Getting Explicit
How hard is Vulkan Really?
“Vulkan is not a good first-step path to learning graphics. It’s maintainers will tell you to start out by learning OpenGL or DX11.”
– Internet Person
“To be fair to Vulkan, it was designed to be used by somebody who knew what they were doing.” – Another Internet Person
“Eh I would say Vulkan decreases reliability. Precisely because Vulkan drivers are much thinner and do no error checking, ...” – Also an Internet Person
“I thought about the Vulkan programming guide, but ended up deciding modern OpenGL would be an easier starting point...

... In a twist of fate, the SuperBible example code only supports OpenGL 4.0+ and my primary computer is stuck at 3.3.” – Same Internet Person
“So at the end of the day, Vulkan is about giving graphics developers foot-guns in order to reduce CPU overhead. That was explicitly its mission statement from the beginning!” – Proud Internet Person
Background

• 2005: Started in web services.
• 2008-2011: Contract to hire Quake Live
• 2011-2013: Dev Services
• 2013-Present: Generalist idTech
• 8yrs QuakeCon Support
!= Graphics Programmer

• Occasional day-to-day fixes.
• Wrote a handful of shaders.
• Played with ShaderToy, OpenCV, Processing, etc
• Read Peter Shirley’s “Ray Tracing in One Weekend”

• More capable hands at id.
  • Tiago, Billy, Axel, Jean, etc
  • Great talks given at previous events regarding state of graphics in idTech.
  • Great work on Wolfenstein II. Hopefully talk about that soon.
Dawn of Vulkan at id

• 2015 – Breaking the ice.
• Q1 2016 – Began work in earnest close to DOOM2016 launch.
• Q3 2016 – DOOM2016 Vulkan binary released to the wild.

• Clear benefits
• Solidifying marriage with idTech
• Eager for others to Adopt
Early Public Perceptions

• HARD
• A LOT of Code
• Graphics++ for developers at the bleeding edge. Hobbyists and beginners need not apply.
• Expensive investment to refactor. (4 – 6mo, $$$$)
• “Unpacking” the driver revealed more responsibilities.
The Early Days – Body of Information

Top 10 %
Expert

Middle
???

Bottom 10-20 %
Intro
Evangelizing the id way – Open Source

• VkQuake:
  • Author: Axel Gneiting
  • Developed: Q2 2016
  • Released: Q3 2016
  • Link: https://github.com/Novum/vkQuake

• VkDOOM3:
  • Author: Dustin Land
  • Developed: Q3 2016
  • Released: Q3 2017
  • Link: https://github.com/DustinHLand/vkDOOM3
7 Part series articulating my experience with the port.

Thought I’d get 3x dozen views TOTAL.

~30,000 views first month. (broke squarespace’s analytics report)

Average ~1,000 view every month after.

Handful of conversations every month; mostly with beginners.

Considerable interest in Vulkan

https://www.fasterthan.life/
Impetus for Porting DOOM 3 BFG

• Expanding Skillset
  • Explicit APIs are a nice inflection point.
  • Everyone still learning.

• Good Fit
  • Manageable by a single person.
  • Familiar with idTech.
  • Open Source
  • Fixed Assets. (actually a problem later on)

• Evangelizing
  • OpenGL -> Vulkan a nice example of old to new.
  • Partners asking how to make Vulkan more approachable.
Video
Port Effort

- Code:
  - Commits: 718
  - LoC: ~5,000
  - GLSL LoC: ~3,000

- Time:
  - Total: 4x full time months.
  - Regularity: Waxed and Waned
  - When: 10pm – 3am
Vulkan Summoning Ritual – Step 1: Stopping For Intuition
RenderDoc

Intuition Begins And Ends In Understanding How Data Is Transformed

Author: Baldur Karlsson
Link: https://renderdoc.org/
GitHub: https://github.com/baldurk/renderdoc
RenderDoc – Dynamic Shadows
RenderDoc – Color Channels
RenderDoc – Post Processing
Vulkan Summoning Ritual – Step 2: The First 1,000 LoC

