Vulkan Guide

Khronos Group

Lina Liu Nov. 22

Logistics Overview

What is Vulkan?

1. Vulkan is a new generation graphics and compute API
2. Vulkan provides high-efficiency, cross-platform to modern GPUs used in a devices like pc/mobile/ embeded platforms
3. Vulkan provide a way for developers to program their modern GPU hardware
    1. Vulkan is a tool for developers to create hardware accelerated applications
4. The Khronos Group is created and maintains Vulkan.

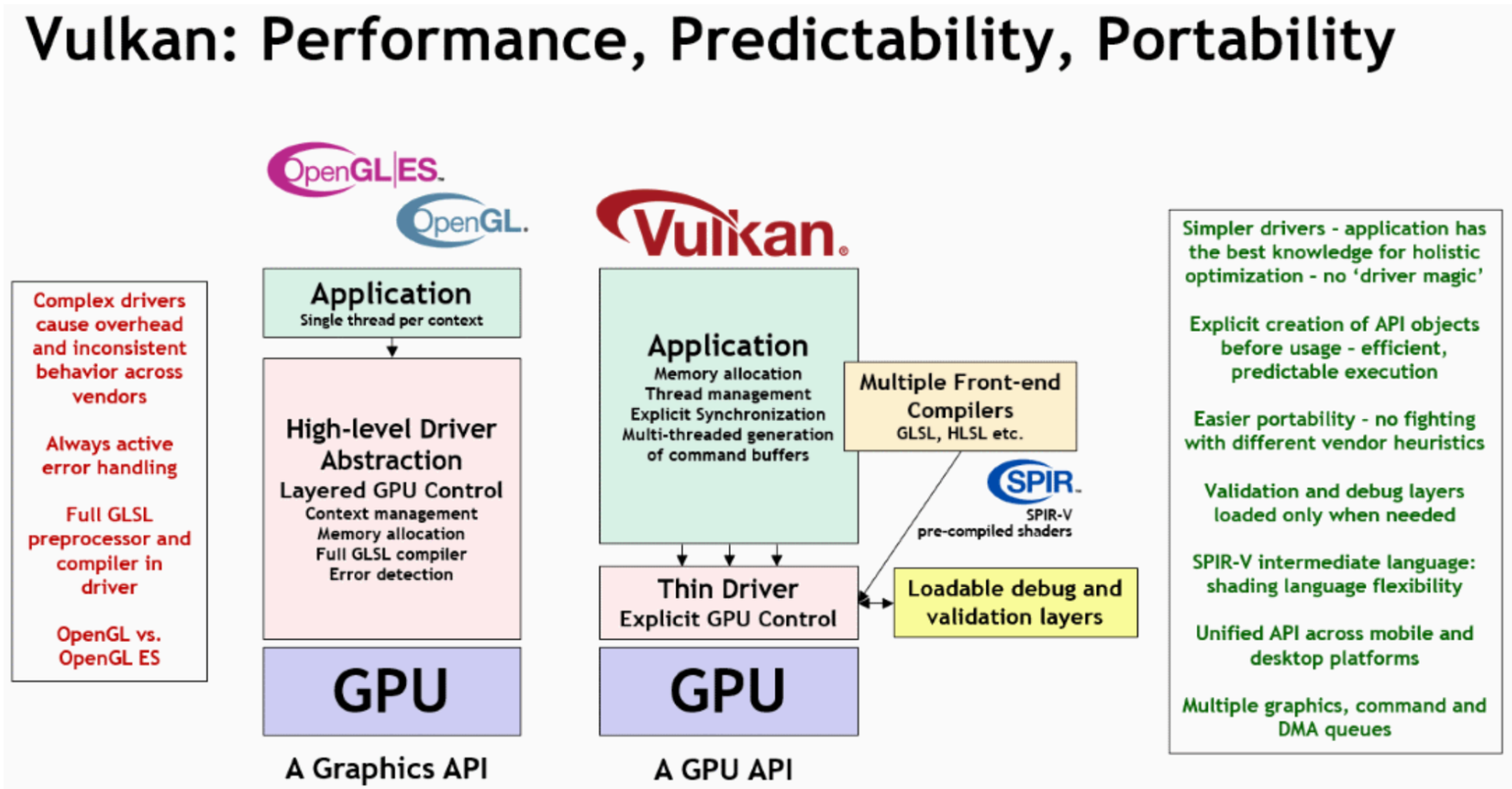
Vulkan and OpenGL

Difference

1. OpenGL is also a 3D Graphics API
2. Vulkan is not a replacement for OpenGL
3. Vulkan is an explicit API allows for more explicit control of the GPU

