



**Oregon State**  
University  
**Mike Bailey**

[mjb@cs.oregonstate.edu](mailto:mjb@cs.oregonstate.edu)

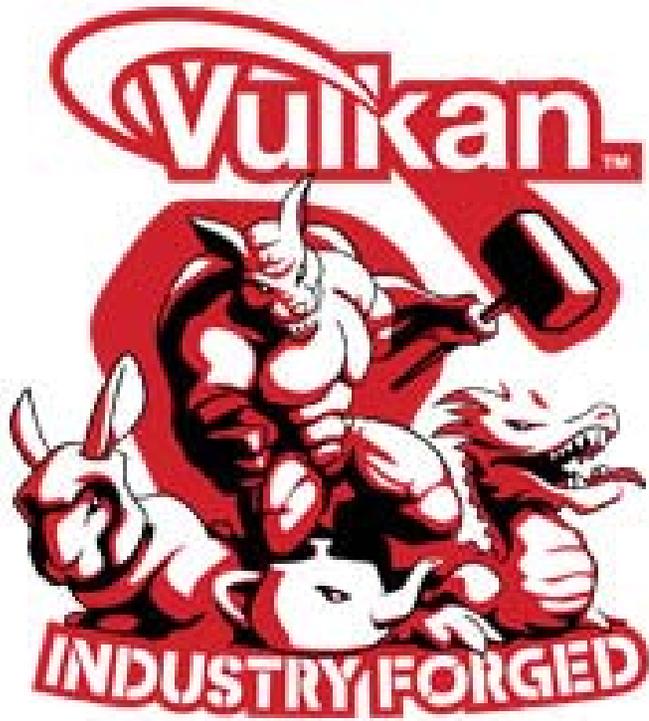


This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



**Oregon State**  
University  
Computer Graphics

## Who is the Real Vulkan?



Can you notice the difference? It's subtle! 😊



Oregon State  
University  
Computer Graphics

## 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.



# Playing “Where’s Waldo” with Khronos Membership

PROMOTER MEMBERS

**KHRONOS GROUP**  
Over 100 members worldwide  
Any company is welcome to join

AMD Apple ARM EPIC GAMES Google  
HUAWEI Imagination intel NOKIA QUALCOMM  
SAMSUNG SONY VALVE NVIDIA VeriSilicon

3D Incorporated AMOTIVE Adobe ALTERA amazon.com AXELL CORPORATION AXIS COMMUNICATIONS BASE MARK BINOMIAL  
BILZARD ENTERTAINMENT BROADCOM BRENWILL cadence CANONICAL CEVA codeplay C-O Continental  
Coordinate COREAVI DASSAULT SYSTEMES CATAPULT Digital omp ETRI FUTUREMARK CORPORATION Gaijin HARMAN HITACHI Inspire the Next HYPERREAL  
igalia Imperial College London 財團法人資訊工業策進會 ITRI KDAB KNU LG Linaro Los Alamos NATIONAL LABORATORY LUNAR  
matrox MAXON MEDIATEK Mentor Graphics Microsoft MIT Lincoln Laboratory mobica Movidius moz://a NINON UNIVERSITY  
NEC Nintendo NXP oculus ON Semiconductor OSU Panasonic PIXAR PLUTO Qt The Qt Company  
RAZER RENESAS Rockwell Collins 서울대학교 sensics Silicon Studio SIRU socionext SPREADTRUM STREAM COMPUTING Performance Engineers  
SYNOPSYS TAKUMI TAMPERE UNIVERSITY OF TECHNOLOGY TU WIEN TECHNISCHE UNIVERSITÄT WIEN TEXAS INSTRUMENTS thinci Think Silicon tobii TOSHIBA umbra  
unity University of BRISTOL UCL University of Windsor 兆芯 Visteon vmware XILINX zSpace



