Vulkanised 2024

The 6th Vulkan Developer Conference
Sunnyvale, California | February 5-7, 2024

Vulkan Development for Apple Desktops & Devices

Richard Wright, LunarG

Presentation:

https://bit.ly/3Hngbm9

Overview

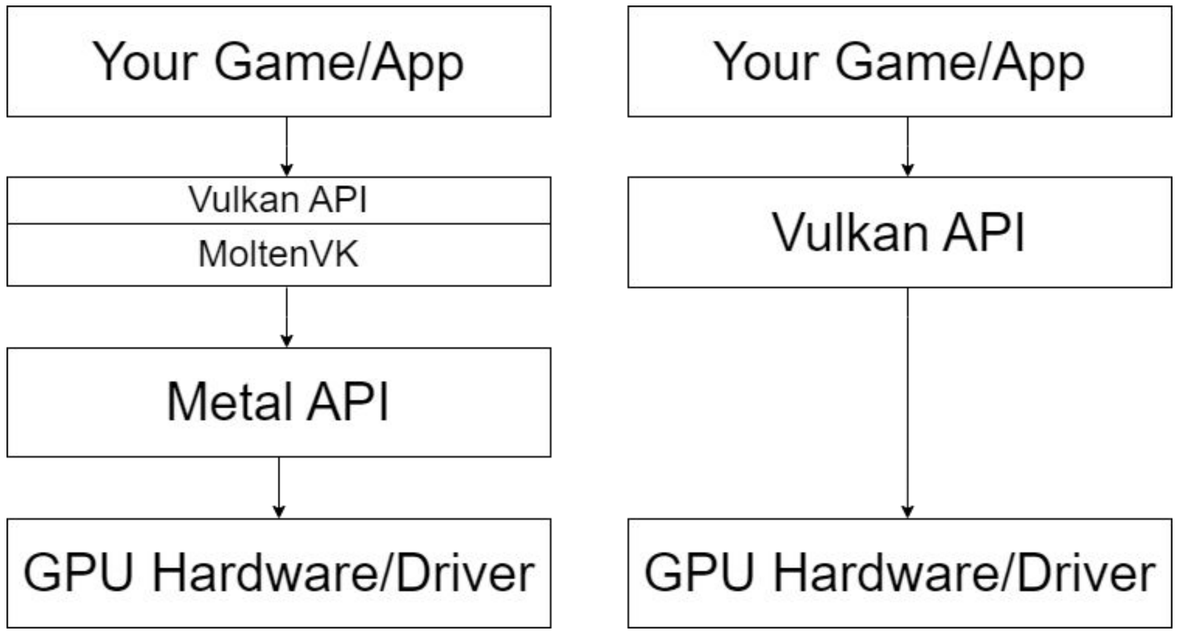
- No native Vulkan “driver” on Apple devices?
- How MoltenVK provides a layered approach to making a Vulkan ICD
- Shipping a “Vulkan” application on Apple OS’s
- Validation Layers and the Vulkan Configurator
- How to use the two “portability extensions”
- Vulkan Loader and Validation on iOS details

Apple does things different

- In the past Apple worked with IHVs (AMD/NVIDIA/Intel) to produce the low-level drivers (OpenGL) for GPU hardware
- The developer-facing API is (now) Metal, a proprietary Apple-only API
- Today OpenGL on Apple is implemented on Metal (much like ANGLE)
- Metal is an, explicit, and thin API… much like Vulkan in some ways
- Simple solution: Write a Vulkan ICD on top of Metal
- Tada - MoltenVK!
- You do not have to learn Metal, you do not have to learn two APIs. MoltenVK is just Vulkan

