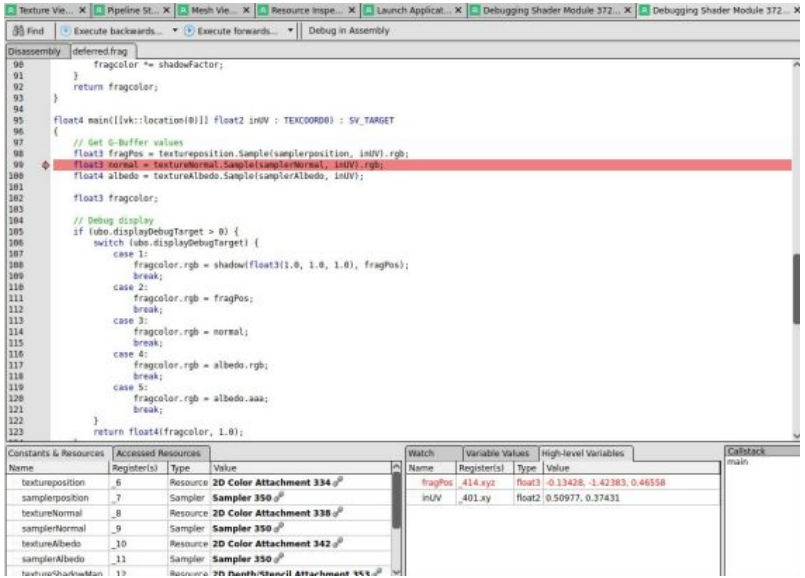
baldurk / renderdoc Public



- Popular tool for debugging
- Works per-frame

RenderDoc

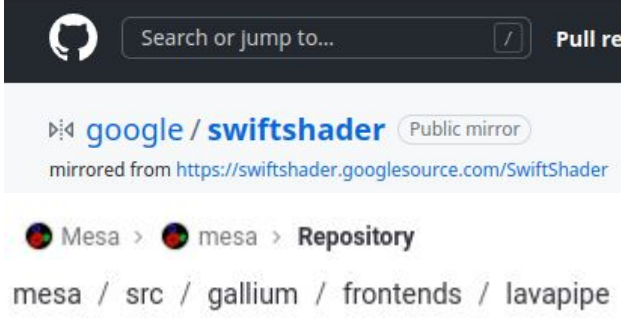
- Shader level debugging now supported
- See [Greg Fischer's 2023 Vulkanised talk!]



Hardware / Platform Profiling tool

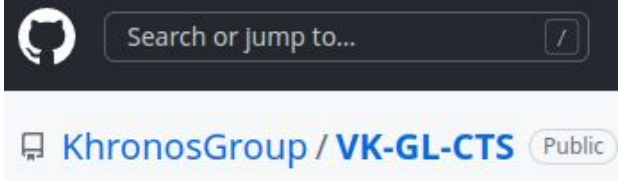
- AMD Radeon GPU Profiler
- Android GUI Inspector (AGI)
- ARM Mobile Studio
- Intel Graphics Performance Analyzers
- NVIDIA Nsight Tools
- Qualcomm Snapdragon Profiler
- Tracy Profiler (cross-vendor)

Swiftshader and Lavapipe



- Open source CPU implementations
- **Swiftshader** created by Google
- **Lavapipe** created by Mesa team
- Can be useful to remove issue of driver bugs
- Note: These are not “reference drivers,” but **can** be used as **a** reference

CTS

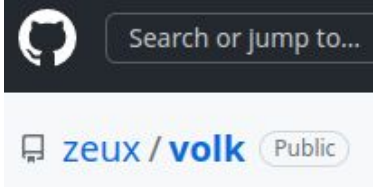


- Set of tests required to pass for all implementations
- Most likely will not use
- Best way to prevent a bug is having a test for it

SDK

Volk

- Meta-Loader for Vulkan
- Used to help reduce the function call overhead
- Only for when Indirectly linking the loader



LUNARG

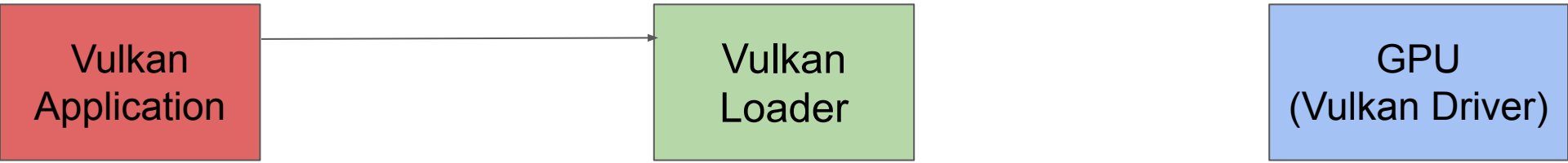
Volk

- Directly linking
  - Compile time
  - Need Loader to link against
- Indirectly linking
  - Runtime
  - Makes calls such dlsym and dlopen
  - What Volk uses