Feature	OpenGL ES	Vulkan
State management	Global state	State objects
API execution model	Synchronous	Asynchronous
API threading model	Single threaded	Multi-threaded
API error checking	Extensive runtime checks	Only via layers
Render pass abstraction	Inferred render passes	Explicit render passes
Memory allocation	Client-server pools	Shared memory pool
Memory usage	Typed allocations	Typed views

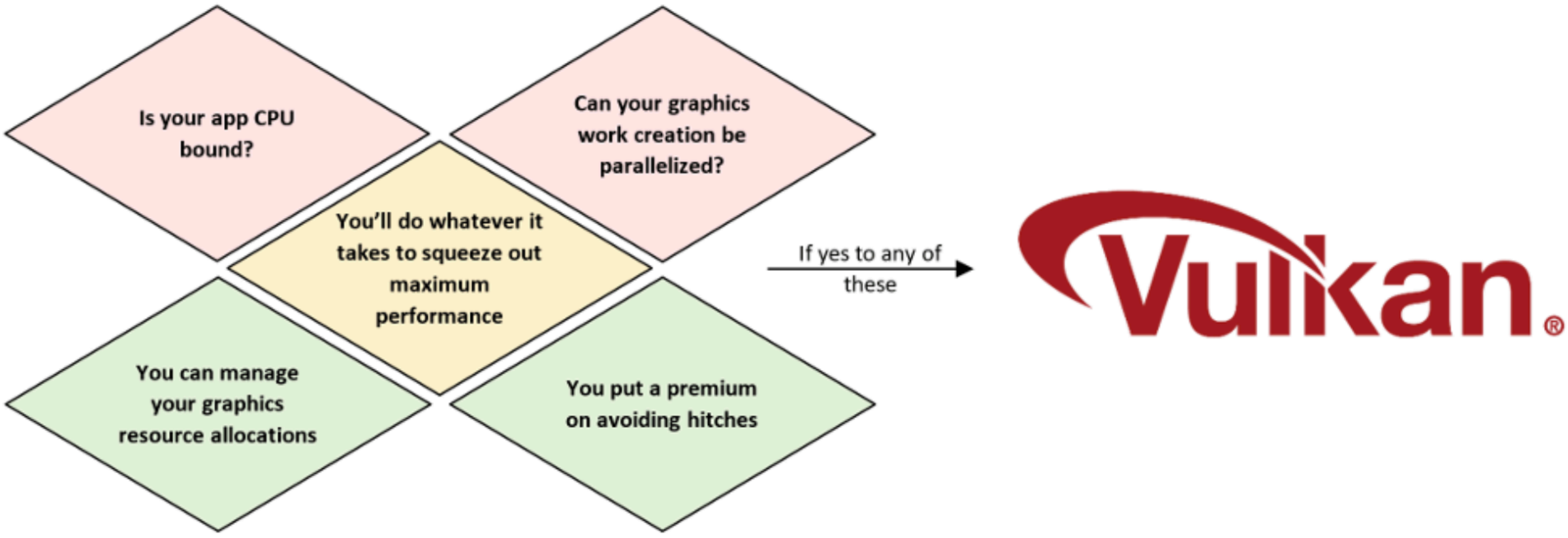
Vulkan and OpenGL



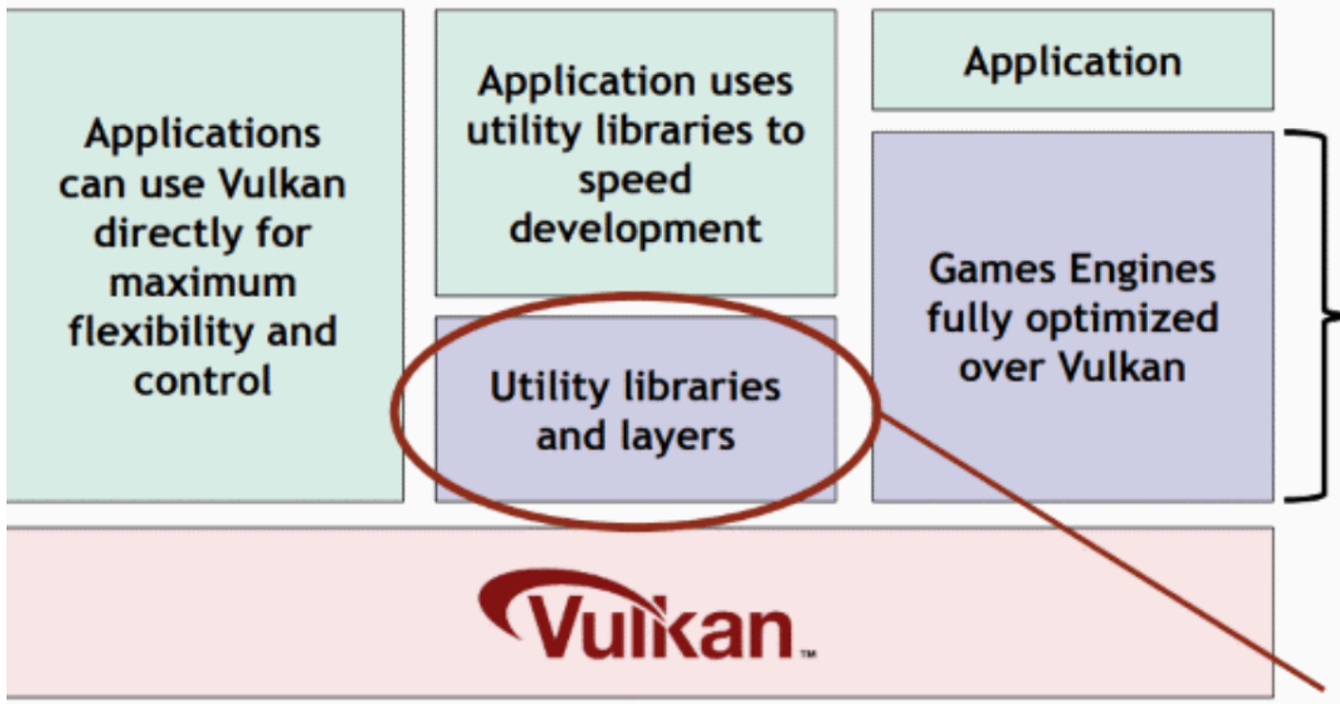
Vulkan and OpenGL

Vulkan puts more work and responsibility into application

1. Why Vulkan put more work and responsibility into application?
    1. For those who use Vulkan correctly can find power and performance improvements



The Power of a Three Layer Ecosystem



The industry's leading games and engine vendors are participating in the Vulkan working group

OXIDE · LUCASFILM Ltd · BLIZZARD ENTERTAINMENT · EA · unity · EPIC GAMES · VALVE

Developers can choose at which level to use the Vulkan Ecosystem

Rich Area for Innovation

- Many utilities and layers will be in open source
- Layers to ease transition from OpenGL
- Domain specific flexibility

The same ecosystem dynamic as WebGL

A widely pervasive, powerful, flexible foundation layer enables diverse middleware tools and libraries

What Vulkan Can Do?

User can use Vulkan to develop application for many use cases

1. Graphics
    1. Developers can create 2D/3D hardware accelerated graphical applications
2. Compute
    1. Vulkan supports compute variation of VkQueues/VkPipelines, so Developer can use Vulkan for general computation
3. Ray Tracing
    1. What is Ray Tracing?
        1. Ray tracing is an alternative rendering technique, based around the concept of simulating the physical behavior of light
        2. Vulkan support VK_KHR_ray_tracing_pipeline
4. Vulkan Video
    1. Vulkan Video provide fine-grained control over video processing scheduling, synchronization, and memory utilization to the application.
5. Machine Learning
    1. Make Vulkan a first class API for exposing ML compute capabilities of modern GPUs.
6. Safety Critical
    1. Bring the graphics and compute capabilities of modern GPUs to safety-critical systems in the automotive, avionics, industrial and medical space.

