



Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

 This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

You can learn more at: <http://cs.oregonstate.edu/~mjb/vulkan>

 Vulkan 2016

1

Who is the Real Vulkan?





Do you notice the difference? It's subtle! ☺


 Vulkan 2016

2

Who is the Khronos Group?

The Khronos Group, Inc. is a non-profit member-funded industry consortium, focused on the creation of open standard, royalty-free application programming interfaces (APIs) for authoring and accelerated playback of dynamic media on a wide variety of platforms and devices. Khronos members may contribute to the development of Khronos API specifications, vote at various stages before public deployment, and accelerate delivery of their platforms and applications through early access to specification drafts and conformance tests.



 Vulkan 2016

3

Playing "Where's Waldo" with Khronos Membership



 Vulkan 2016

4

Who's Been Specifically Working on Vulkan?




 Vulkan 2016

5

Vulkan

- Largely derived from AMD's Mantle API
- Also heavily influenced by Apple's Metal API and Microsoft's DirectX 12
- There is no fixed-function – it is all shaders-based
- Fortunately, the shader language Vulkan uses is GLSL
- Goal: much less driver complexity and overhead than OpenGL has
- Goal: much less user hand-holding
- Goal: higher single-threaded performance than OpenGL can deliver
- Goal: able to do multithreaded graphics
- Goal: able to run on desktops and mobile devices

 Vulkan 2016

6

Vulkan Code has a Distinct "Style" of Setting Information in structs and then Passing that Information as a pointer-to-the-struct

```

VkBufferCreateInfo vbci;
vbci.sType = VK_STRUCTURE_TYPE_BUFFER_CREATE_INFO;
vbci.pNext = nullptr;
vbci.flags = 0;
vbci.size = << buffer size in bytes >>
vbci.usage = VK_USAGE_UNIFORM_BUFFER_BIT;
vbci.sharingMode = VK_SHARING_MODE_EXCLUSIVE;
vbci.queueFamilyIndexCount = 0;
vbci.pQueueFamilyIndices = nullptr;

VK_RESULT result = vkCreateBuffer( LogicalDevice, IN &vbci, PALLOCATOR, OUT &Buffer );

VkMemoryRequirements vmr;
result = vkGetBufferMemoryRequirements( LogicalDevice, Buffer, OUT &vmr ); // fills vmr

VkMemoryAllocateInfo vmai;
vmai.sType = VK_STRUCTURE_TYPE_MEMORY_ALLOCATE_INFO;
vmai.pNext = nullptr;
vmai.flags = 0;
vmai.allocationSize = vmr.size;
vmai.memoryTypeIndex = 0;

result = vkAllocateMemory( LogicalDevice, IN &vmai, PALLOCATOR, OUT &MatrixBufferMemoryHandle );

result = vkBindBufferMemory( LogicalDevice, Buffer, IN MatrixBufferMemoryHandle, 0 );

```

Annotations:

- vbci**: Points to the `VkBufferCreateInfo` struct.
- vmr**: Points to the `VkMemoryRequirements` struct.
- vmai**: Points to the `VkMemoryAllocateInfo` struct.
- MatrixBufferMemoryHandle**: Points to the `vkAllocateMemory` output parameter.

Oregon State University Computer Graphics

7

Vulkan Command Buffers

- Graphics commands are sent to command buffers
- Think OpenGL...
- E.g., `vkCmdDoSomething(cmdBuffer, ...);`
- You can have as many simultaneous Command Buffers as you want
- Buffers are flushed when the application wants them flushed
- Each command buffer can be filled from a different thread (i.e., filling is thread-safe)

Oregon State University Computer Graphics

8

Vulkan Graphics Pipelines

- In OpenGL, your graphics "pipeline state" is whatever combination you most recently set: color, transformations, textures, shaders, etc.
- In OpenGL, changing the state is very expensive
- Vulkan forces you to set all your state at once into a "pipeline state object" (PSO) and then invoke the entire PSO whenever you want to use that state combination
- Think of pipeline state as being immutable.
- Potentially, you could have thousands of these pre-prepared states – if there are N things to set, there could be N! possible combinations.
- This is a good time to talk about how game companies view Vulkan...

Oregon State University Computer Graphics

9

Vulkan: Creating a Pipeline

Oregon State University Computer Graphics

10

Vulkan GPU Memory

- Your application allocates GPU memory for the objects it needs
- Your application is responsible for making sure what you put into that memory is actually in the right format, is the right size, etc.