Vulkan/MoltenVK Layered Approach

Native Vulkan Drivers



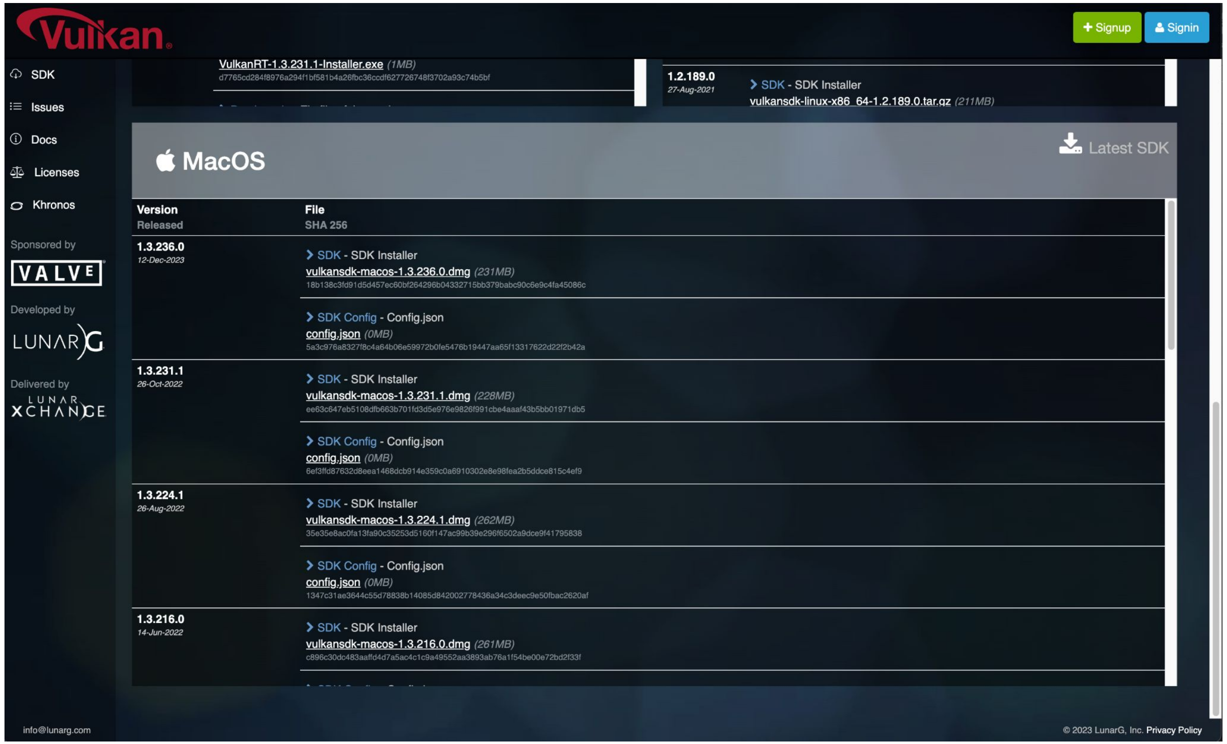
*It’s that simple…

Where do you get this magic library?

Included in the Vulkan SDK available for free at: vulkan.lunarg.com

OR

https://github.com/KhronosGroup/MoltenVK If you like building things yourself



VulkanRT-1.3.231.1-Installer.exe *(1MB)*
d7765cd284f8976a294f1bf581b4a26fbc36ccdf627726748f3702a93c74b5bf

1.2.189.0 *27-Aug-2021* SDK - SDK Installer vulkansdk-linux-x86_64-1.2.189.0.tar.gz *(211MB)*

MacOS

Version / Released	File / SHA 256
1.3.236.0 *12-Dec-2023*	SDK - SDK Installer vulkansdk-macos-1.3.236.0.dmg *(231MB)* 18b138c3fd91d5d457ec60bf264296b04332715bb379babc90c6e9c4fa45086c
	SDK Config - Config.json config.json *(0MB)* 5a3c976a8327f8c4a64b06e59972b0fe5476b19447aa65f13317622d22f2b42a
1.3.231.1 *26-Oct-2022*	SDK - SDK Installer vulkansdk-macos-1.3.231.1.dmg *(228MB)* ee63c647eb5108dfb663b701fd3d5e976e9826f991cbe4aaaf43b5bb01971db5
	SDK Config - Config.json config.json *(0MB)* 6ef3ffd87632d8eea1468dcb914e359c0a6910302e8e98fea2b5ddce815c4ef9
1.3.224.1 *26-Aug-2022*	SDK - SDK Installer vulkansdk-macos-1.3.224.1.dmg *(262MB)* 35e35e8ac0fa13fa90c35253d5160f147ac99b39e296f6502a9dce9f41795838
	SDK Config - Config.json config.json *(0MB)* 1347c31ae3644c55d78838b14085d842002778436a34c3deec9e50fbac2620af
1.3.216.0 *14-Jun-2022*	SDK - SDK Installer vulkansdk-macos-1.3.216.0.dmg *(261MB)* c896c30dc483aaffd4d7a5ac4c1c9a49552aa3893ab76a1f54be00e72bd2f33f

info@lunarg.com

© 2023 LunarG, Inc. Privacy Policy

LUNARG

Packaging and use of MoltenVK

- **System Wide Loader/ICD** (*Development Only*)
    - Useful for development
    - Works seamlessly with the vkconfig and the validation layers
    - The Vulkan SDK will set this up for you
    - **DO NOT SHIP** your applications expecting this
- **Include loader/MoltenVK in your app bundle**

  - Works with the loader, vkconfig, and validation layers
- **Link dynamically, embed in your bundle (in /Frameworks)**

  - Does not work with the loader, vkconfig, or validation layers (or iOS App Store)
- **Link statically***