Platforms

Vulkan runs on many platforms, each has small variations

`DVK_USE_PLATFORM_WIN32_KHX`

`DVK_USE_PLATFORM_MACOS_MVK`

`DVK_USE_PLATFORM_ANDROID_KHX`

`DVK_USE_PLATFORM_WAYLAND_KHR`

`DVK_USE_PLATFORM_XCB_KHR`

Checking for Vulkan Support

Platform support and Device Support

1. How to check if your platform is support Vulkan?
    1. Each platform uses a different mechanism to manage how the Vulkan Loader is implemented
        1. Android: run Vulkan Hardware Capability Viewer app developed by Sascha Willems.
        2. BSD Unix: run vulkaninfo in VulkanSDK.
        3. iOS: Vulkan Hardware Capability Viewer provided by LunarG.
        4. Linux: run Vulkaninfo in VulkanSDK
        5. macOS: run Vulkaninfo in VulkanSDK
        6. Windows: run Vulkaninfo.exe in VulkanSDK
    2. The loader is then in charge of determining if a Vulkan Driver is exposed correctly.

Checking for Vulkan Support

Platform Support and Device Support

2. How to Check if your device is support Vulkan?
    1. For device support, one will need to make sure a Vulkan Driver is available that full implements Vulkan.
    2. There are a few different variations of a Vulkan Driver
        1. Hardware Implementation
            1. A certain GPU might have the physical capabilities of running Vulkan, it still requires a driver to control it.
            2. The driver is in charge of getting the Vulkan calls mapped to the hardware in the most efficient way possible.
            3. Drivers, like software have different versions and update, so there can be many variations of drivers for the same physical device and platform.
                1. Vulkan Database https://vulkan.gpuinfo.org/ is the largest collection of recorded Vulkan implementation details.
        2. Null Driver
            1. The term “Null Driver” is given to any driver that accepts Vulkan API calls, but does not do anything with them.
            2. The term “Null Driver” is common for testing interactions with the driver without needing any working implementation backing it.
            3. Many uses cases such as creating CTS Tests for new features, testing the validation layers, and more rely on the idea of a null driver.
            4. Khronos provides Mock ICD as one implementation of a null driver that works on various platforms
    3. Software Implementation
        1. It is possible to create a Vulkan implementation that only runs on the CPU.
        2. Is useful to test Vulkan that is hardware independent, but unlike the null driver, also outputs a valid result
        3. SwiftShader is an example of CPU-based implementation.

Vulkan Versions

How many version types in Vulkan?

1. Instance-level version
    1. Applications use vkEnumerateInstanceVersion to check what version of a Vulkan instance is supported.
2. Device-level version
3. Loader version
4. SPIR-V version
    1. Every minor version of Vulkan maps to a version of SPIR-V that must be supported.
    2. the application to make sure that the SPIR-V in VkShaderModule is of a valid version to the corresponding Vulkan version

What is SPIR-V?

SPIR-V byte-format used in vkCreateShaderModule

1. Standard Portable Intermediate Representation Vulkan
2. SPIR-V white paper
    1. https://registry.khronos.org/SPIR-V/papers/WhitePaper.pdf
3. Introduction to SPIR-V Shaders
    1. https://www.khronos.org/assets/uploads/developers/library/2016-vulkan-devday-uk/3-Intro-to-spir-v-shaders.pdf
4. Using SPIR-V in practice with SPIRV-Cross
    1. https://www.khronos.org/assets/uploads/developers/library/2016-vulkan-devday-uk/4-Using-spir-v-with-spirv-cross.pdf
5. GLSLang
    1. https://github.com/KhronosGroup/glslang
6. Shaderc
    1. A collection of tools, libraries and tests for shader compilation
    2. https://github.com/google/shaderc
    3. Glslc is command line compiler for GLSL/HLSL to SPIR-V
    4. Libshaderc is a library API for accessing glslc functionality.