Oregon State University Computer Graphics

11

Vulkan Synchronization

- Events can be set, polled, and waited for (much like OpenGL)
- Vulkan does not ever synchronize – that's the application's (i.e., your) job

Oregon State University Computer Graphics

12

Vulkan Shaders

GLSL is the same as before ... well, almost – here's what's different:

- An implied **#define VULKAN 100** is automatically supplied by the compiler
- You pre-compile your shaders with an external compiler called **glslang**
- Your shaders get turned into a vendor-independent intermediate form known as SPIR-V
- SPIR-V gets turned into fully-compiled, vendor-specific code at runtime
- The SPIR-V spec has been public for years – new shader languages could be developed
- OpenCL and OpenGL have adopted SPIR-V as well

GLSL Source → **External GLSL Compiler** → SPIR-V → **Compiler in driver** → Vendor-specific code

Advantages:

1. Software vendors don't need to ship their shader source
2. Software can launch faster because half of the compilation has already taken place
3. This guarantees a common front-end syntax
4. This allows for other language front-ends

Oregon State University Computer Graphics | 13 | December 1, 2015

13

SPIR-V

SPIR-V stands for **Standard Portable Intermediate Representation – Vulkan**. It's the file format that Vulkan GLSL shaders get compiled into. The name of that front-end compiler is **glslang**. At runtime, that file is read and the driver compiles it the rest of the way into the machine instruction set for that particular vendor.

SPIR-V started out as something for Vulkan but is now also used with OpenGL and OpenCL. Here is how it fits into the overall Khronos Ecosystem:

Oregon State University Computer Graphics | 14 | December 1, 2015

14

Ray-tracing Acceleration is Appearing in Graphics Hardware and it has been Added to the Vulkan API

Oregon State University Computer Graphics | 15 | December 1, 2015

15

Ray-tracing Acceleration is Appearing in Graphics Hardware and it has been Added to the Vulkan API

Oregon State University Computer Graphics | 16 | December 1, 2015

16

The Vulkan Ray Tracing Pipeline Involves Five New Shader Types

Unlike the rasterization pipeline, there is no constant flow from one shader to the next. Rather, particular shaders are called to respond to particular events.

Note: none of this lives in the hardware meant for rasterization graphics. This is all built on top of the GPU compute functionality.

- A Ray Generation Shader runs on a 2D grid of threads. It begins the entire ray-tracing operation.
- An Intersection Shader implements ray-primitive intersections.
- An Any Hit Shader is called when the Intersection Shader finds a hit. It decides if that intersection should be accepted or ignored.
- The Closest Hit Shader is called with the information about the hit that happened closest to the viewer. Typically, lighting is done here, or firing off new rays to handle shadows, reflections, and refractions.
- A Miss Shader is called when no intersections are found for a given ray. Typically, it just sets its pixel color to the background color.

Oregon State University Computer Graphics | 17 | December 1, 2015

17

Ray-Tracing Acceleration Structures

- A Bottom-level Acceleration Structure (BLAS) reads the vertex data from vertex (and possibly index VbBuffers) to determine Axis-Aligned Bounding Boxes (AABBs).
- You can also supply your own AABB information to a BLAS.
- A single Top-level Acceleration Structure (TLAS) holds Instances, which are transformations and pointers to 9potentially) multiple BLASes.
- Each BLAS is essentially used as a Model Coordinate bounding box, while the single TLAS is used as a World Coordinate bounding box.

Oregon State University Computer Graphics | 18 | December 1, 2015

18

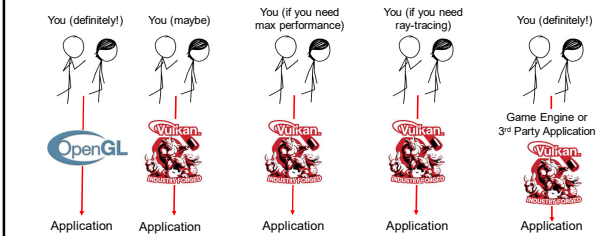
So What Do We All Do Now?

- I don't see Vulkan replacing OpenGL *ever*
- I see the OSU CS 450/550 class using OpenGL existing *forever*
- I see the OSU Vulkan class as always being a one-term standalone course, not part of another OpenGL-based course



So What Do We All Do Now?

This is what I think the model of the immediate future is:



You can learn more at: <http://cs.oregonstate.edu/~mjb/vulkan>