## Who's Been Specifically Working on Vulkan?

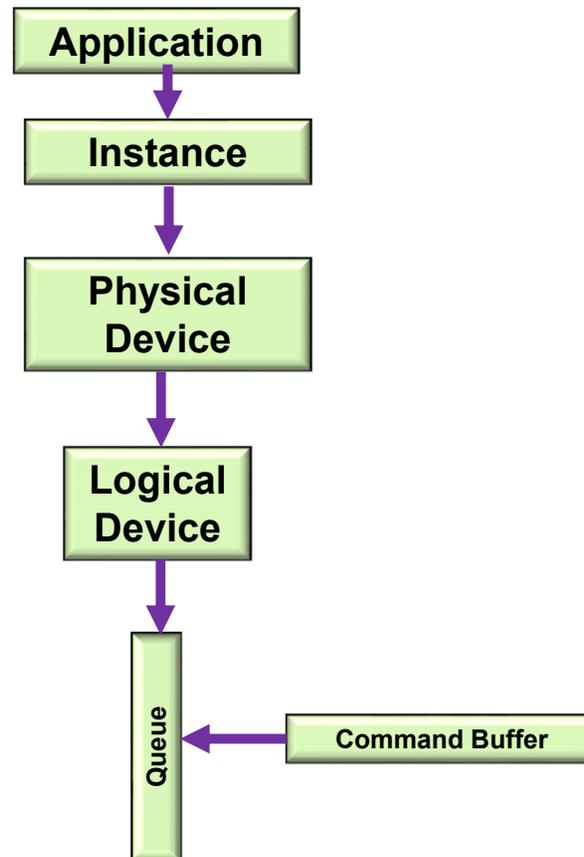


## Vulkan

- Largely derived from AMD's *Mantle* API
- Also heavily influenced by Apple's *Metal* API and Microsoft's *DirectX 12*
- 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 handle tiled rendering



## Vulkan: a Simplified Block Diagram



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

```

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

VK_RESULT result = vkCreateBuffer ( LogicalDevice, IN &vbcj, 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;

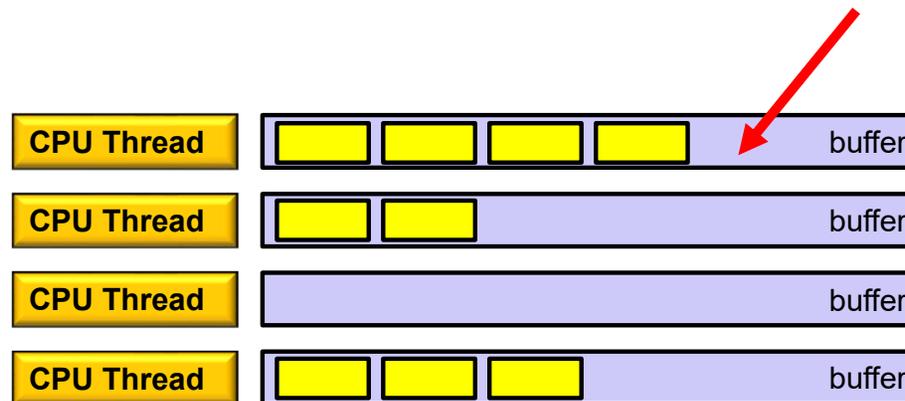
result = vkAllocateMemory( LogicalDevice, IN &vmai, PALLOCATOR, OUT &MatrixBufferMemoryHandle );
result = vkBindBufferMemory( LogicalDevice, Buffer, MatrixBufferMemoryHandle, 0 );