The First 1,000 LoC

Stopping For Intuition
The First 1,000 LoC – “An” Entry Trajectory Into Creating Vulkan Applications

1. Create Instance
2. Select Suitable Physical Device
3. Create Surface
4. Create Logical Device & Get Queues
5. Create Render Pass(es)
6. Create Pools
7. Create Swapchain & Get Images
8. Create Command Buffer
9. Allocate Your Frame Objects
10. Create Synchronization Primitives
11. Create Frame Buffer

For Each Swapchain Image

Allocate Your Frame Objects
Create Synchronization Primitives
Create Frame Buffer
Create Command Buffer
The First 1,000 LoC - RenderPass

- Area of interest for Vulkan adopters.
- Doesn’t have to be complicated. (Can be if you’re into that.)
- `vkDOOM3` has 1x RenderPass with 1x subpass and 3x attachments.
- Manually transition layout to `PRESENT_SRC_KHR`.

Depth  | Color  | Resolve - MSAA
Vulkan Summoning Ritual – Step 3: Resources Take Center Stage

The First 1,000 LoC

Stopping For Intuition

Resources Take Center Stage
Resources Take Center Stage

• Buffers in vkDOOM3
  • Static Geo (DEVICE_LOCAL)
  • Dynamic Geo double buffered. (HOST_VISIBLE)
  • UBOs for skinning and render parms.
  • idBuffer treats range as discrete

• Images in vkDOOM3
  • 12x Formats
  • All DEVICE_LOCAL.
  • Track VkImageLayout for descriptor updates.
  • TRANSFER_DST -> SHADER_READ_ONLY
Resources Take Center Stage - Allocator

• There are options.
  • Simple Apps – Just use the API directly. Watch your numbers.
  • Small to Medium Apps – Simple block allocator. Be cognizant of the rules.
  • Medium to Large – You really need to invest in something serious.

https://vulkan.gpuinfo.org/
Resources Take Center Stage - What I did for allocations

```c
VkResult ret = vkCreateBuffer(
  vkcontext.device,
  &bufferCreateInfo,
  NULL,
  &m_apiObject );

VkMemoryRequirements memoryRequirements;
vkGetBufferMemoryRequirements(
  vkcontext.device,
  m_apiObject,
  &memoryRequirements );

// determine mem usage
m_allocation = vulkanAllocator.Allocate(
  memoryRequirements.size,
  memoryRequirements.alignment,
  memoryRequirements.memoryTypeBits,
  memUsage,
  VULKAN_ALLOCATION_TYPE_BUFFER );
```

```c
struct vulkanAllocation_t {
  idVulkanBlock * block;
  uint32       id;
  VkDeviceMemory deviceMemory;
  VkDeviceSize  offset;
  VkDeviceSize  size;
  byte *       data;
};
```
Don’t shoot yourself in the face.
Only 120/600 LoC header/source.

Resources Take Center Stage - Allocator Breakdown

Get Heap Index
Allocate Block Chain Per Heap
Subdivide On Allocation
Merge On Free
VMA – Vulkan Memory Awards
Vulkan Memory Allocator

Author: Adam Sawicki
Link: https://gpuopen.com/
GitHub: https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator
Vulkan Summoning Ritual – Step 4: Coupling to idTech 4

Coupling to idTech 4

The First 1,000 LoC

Stopping For Intuition

Resources Take Center Stage
Coupling to idTech 4

- Long history with OpenGL pedigree. (2000 – Present)
- GL code had “grown roots”. Present in 18x files. Not bad, but still leaky
- PS3 and 360 code had gone down a different path.
- Wanted to keep OpenGL working side-by-side for comparison.
- Large part of port effort was “putting the beast back in its cage”.
- In the end, GL code sequestered to 8x files.
Coupling to idTech 4 – State Tracking

- Designed to change state “immediately”. (take that driver)
  - GL_State
  - GL_SelectTexture / GL_BindTexture
  - GL_Clear

- Required state tracking till data could be used at appropriate times.
  - Descriptor Set Updates
  - Pipeline Creation / Selection