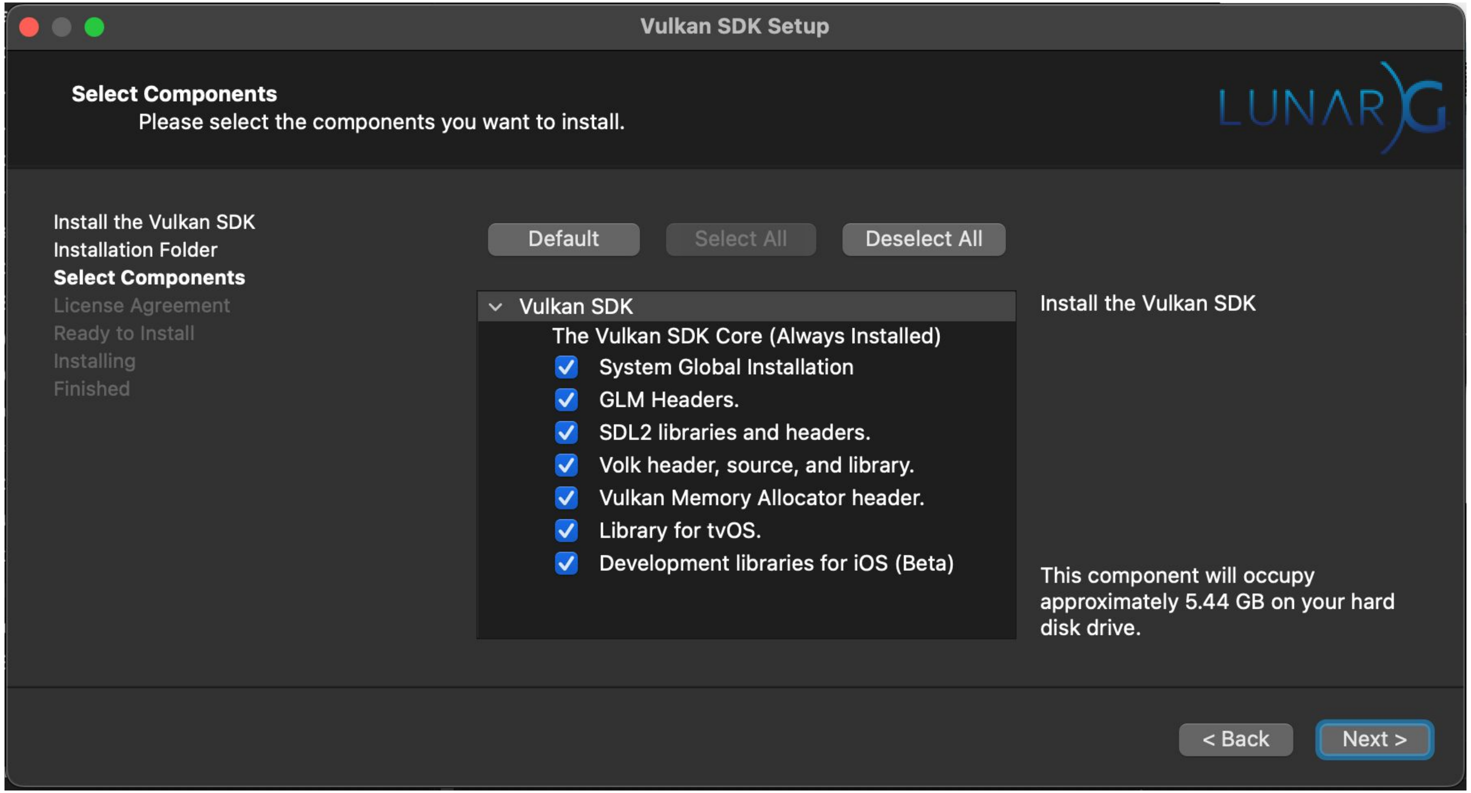
  - Does not work with loader, vkconfig, or validation layers

  - Does allow for non bundled executables to use Vulkan (i.e. command line programs)

  - Does work with all Apple App stores

*Must use this for shipping tvOS applications (for now)

System Wide Loader/ICD



Vulkan SDK Setup

Select Components
Please select the components you want to install.

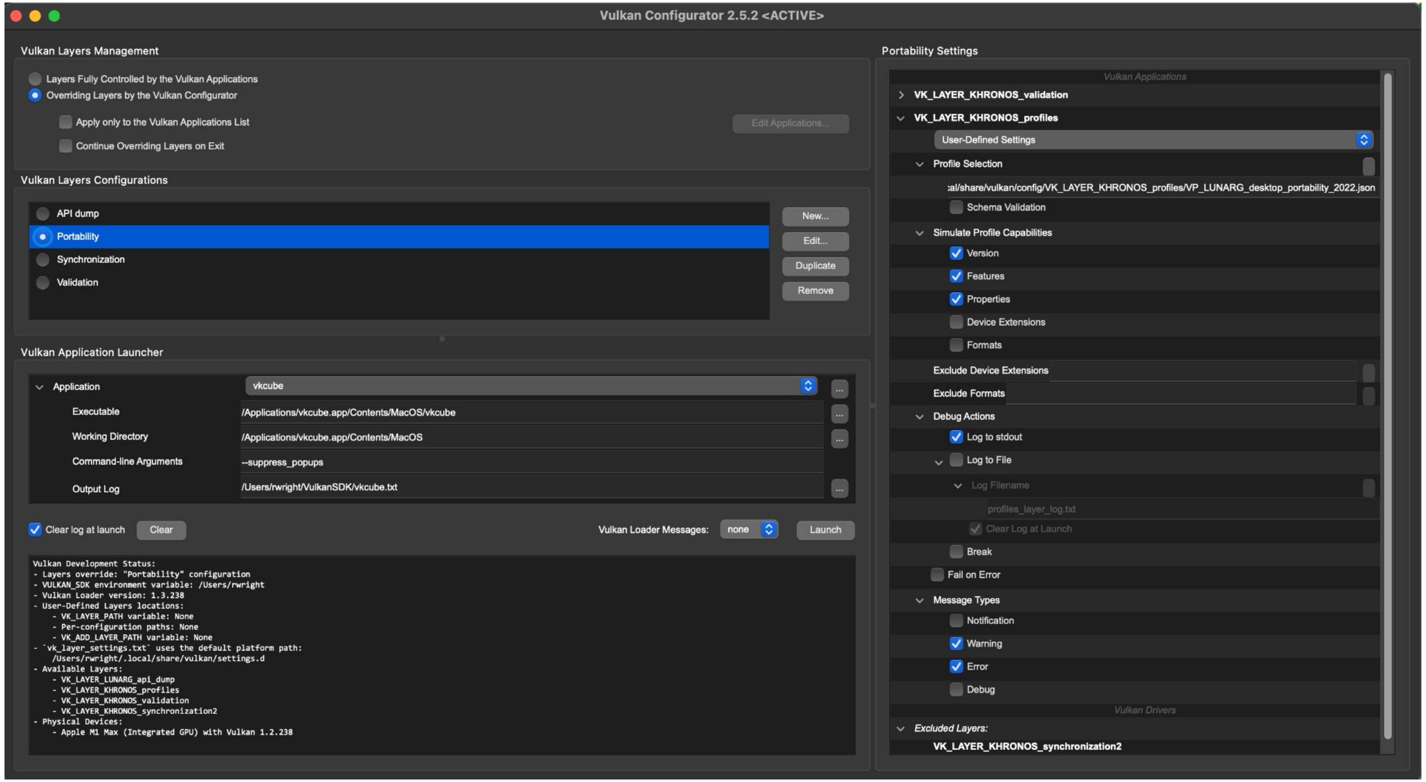
Install the Vulkan SDK
Installation Folder
Select Components
License Agreement
Ready to Install
Installing
Finished