```



## Vulkan Command Buffers

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

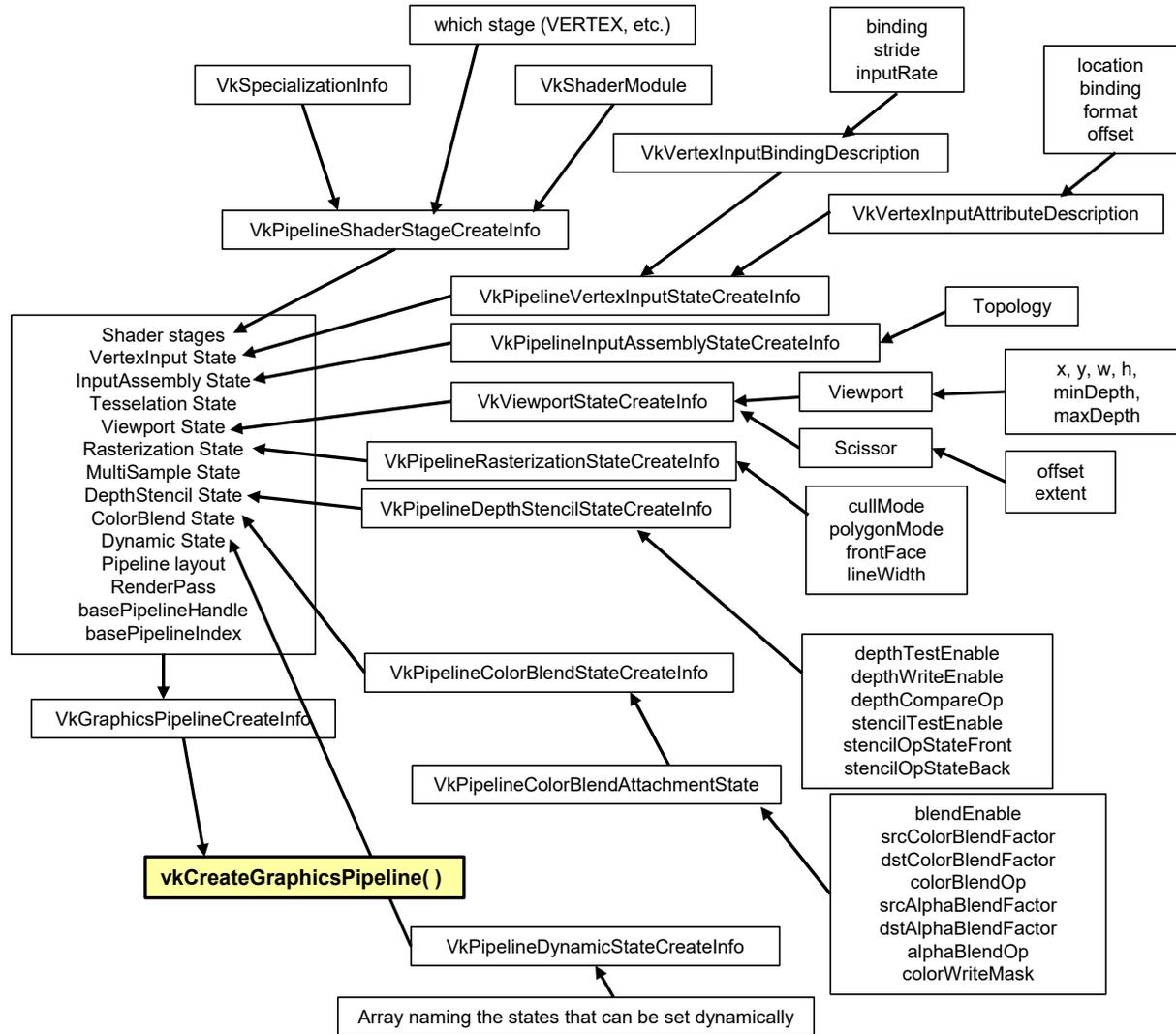


## Vulkan Graphics Pipelines

- In OpenGL, your graphics “pipeline state” is whatever combination you most recently set: color, transformations, textures, shaders, etc.
- 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 would be  $N!$  possible combinations.
- This is a good time to talk about how game companies view Vulkan...



