Vulkan Ecosystem and Vulkan Architecture

Jian
Who am I...

- Jian Tang
  - Current Master student at Oregon State University
  - Video game (MOG) developer 3 years
  - Integrated circuit engineer 1 year
A deep dive into Vulkan
What roles do they play?
Goals

• To know the handshakes in the Vulkan ecosystem.
• To understand how Vulkan has been created from the implementor’s view of Vulkan, pretending to be a LunarG graphics driver engineer.
• To see the principle for creating a “GLEW” or “GL3W” for Vulkan.
• To strengthen your debug ability in Vulkan.
• To make high performance Vulkan programming possible.
Hardware Related Levels

- Vulkaneers
- Vulkan Application
- Vendors
- Vulkan ICDs
- Vendors
- Vulkan Physical Device
Install Vulkan ICDs (Installable Client Driver)

- Nvidia ICD on Win32
Properly-Installed ICDs

Vulkan ICD associated JSON Manifest file

On 64-bit Windows
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Khronos\Vulkan\Drivers

```json
{
    "file_format_version": "1.0.0",
    "ICD": {
        "library_path": ".\nvogl64.dll",
        "api_version": "1.0.65"
    },
    "layer": {
        "name": "VK_LAYER_NV_optimus",
        "type": "INSTANCE",
        "library_path": ".\nvogl64.dll",
        "api_version": "1.0.65",
        "implementation_version": "1",
        "description": "NVIDIA Optimus layer",
        "functions": {
            "vkGetInstanceProcAddr": "vk_optimusGetInstanceProcAddr",
            "vkGetDeviceProcAddr": "vk_optimusGetDeviceProcAddr"
        },
        "disable_environment": {
            "DISABLE_LAYER_NV_OPTIMUS_1": ""
        }
    }
}
```

On Linux

```
/usr/local/etc/vulkan/icd.d
/usr/local/share/vulkan/icd.d
/etc/vulkan/icd.d
/usr/share/vulkan/icd.d
$HOME/.local/share/vulkan/icd.d
```

Including multiple functionality
There are several ways to use the Vulkan API

- 1. Dynamically load the ICD library that provides Vulkan API implementation into our application.
- 2. ?
- 3. ?

vk_icdNegotiateLoaderICDInterface
vk_icdGetInstanceProcAddr
vk_icdGetPhysicalDeviceProcAddr

- Names can be overridden in Manifest file
There are several ways to use the Vulkan API

No LunarG so far!

Q. Why can I run Vulkan programs, even though I have not installed LunarG SDK yet?
• Where should the LunarG stuff be?
• Why LunarG stuff?
• DEBUG!
Vulkan Ecosystem

That’s it?
May be an additional middleware?
There are several ways to use the Vulkan API

1. Dynamically load the ICD library that provides Vulkan API implementation into our application.
2. Directly link the exports.
3. Need to call the trampoline functions that jump to the appropriate dispatch table entry.
There are several ways to use the Vulkan API

• 1. Dynamically load the ICD library that provides Vulkan API implementation into our application.

• 2. Directly link the exports.

• 3. Dynamically load the Vulkan Runtime library.

• Part of driver installation
Global-Level Functions

- **GetProcAddress() // Win32**
- **dlsym() // Linux**

```
vkGetInstanceProcAddr(nullptr, ...) // platform-independent
```

```
vkCreateInstance
vkEnumerateInstanceExtensionProperties
vkEnumerateInstanceLayerProperties
```

**Acquiring Pointers**

- **vkGetInstanceProcAddr**
  - `nullptr, ...`

**then, platform-independent**

```
vkCreateInstance
```

**VkInstance**

**Acquiring Pointers**
Global-Level Functions

- `vkCreateInstance`
- `vkEnumerateInstanceExtensionProperties`
- `vkEnumerateInstanceLayerProperties`

- Work with Extensions & Layers
- Create VkInstance for the next level
Instance-Level Functions

vkGetInstanceProcAddr(
  VkInstance instance,
  ...
)

Acquiring Pointers

Some instance level functions:

vkEnumeratePhysicalDevices
vkGetPhysicalDeviceProperties
vkGetPhysicalDeviceFeatures
vkGetPhysicalDeviceQueueFamilyProperties
vkCreateDevice
vkGetDeviceProcAddr
vkDestroyInstance
...

"logical devices" (simply called devices)

VkInstance

vkCreateDevice

VkDevice
Instance-Level Functions

Some instance level functions:

- vkEnumeratePhysicalDevices
- vkGetPhysicalDeviceProperties
- vkGetPhysicalDeviceFeatures
- vkGetPhysicalDeviceQueueFamilyProperties
- vkCreateDevice
- vkGetDeviceProcAddr
- vkDestroyInstance
...

- Work with VkInstance, ICDs, & VkPhysicalDevice
- Create VkDevice for the next level
Device-Level Functions

```
vkGetDeviceProcAddr(VkDevice device, ...)
```

Acquiring Pointers

Some Device level functions:
- vkGetDeviceQueue
- vkDestroyDevice
- vkDeviceWaitIdle
- ...

Q. What if `vkGetInstanceProcAddr` to acquire the pointers for the device-level functions?
vkGetDeviceProcAddr for vkQueueSubmit

on Physical Device A (Nvidia)

Call it with Physical Device B (Intel)
Device-Level Functions

Some Device level functions:

- vkGetDeviceQueue
- vkDestroyDevice
- vkDeviceWaitIdle
- ...

- Logical accessor to a particular Physical Device
- Work with VkDevice, VkQueue, VkCmdBuffer and other derived objects
Trampolines and Terminators

- entrypoint
- call-chain
  - Instance chain generated during vkCreateInstance
  - Device chain generated during vkCreateDevice
- multiple ICDs
Instance functions are broadcasted to every available ICD on the system.

Example.

`vkEnumeratePhysicalDevices`
Call Chains

Device Call Chain using `vkGetInstanceProcAddr`:

A loader-exported Vulkan Device function:

Device Call Chain using `vkGetDeviceProcAddr`:
https://github.com/KhronosGroup/Vulkan-LoaderAndValidationLayers
Thanks!

Questions
Reference

[1] LunarG Presentation for Vulkan