Vulkan SDK
The Vulkan SDK Core (Always Installed)
- ✓ System Global Installation
- ✓ GLM Headers.
- ✓ SDL2 libraries and headers.
- ✓ Volk header, source, and library.
- ✓ Vulkan Memory Allocator header.
- ✓ Library for tvOS.
- ✓ Development libraries for iOS (Beta)

Install the Vulkan SDK

This component will occupy approximately 5.44 GB on your hard disk drive.

Vulkan Configurator “Just Works*”



Vulkan Configurator 2.5.2 <ACTIVE>

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

Vulkan Layers Configurations

- ○ API dump
- ● Portability
- ○ Synchronization
- ○ Validation

Vulkan Application Launcher

Application	vkcube
Executable	/Applications/vkcube.app/Contents/MacOS/vkcube
Working Directory	/Applications/vkcube.app/Contents/MacOS
Command-line Arguments	--suppress_popups
Output Log	/Users/rwright/VulkanSDK/vkcube.txt

✓ Clear log at launch — Vulkan Loader Messages: none

```
Vulkan Development Status:
- Layers override: "Portability" configuration
- VULKAN_SDK environment variable: /Users/rwright
- Vulkan Loader version: 1.3.238
- User-Defined Layers locations:
    - VK_LAYER_PATH variable: None
    - Per-configuration paths: None
    - VK_ADD_LAYER_PATH variable: None
- `vk_layer_settings.txt` uses the default platform path:
    /Users/rwright/.local/share/vulkan/settings.d
- Available Layers:
    - VK_LAYER_LUNARG_api_dump
    - VK_LAYER_KHRONOS_profiles
    - VK_LAYER_KHRONOS_validation
    - VK_LAYER_KHRONOS_synchronization2
- Physical Devices:
    - Apple M1 Max (Integrated GPU) with Vulkan 1.2.238
```

Portability Settings

Vulkan Applications

- VK_LAYER_KHRONOS_validation
- VK_LAYER_KHRONOS_profiles
  - User-Defined Settings
  - Profile Selection
    - al/share/vulkan/config/VK_LAYER_KHRONOS_profiles/VP_LUNARG_desktop_portability_2022.json
    - ☐ Schema Validation
  - Simulate Profile Capabilities
    - ✓ Version
    - ✓ Features
    - ✓ Properties
    - ☐ Device Extensions
    - ☐ Formats
  - Exclude Device Extensions
  - Exclude Formats
  - Debug Actions
    - ✓ Log to stdout
    - ☐ Log to File
      - Log Filename
        - profiles_layer_log.txt
      - ✓ Clear Log at Launch
    - ☐ Break
  - ☐ Fail on Error
  - Message Types
    - ☐ Notification
    - ✓ Warning
    - ✓ Error
    - ☐ Debug

