Vulkan, OpenGL, and OpenGL ES

SIGGRAPH 2017
Agenda

• OpenGL
  - Piers Daniell, NVIDIA

• OpenGL ES
  - Tobias Hector, Imagination Technologies

• Vulkan
  - Tom Olson, ARM
  - ...with the Vulkan working group and community

• Par-tay!
  - Everyone
OpenGL Update

Piers Daniell, NVIDIA
OpenGL Working Group chair
New OpenGL working group chair

Barthold Lichtenbelt
ARB Chair 2006 - 2016
11 OpenGL releases!

Piers Daniell
OpenGL Chair 2016 - ?
1 release…

Thanks Barthold!
New OpenGL working group chair

Principal Software Engineer at NVIDIA
OpenGL/Vulkan core driver team
With ARB working group since 2008
Also in the Vulkan working group
API specification editor: Jon Leech
GLSL specification editor: John Kessenich

From the OpenGL 4.6 press release:
“The OpenGL working group will continue to respond to market needs and work with GPU vendors to ensure OpenGL remains a viable and evolving graphics API for all its customers and users across many vital industries.”
said Piers Daniell, chair of the OpenGL Working Group at Khronos
Happy 25th Birthday OpenGL!
Happy 25th Birthday OpenGL!

OpenGL 1.0 - 1992
OpenGL 1.1 - 1997
OpenGL 1.2 - 1998
OpenGL 1.3 - 2001
OpenGL 1.4 - 2002
OpenGL 1.5 - 2003
OpenGL 2.0 - 2004
OpenGL 2.1 - 2006
OpenGL 3.0 - 2008
OpenGL 3.1 - 2009
OpenGL 3.2 - 2009
OpenGL 3.3 - 2010
OpenGL 4.0 - 2010
OpenGL 4.1 - 2010
OpenGL 4.2 - 2011
OpenGL 4.3 - 2012
OpenGL 4.4 - 2013
OpenGL 4.5 - 2014
OpenGL 4.6 - 2017
Happy 25th Birthday OpenGL!

OpenGL 25th Anniversary T-Shirt and stuff available to purchase from the Khronos store:
  https://www.khronos.org/store/
  https://teespring.com/opengl-25th-anniversary-black
Commemorative drink koozie
  BOF Blitz After-Party
## OpenGL Then and Now

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Triangles / sec (millions)</strong></td>
<td>1</td>
<td>~1,200 (x1,200)</td>
<td>~20,000 (x20,000)</td>
</tr>
<tr>
<td><strong>Pixel Fragments / sec (millions)</strong></td>
<td>240</td>
<td>19,600 (x81)</td>
<td>152,000 (x633)</td>
</tr>
<tr>
<td><strong>GigaFLOPS (fp32)</strong></td>
<td>0.64</td>
<td>750 (x1,170)</td>
<td>10,960 (x17,125)</td>
</tr>
<tr>
<td><strong>Power consumption</strong></td>
<td>1.5kW</td>
<td>&lt;15W</td>
<td>250W</td>
</tr>
</tbody>
</table>

**Ideas in Motion - SGI**

1992 - 2017

**DOOM 2016 - id Software**

1992 - 2017
## Evolution of the OpenGL draw call

<table>
<thead>
<tr>
<th>Version</th>
<th>Function</th>
<th>Character count</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL 1.0</td>
<td>glBegin/glVertex/glEnd</td>
<td>8</td>
</tr>
<tr>
<td>OpenGL 1.1</td>
<td>glDrawElements</td>
<td>14</td>
</tr>
<tr>
<td>OpenGL 1.2</td>
<td>glDrawRangeElements</td>
<td>19</td>
</tr>
<tr>
<td>OpenGL 1.4</td>
<td>glMultiDrawElements</td>
<td>19</td>
</tr>
<tr>
<td>OpenGL 3.1</td>
<td>glDrawElementsInstanced</td>
<td>23</td>
</tr>
<tr>
<td>OpenGL 3.2</td>
<td>glDrawElementsInstancedBaseVertex</td>
<td>33</td>
</tr>
<tr>
<td>OpenGL 4.2</td>
<td>glDrawElementsInstancedBaseVertexBaseInstance</td>
<td>45</td>
</tr>
<tr>
<td>OpenGL 4.6</td>
<td>glMultiDrawElementsIndirectCount</td>
<td></td>
</tr>
</tbody>
</table>
Announcing...

Credits:

Eric Lengyel, Terathon Software
OpenGL 4.6 Design Philosophy

- Raise the baseline OpenGL feature set
  More features for developers that require core functionality
- Raise OpenGL quality with substantial conformance improvement
  Now available as open source on GitHub
- Support existing hardware
- Remain 100% compatible with OpenGL 4.5 and before
- Fold widely supported and popular extensions into core
  Easy for hardware vendors to implement
What’s new in OpenGL 4.6?