- Current state saved to GLS bits, but not the data. (uint64_t)
  - Blend Ops/Functions
  - Depth State
  - Cull Mode
  - Stencil State
  - etc

// example snippet
static const uint64 GLS_SRCBLEND_ONE = 0 << 0;
static const uint64 GLS_SRCBLEND_ZERO = 1 << 0;
static const uint64 GLS_SRCBLEND_DST_COLOR = 2 << 0;
static const uint64 GLS_SRCBLEND_ONE_MINUS_DST_COLOR = 3 << 0;
static const uint64 GLS_SRCBLEND_SRC_ALPHA = 4 << 0;
static const uint64 GLS_SRCBLEND_ONE_MINUS_SRC_ALPHA = 5 << 0;
static const uint64 GLS_SRCBLEND_DST_ALPHA = 6 << 0;
static const uint64 GLS_SRCBLEND_ONE_MINUS_DST_ALPHA = 7 << 0;
static const uint64 GLS_SRCBLEND_BITS = 7 << 0;

// In StencilShadowPass
GL_State(
  glState |
  GLS_STENCIL_OP_FAIL_KEEP |
  GLS_STENCIL_OP_ZFAIL_KEEP |
  GLS_STENCIL_OP_PASS_INCR |
  GLS_STENCIL_MAKE_REF(STENCIL_SHADOW_TEST_VALUE) |
  GLS_STENCIL_MAKE_MASK(STENCIL_SHADOW_MASK_VALUE) |
  GLS_POLYGON_OFFSET |
  GLS_CULL_TWOSIDED );
Coupling to idTech 4 – State Tracking Example

**OpenGL::GL_State()**
- 210 LoC
- glEnable/glDisable
- glCullFace
- glDepthFunc
- glDepthMask
- glBlendFunc
- glColorMask
- glPolygonMode
- glPolygonOffset
- glStencilOp

**Vulkan::GL_State()**
- 5 LoC
- Save off GLS bits
- Submitted at pipeline creation
Vulkan Summoning Ritual – Step 5: Pixels & Pipelines

Coupling to idTech 4

The First 1,000 LoC

Stopping For Intuition

Resources Take Center Stage

Pixels & Pipelines
• Surface based drawing - two routines
  • Main lighting
  • Stencil shadows

• Steps
  • Find offsets into index / vertex buffers.
  • Set joint cache handle if we’re skinning.
  • `renderProgManager.CommitCurrent(GLS bits)`
  • `vkCmdBindIndexBuffer`
  • `vkCmdBindVertexBuffer`
  • `vkCmdDrawIndexed`
  • Profit
Pixels & Pipelines - CommitCurrent

• Backend sets a current “render prog” index. ::BindProgram( int index )
• CommitCurrent uses index to retrieve render prog which has a list of viable pipelines.
• Lookup pipeline using GLS bits.
  • If one doesn’t exist, create it.
  • No noticeable perf hit, so didn’t bother with pipeline cache.
• Desc Set – Brute force approach, but still quite fast.
  • Allocate a DescSet
  • Update it using preallocated UBO and bound textures.
  • Bind
• Bind Pipeline
Pixels & Pipelines – Graphics Is a Series of Tubes

28 built in shaders + change
Under 100 pipelines total at runtime.

Create Built-In Shaders
Create Descriptor Set Layouts
Create Vertex Descriptions
Create Descriptor Pools
Preallocate UBOs For RenderParms
Special Thanks

Blog Readers
Axel Gneiting – vkQuake, Advice
Tiago Sousa – Talk Review, Advice
Stephanie Hurlburt – Encouragement
John Carmack – Open Source
OG & BFG DOOM 3 Teams

Jan Paul van Waveren
Steven Serafin
Getting Explicit:
How hard is Vulkan Really?

Panel Discussion
Khronos Dev Day - GDC 2018
Getting Explicit: How hard is Vulkan Really?

Panelists:

- Matthäus Chajdas, AMD
  @NIV_Anteru
- Arseny Kapoulkine, Roblox
  @zeuxcg
- Dustin Land, id Software
  @DustinHLand

Chair:

- Alon Or-bach, Samsung Electronics
  @alonorbach