Vulkan Drivers

- *Excluded Layers:*
  - VK_LAYER_KHRONOS_synchronization2

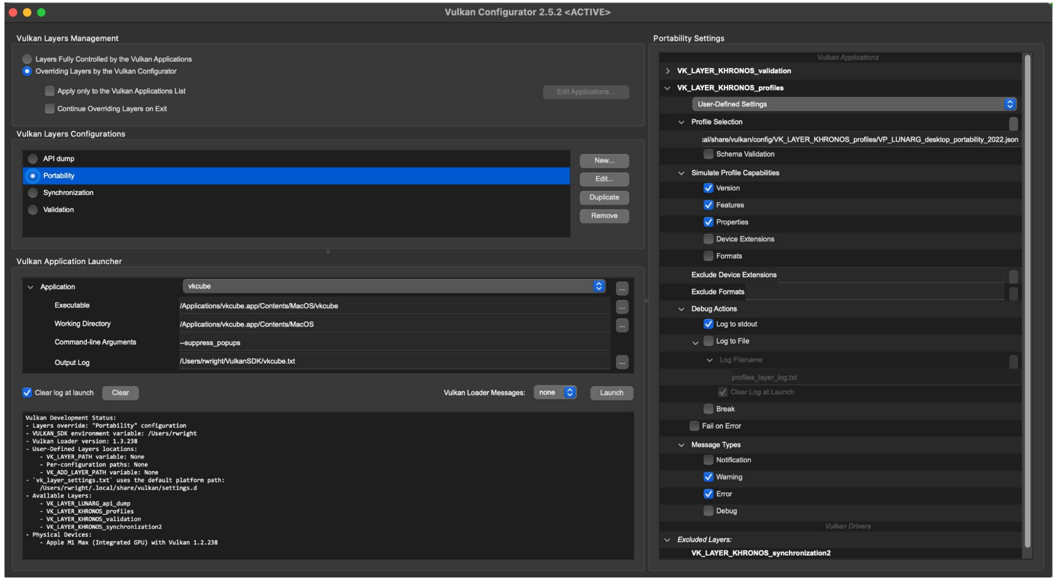
*macOS Desktop Only

Vulkan Configurator "*Just Works"

Bugs you know about

Bugs you DON'T know about

API Usage Bugs



Vulkan Configurator 2.5.2 <ACTIVE>

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

Vulkan Layers Configurations

- ○ API dump
- ● Portability
- ○ Synchronization
- ○ Validation

Vulkan Application Launcher

Application	vkcube
Executable	/Applications/vkcube.app/Contents/MacOS/vkcube
Working Directory	/Applications/vkcube.app/Contents/MacOS
Command-line Arguments	--suppress_popups
Output Log	/Users/rwright/VulkanSDK/vkcube.txt

☑ Clear log at launch

Vulkan Loader Messages: none

```
Vulkan Development Status:
- Layers override: "Portability" configuration
- VULKAN_SDK environment variable: /Users/rwright
- Vulkan Loader version: 1.3.238
- User-Defined Layers locations:
    - VK_LAYER_PATH variable: None
    - Per-configuration paths: None
    - VK_ADD_LAYER_PATH variable: None
- `vk_layer_settings.txt` uses the default platform path:
    /Users/rwright/.local/share/vulkan/settings.d
- Available Layers:
    - VK_LAYER_LUNARG_api_dump
    - VK_LAYER_KHRONOS_profiles
    - VK_LAYER_KHRONOS_validation
    - VK_LAYER_KHRONOS_synchronization2
- Physical Devices:
    - Apple M1 Max (Integrated GPU) with Vulkan 1.2.238
```

Portability Settings

Vulkan Applications

- VK_LAYER_KHRONOS_validation
- VK_LAYER_KHRONOS_profiles
  - User-Defined Settings
  - Profile Selection
    - :al/share/vulkan/config/VK_LAYER_KHRONOS_profiles/VP_LUNARG_desktop_portability_2022.json
    - ☐ Schema Validation
  - Simulate Profile Capabilities
    - ☑ Version
    - ☑ Features
    - ☑ Properties
    - ☐ Device Extensions
    - ☐ Formats
  - Exclude Device Extensions
  - Exclude Formats
  - Debug Actions
    - ☑ Log to stdout
    - ☐ Log to File
      - Log Filename
        - profiles_layer_log.txt
      - ☑ Clear Log at Launch
    - ☐ Break
  - ☐ Fail on Error
  - Message Types
    - ☐ Notification
    - ☑ Warning
    - ☑ Error
    - ☐ Debug

Vulkan Drivers

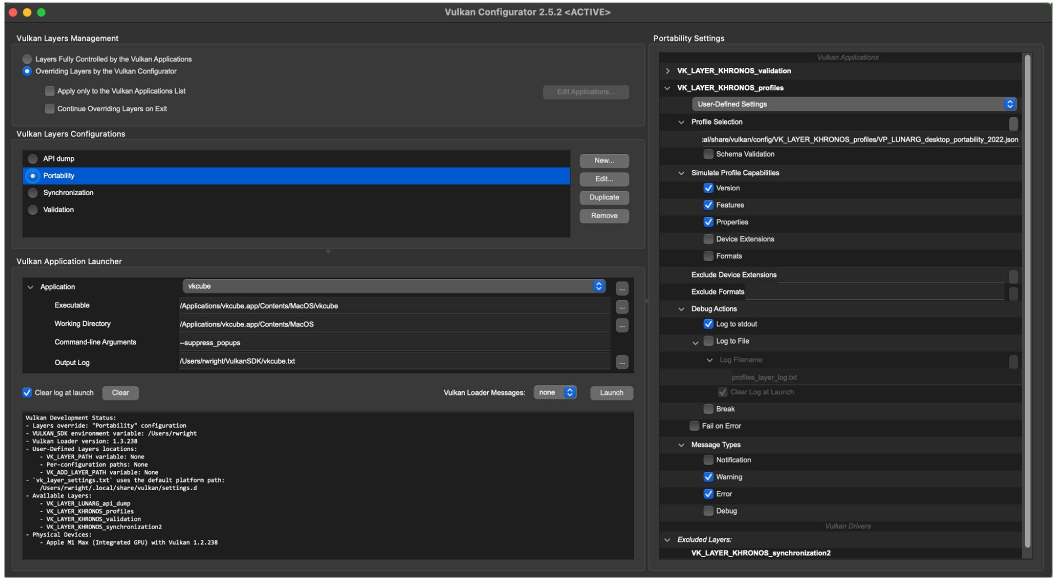
Excluded Layers:

- VK_LAYER_KHRONOS_synchronization2

*macOS Desktop Only

Vulkan Layers on macOS

- Khronos Validation
  - No DebugPrintf
  - No GPU/AV
- Khronos Synchronization2
- Shader Objects Extension
- Khronos Profiles
- API Dump
- Screenshot (new to macOS)
- GFXReconstruct (coming soon)



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

Vulkan Layers Configurations
- API dump
- [✓ Portability]
- Synchronization
- Validation

Application	vkcube
Executable	/Applications/vkcube.app/Contents/MacOS/vkcube
Working Directory	/Applications/vkcube.app/Contents/MacOS
Command-line Arguments	--suppress_popups
Output Log	/Users/rwright/VulkanSDK/vkcube.txt

✓ Clear log at launch — Vulkan Loader Messages: none

```
Vulkan Development Status:
- Layers override: "Portability" configuration
- VULKAN_SDK environment variable: /Users/rwright
- Vulkan Loader version: 1.3.238
- User-Defined Layers locations:
    - VK_LAYER_PATH variable: None
    - Per-configuration paths: None
    - VK_ADD_LAYER_PATH variable: None
- `vk_layer_settings.txt` uses the default platform path:
    /Users/rwright/.local/share/vulkan/settings.d
- Available Layers:
    - VK_LAYER_LUNARG_api_dump
    - VK_LAYER_KHRONOS_profiles
    - VK_LAYER_KHRONOS_validation
    - VK_LAYER_KHRONOS_synchronization2
- Physical Devices:
    - Apple M1 Max (Integrated GPU) with Vulkan 1.2.238
```

Portability Settings
- VK_LAYER_KHRONOS_validation
- VK_LAYER_KHRONOS_profiles
  - User-Defined Settings
  - Profile Selection
    - :al/share/vulkan/config/VK_LAYER_KHRONOS_profiles/VP_LUNARG_desktop_portability_2022.json
    - ☐ Schema Validation
  - Simulate Profile Capabilities
    - ✓ Version
    - ✓ Features
    - ✓ Properties
    - ☐ Device Extensions
    - ☐ Formats
  - Exclude Device Extensions
  - Exclude Formats
  - Debug Actions
    - ✓ Log to stdout
    - ☐ Log to File
      - Log Filename
        - profiles_layer_log.txt
      - ✓ Clear Log at Launch
    - ☐ Break
  - ☐ Fail on Error
  - Message Types
    - ☐ Notification
    - ✓ Warning
    - ✓ Error
    - ☐ Debug
- *Excluded Layers:*
  - VK_LAYER_KHRONOS_synchronization2

Bundled Loader and Layers on macOS

```
VulkanRocks.app
    /Contents
        /Frameworks
            libMoltenVK.dylib
            libvulkan.1.[version number].dylib
            libvulkan.1.dylib -> libvulkan.1.[version number].dylib
            libVkLayer_api_dump.dylib
        /MacOS
            VulkanRocks
        /Resources
            /vulkan
                /icd.d
                    MoltenVK_icd.json

               /explicit_layer.d
                    VkLayer_api_dump.json
```

https://vulkan.lunarg.com/doc/sdk/latest/mac/getting_started.html

Bundled Loader and Layers on iOS

(development only)

```
VulkanRocks.app
        /Frameworks
            libMoltenVK.dylib
            libvulkan.1.[version number].dylib
            libvulkan.1.dylib -> libvulkan.1.[version number].dylib
            libVkLayer_api_dump.dylib

        VulkanRocks
        vk_layer_settings.txt

        /vulkan
           /icd.d
             MoltenVK_icd.json
           /explicit_layer.d
              VkLayer_api_dump.json
```

Include a Dynamic Library (very common today)

- MoltenVK as a dynamic library can be placed in /Frameworks in the app bundle
- MoltenVK has all the loader entry points, so it can “fake” the loader, but it doesn’t actually load layers, etc.
- Works on all Apple Platforms, but not allowed on iOS app store
- Remember: this bypasses the loader - no layers!
- VK_EXT_metal_objects -> Use this for Vulkan-Metal interoperability

Static Link

- MoltenVK can also be linked to your app as a static library.
- Include the MoltenVK.xcframework
- This contains static libraries for each platform

  macOS

  iOS/Simulator

  tvOS/Simulator
- Great option for shipping applications - especially non-bundled apps
  - Works on all Apple devices.
  - Cannot use any layers (validation or otherwise)
  - Use Loader/Layers for development and static for shipping on iOS

- VK_EXT_metal_objects -> Use this for Vulkan-Metal interoperability

Okay, that’s the overview of linking and packaging…

What about the code?