SDK

Volk

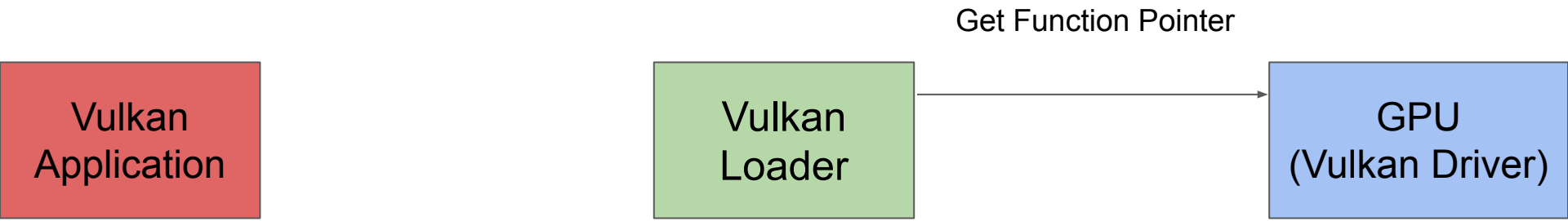
vkGetDeviceProcAddr(device, “vkCmdDraw”)



LUNARG

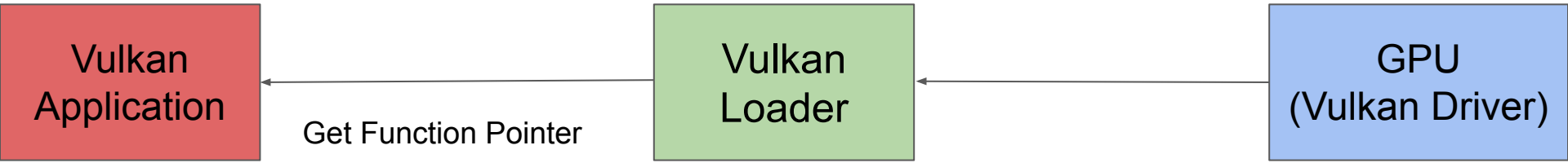
SDK

Volk


Get Function Pointer
Vulkan Application
Vulkan Loader
GPU (Vulkan Driver)

SDK

Volk



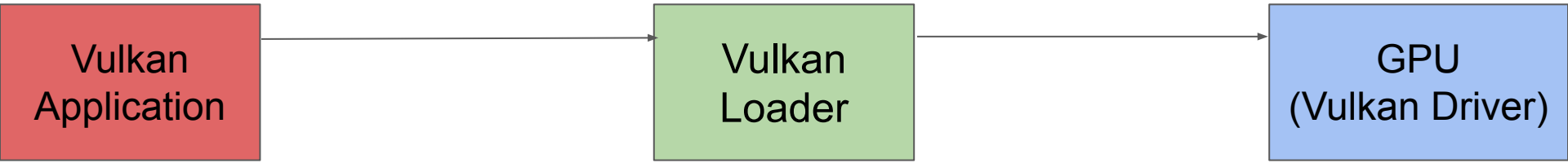
LUNARG

SDK

Volk

(without Volk)

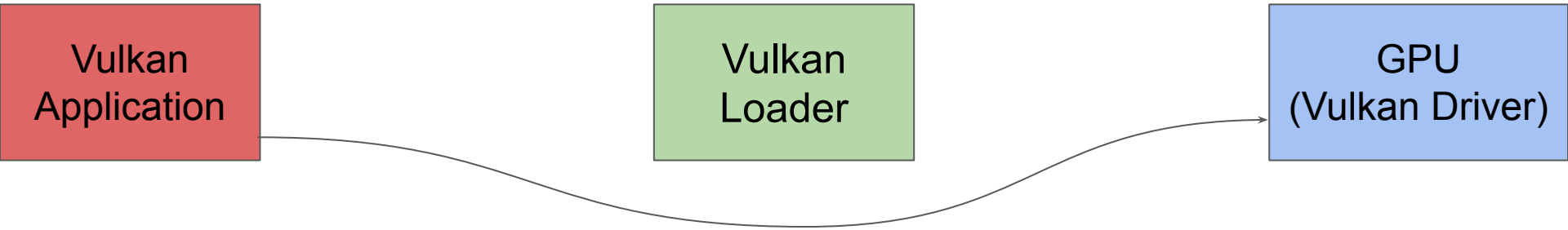
Call vkCmdDraw



LUNARG

SDK

Volk



Use Volk to call vkCmdDraw directly to driver after

LUNARG

SDK

VMA (Vulkan Memory Allocator)

- Library used to manage memory allocation
- Used all over the industry
  - Created and maintained by AMD

SDK

VMA (Vulkan Memory Allocator)

```
VkBufferCreateInfo bufferInfo = { VK_STRUCTURE_TYPE_BUFFER_CREATE_INFO };
bufferInfo.size = 65536;
bufferInfo.usage = VK_BUFFER_USAGE_VERTEX_BUFFER_BIT | VK_BUFFER_USAGE_TRANSFER_DST_BIT;

VmaAllocationCreateInfo allocInfo = {};
allocInfo.usage = VMA_MEMORY_USAGE_AUTO;

VkBuffer buffer;
VmaAllocation allocation;
vmaCreateBuffer(allocator, &bufferInfo, &allocInfo, &buffer, &allocation, nullptr);
```


Questions?