# Vulkan: Creating a Pipeline



## Vulkan GPU Memory

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



## Vulkan Render Passes

- Drawing is done inside a render pass
- Each render pass contains what framebuffer attachments to use
- Each render pass is told what to do when it begins and ends



## Vulkan Synchronization

- Synchronization is the responsibility of the *application*
- Events can be set, polled, and waited for (much like OpenCL)
- Vulkan does not ever lock – that's the application's job
- Threads can concurrently read from the same object
- Threads can concurrently write to different objects



## Vulkan Shaders

- GLSL is the same as before ... almost
- For places it's not, an implied **#define VULKAN 100** is automatically supplied by the compiler
- You pre-compile your shaders with an external compiler
- Your shaders get turned into an intermediate form known as SPIR-V
- SPIR-V gets turned into fully-compiled code at runtime
- The SPIR-V spec has been public for months –new shader languages are surely being developed
- OpenCL and OpenGL will be moving to SPIR-V as well



### 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



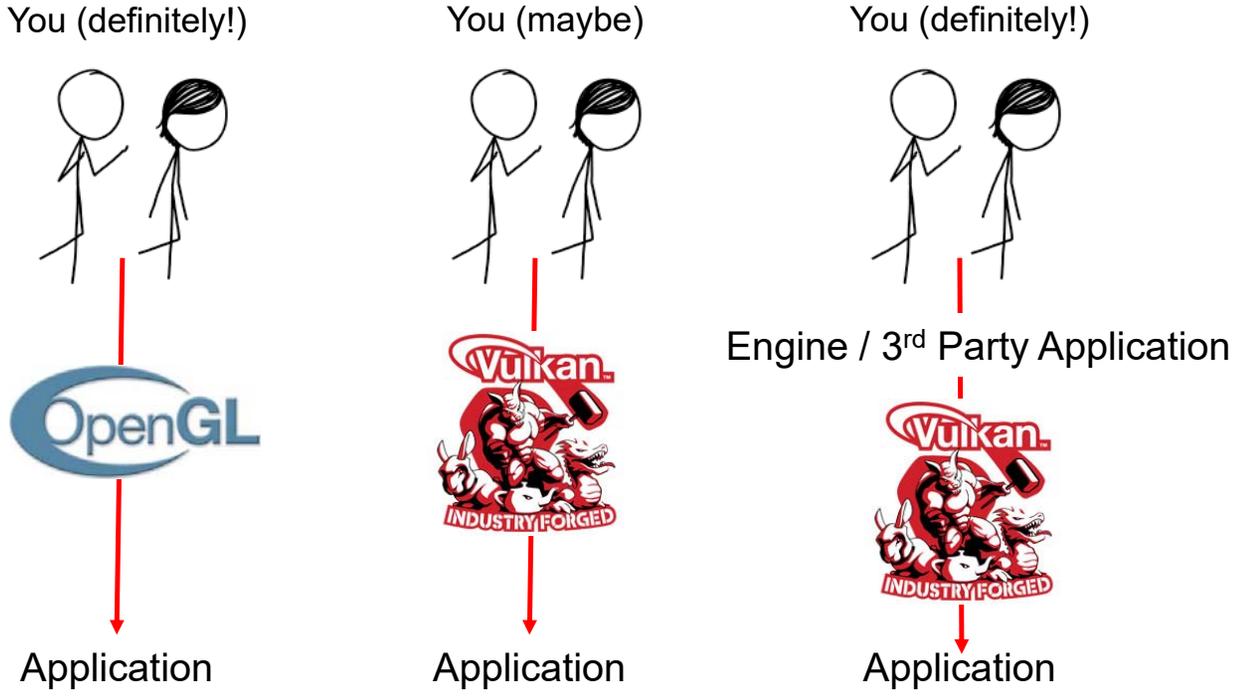
## So What Do We All Do Now?

- I don't see Vulkan replacing OpenGL *ever*
- However, I wonder if Khronos will become less and less excited about adding new extensions to OpenGL. I see no evidence of this right now.
- And, I also wonder if vendors will become less and less excited about improving OpenGL drivers. I see no evidence of this right now.
- 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:



xkcd.com

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