There are two important extensions you need to know about if you are going to target Apple devices… in fact, this goes for ANY layered Vulkan implementation on ANY platform.

`VK_KHR_portability_enumeration`

`VK_KHR_portability_subset`

VK_KHR_portability_enumeration

The purpose of this extension is to keep games/apps from “accidentally” selecting an incomplete (but Portability Compliant) Vulkan Implementation*. While important today on macOS, it may be more important soon on Windows and Linux.

*This does require that a layered, Portability Conformant Vulkan implementation must identify itself to be so by supporting this extension.

VK_KHR_portability_enumeration

This is an instance extension. You are telling the Loader what devices you want to see.

1. If “VK_KHR_portability_enumeration” is listed by vkEnumerateInstanceExtensionProperties, it means you have a (newish) loader that supports this extension. You must add the extension name to the ppEnableExtensions list in the VkInstanceCreateInfo structure if you want to make use of a portability implementation.
2. You must also add the VK_INSTANCE_CREATE_ENUMERATE_PORTABILITY_BIT_KHR flag to the flags member.

If you do not do BOTH of the above (on macOS currently), you will get VK_ERROR_INCOMPATIBLE_DRIVER from vkCreateInstance

VK_KHR_portability_enumeration

Important: If multiple drivers are found, and one is “portable,” and you’ve not enabled this extension, you will only see the fully conformant hardware driver.

This will likely happen on Windows/Linux before it happens on macOS!

VK_KHR_portability_enumeration

```
////////////////////////////////////////////////////////////////////////////
// Get the list of instance extensions
uint32_t extensionCount = 0;
vkEnumerateInstanceExtensionProperties(nullptr, &extensionCount, nullptr);


std::vector<VkExtensionProperties> extensions(extensionCount);
vkEnumerateInstanceExtensionProperties(nullptr, &extensionCount, extensions.data());
```

Look for the extensions you want

```
std::vector<const char *> extNames;
bool bPortableEnumeration = false;
for (uint32_t i = 0; i < extensionCount; i++) {

    // If the extension is present, you must use it to get portable implementations
    if(!strcmp(extensions[i].extensionName, VK_KHR_PORTABILITY_ENUMERATION_EXTENSION_NAME))
        {
        bPortableEnumeration = true;
        extNames.push_back(VK_KHR_PORTABILITY_ENUMERATION_EXTENSION_NAME);
        }

    ...
    ...
    }
```

Create the Vulkan Loader Instance

```
VkInstanceCreateInfo inst_info = {};
inst_info.sType = VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO;
inst_info.pNext = NULL;
inst_info.pApplicationInfo = &appInfo;
inst_info.enabledLayerCount = 0;
inst_info.ppEnabledLayerNames = nullptr;
inst_info.enabledExtensionCount = (int)extNames.size();
inst_info.ppEnabledExtensionNames = extNames.data();

if(bPortableEnumeration)
    inst_info.flags |= VK_INSTANCE_CREATE_ENUMERATE_PORTABILITY_BIT_KHR;

// Create the Instance
lastResult = vkCreateInstance(&inst_info, NULL, &vulkanInstance);
```

Create the Vulkan Loader Instance

```
// Create the Instance
lastResult = vkCreateInstance(&inst_info, NULL, &vulkanInstance);
```

Forget one of these two things? With SDK/Loader 1.3.216 or later, you will get the dreaded:

```
lastResult == VK_ERROR_INCOMPATIBLE_DRIVER
```

So, now you’ve told the loader you are interested in a “Portability conformant” driver. You got one.

Now what?

VK_KHR_portability_subset

A layered implementation of Vulkan may have some gaps in it’s capabilities. This extension gives you the ability to query for missing features so you can work around them, or simply punt and tell the user you cannot run using this hardware device.

Version 1.0 (provisional*) of this extension lists a specific set of features that may or may not be present… we’ll get to those soon.

*VK_KHR_portability_subset_metal is coming soon

VK_KHR_portability_subset

This is a **device** extension.

vkEnumerateDeviceExtensionProperties will list “VK_KHR_portability_subset”

Yep, add it to the ppEnabledExtensionNames member of VkDeviceCreateInfo.

VK_KHR_portability_subset

```
// We have a physical device, now we need a list of it's extensions
uint32_t deviceExtensionCount;
vkEnumerateDeviceExtensionProperties(physicalDevice, nullptr, &deviceExtensionCount, nullptr);


std::vector<VkExtensionProperties> deviceExtensions(deviceExtensionCount);
vkEnumerateDeviceExtensionProperties(physicalDevice, nullptr, &deviceExtensionCount,
                                                                   deviceExtensions.data());
std::vector<const char *> extNamesDevice;

for (uint32_t i = 0; i < deviceExtensionCount; i++){
    if(strcmp(deviceExtensions[i].extensionName, "VK_KHR_portability_subset") == 0)
            extNamesDevice.push_back(deviceExtensions[i].extensionName)

      . . .
    }
```

VK_KHR_portability_subset

Query for what features are available/missing

```
VkPhysicalDevicePortabilitySubsetFeaturesKHR portabilityFeatures = {};

portabilityFeatures.sType =
                VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_PORTABILITY_SUBSET_FEATURES_KHR
;

VkPhysicalDeviceFeatures2 physicalDeviceFeatures2 = {};
physicalDeviceFeatures2.sType =
VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_FEATURES_2;
physicalDeviceFeatures2.pNext = & portabilityFeatures;
vkGetPhysicalDeviceFeatures2(physicalDevice, &physicalDeviceFeatures2);
```

Note vkGetPhysicalDeviceFeatures2 is an extension prior to Vulkan 1.1

The structure is basically a set of flags…

```
typedef struct VkPhysicalDevicePortabilitySubsetFeaturesKHR {
    VkStructureType    sType;
    void*              pNext;
    VkBool32           constantAlphaColorBlendFactors;
// 1
    VkBool32           events;                                    // 1
    VkBool32           imageViewFormatReinterpretation;           // 0
    VkBool32           imageViewFormatSwizzle;                    // 1
    VkBool32           imageView2DOn3DImage;                      // 1
    VkBool32           multisampleArrayImage;                     // 1
    VkBool32           mutableComparisonSamplers;
// 1
    VkBool32           pointPolygons;                             // 0
    VkBool32           samplerMipLodBias;                         // 0
    VkBool32           separateStencilMaskRef;                    // 1
    VkBool32           shaderSampleRateInterpolationFunctions;    // 1
    VkBool32           tessellationIsolines;                      // 0
    VkBool32           tessellationPointMode;                     // 0
    VkBool32           triangleFans;                              // 0
    VkBool32           vertexAttributeAccessBeyondStride;         // 1
} VkPhysicalDevicePortabilitySubsetFeaturesKHR;
```

Values (old) on my M1 Mac (might be different on other Macs/GPUs)

Zero means the feature is not present on this device

THESE ARE “SUBJECT” TO CHANGE!!

AS IN “LIKELY”...
(e.g. triangle fans were added recently)

You must enable the ones you want!

```
VkDeviceCreateInfo createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO;

physicalDeviceFeatures2.pNext = &portabilityFeatures

createInfo.pNext = physicalDeviceFeatures2;


logicalDevice = VK_NULL_HANDLE;
VkResult result = vkCreateDevice(physicalDevice, &createInfo, nullptr, &logicalDevice);

if (result != VK_SUCCESS)
    return false;
```

iOS Layer Notes

Loader and Layers work as of January 2024 SDK

- Only Explicit Layers - must be enabled in source code
- No vkConfig for devices, you have to include the layer settings file
- OR use the new VK_EXT_layer_settings extension!

Validation layer output goes to stdout, which is captured by XCode

iOS Appstore does not allow .dylibs - Frameworks coming in next SDK release

CMake support also coming soon for finding iOS versions of SDK components

Turning on an explicit layer in code

```
const std::vector<const char*> layerList = { "VK_LAYER_LUNARG_api_dump" };

… <your stuff>

VkInstanceCreateInfo inst_info = { VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO };

… <other stuff>

inst_info.enabledLayerCount = (uint32_t)layerList.size();
inst_info.ppEnabledLayerNames = layerList.data();

… <yet more stuff>

result = vkCreateInstance(&inst_info, NULL, &vulkanInstance);
```

Make sure results != VK_ERROR_LAYER_NOT_PRESENT

VK_KHR_layer_settings

```
const char* name = "VK_LAYER_KHRONOS_validation";
const VkBool32 setting_validate_core = VK_TRUE;

const VkLayerSettingEXT settings[] = {                                 <- Array of settings
    {name, "validate_core", VK_LAYER_SETTING_TYPE_BOOL32_EXT,
     1, &setting_validate_core}};

const VkLayerSettingsCreateInfoEXT layer_settings_create_info = {   <- Settings structure
    VK_STRUCTURE_TYPE_LAYER_SETTINGS_CREATE_INFO_EXT, nullptr,
    static_cast<uint32_t>(std::size(settings)), settings};

inst_info.pNext = &layer_settings_create_info;
result = vkCreateInstance(&inst_info, NULL, &vulkanInstance);
```

https://www.lunarg.com/wp-content/uploads/2024/01/Configuring-Vulkan-Layers-LunarG-Christophe-Riccio-01-16-2024.pdf

Conclusion

- MoltenVK is a “Layered Vulkan Implementation”
- Work around missing extensions and features like any other platform
- Portability extensions (two of them) are there to help navigate this
- Performance is very good
- Loader and layer support on iOS (Beta)
- Next SDK - Full Frameworks and compatibility with App Store
- Try it, you’ll like it!

Help Us Improve the Vulkan SDK and Ecosystem

Share Your Feedback

Take the LunarG annual developer's survey

https://www.surveymonkey.com/r/KTBZDCM

- Survey results are tabulated
- Shared with the Vulkan Working Group
- Actions are assigned
- Results are reported

Survey closes February 26, 2024

Today's Presentation:

https://bit.ly/3Hngbm9

Get A FREE Tumbler at the LunarG Sponsor Table!

Thank you!

QUESTIONS?