Shader functionality
- ARB_gl_spirv
- ARB_spirv_extensions
- ARB_shader_group_vote
- ARB_shader_atomic_counter_ops

AZDO (Approaching Zero Driver Overhead) functionality
- ARB_indirect_parameters
- ARB_shader_draw_parameters

Improving rendering quality
- ARB_texture_filter_anisotropic (finally)
- ARB_polygon_offset_clamp

Other functionality
- ARB_pipeline_statistics_query
- ARB_transform_feedback_overflow_query
- KHR_no_error
OpenGL 4.6 Specs and Drivers

OpenGL 4.6 and GLSL 4.60 specifications:
https://www.khronos.org/registry/OpenGL/index_gl.php

OpenGL 4.6 beta drivers from NVIDIA:
https://developer.nvidia.com/opengl-driver

Most features already implemented in Mesa:
https://www.mesa3d.org/
https://mesamatrix.net/
SPIR-V Ecosystem

SPIR-V
- Khronos defined and controlled cross-API intermediate language
- Native support for graphics and parallel constructs
- 32-bit Word Stream
- Extensible and easily parsed
- Retains data object and control flow information for effective code generation and translation

Third party kernel and shader languages
- glslang
- GLSL
- HLSL
- MSL

OpenCL C Front-end
OpenCL C++ Front-end
LLVM

SPIR-V (Dis)Assembler
SPIR-V Cross
SPIR-V Validator

Other Intermediate Forms

IHV Driver Runtimes

Khronos coordinating liaison with Clang/LLVM Community
E.g. discussing SPIR-V as supported Clang target

Standard in OpenGL 4.6

https://github.com/KhronosGroup/SPIRV-Tools
Using GLSL Shaders with OpenGL

Read GLSL text from file

- glCreateProgram
- glCreateShader
- glShaderSource
- glCompileShader
- glAttachShader

while more shader domains

- glGetAttribLocation
- glGetUniformLocation
- glProgramUniform
- glUseProgram

while more attributes to introspect
while more uniforms to introspect

```c
void main()
{
    vec4 rgb = texture
    vec3 N = normalize
    vec3 L = normalize
}```
Using SPIR-V Shaders with OpenGL

Read SPIR-V binary blob from file

\[
\begin{align*}
glCreateProgram & \rightarrow \text{glCreateShader} \\
 & \rightarrow \text{glShaderBinary} \\
 & \rightarrow \text{glSpecializeShader} \\
 & \rightarrow \text{glAttachShader} \\
\end{align*}
\]

\[
\begin{align*}
glLinkProgram & \rightarrow \text{app assume locations assigned within the shader, obviating dynamic introspection} \\
 & \rightarrow \text{while more uniforms to initialize} \\
 & \rightarrow \text{glProgramUniform} \\
 & \rightarrow \text{glUseProgram} \\
\end{align*}
\]

while more shader domains

© Copyright Khronos Group 2017 - Page 17
GLSL -> SPIR-V compiler

*glslang* in GitHub already updated to support GLSL 4.60

[https://github.com/KhronosGroup/glslang](https://github.com/KhronosGroup/glslang)

Supports all new features:

- ARB_shader_group_vote
- ARB_shader_atomic_counter_ops
- ARB_shader_draw_parameters

#version 460
AZDO Features

New buffer binding

```c
glBindBuffer(GL_PARAMETER_BUFFER);
```
Buffer source for reading the indirect draw count

Two new draw commands:

```c
glMultiDrawArraysIndirectCount(mode, indirect, drawcount, );
glMultiDrawElementsIndirectCount(mode, type, indirect, drawcount, );
```

Uses same indirect structs in `GL_DRAW_INDIRECT_BUFFER` as before:

```c
struct DrawArraysIndirectCommand {
    GLuint count;
    GLuint primCount;
    GLuint first;
    GLuint baseInstance;
};
```

```c
struct DrawElementsIndirectCommand {
    GLuint count;
    GLuint primCount;
    GLuint firstIndex;
    GLint baseVertex;
    GLuint baseInstance;
};
```

New vertex shader builtins:

- `gl_DrawID` - index of draw command vertex belongs to
- `gl_BaseVertex`, `gl_BaseInstance` - from command buffer
Anisotropic Texture Filter

Improve texture rendering quality of long and narrow textures
Polygon Offset Clamp

Eliminates light cracks with large depth-slope shadow cast rendering

```c
glPolygonOffsetClamp(factor, units, clamp);
```

\[
o = \begin{cases} 
    m \times \text{factor} + r \times \text{units}, & \text{clamp} = 0 \text{ or } NaN \\
    \min(m \times \text{factor} + r \times \text{units}, \text{clamp}), & \text{clamp} > 0 \\
    \max(m \times \text{factor} + r \times \text{units}, \text{clamp}), & \text{clamp} < 0
\end{cases}
\]

Image credit: Eric Lengyel
Other Extensions

GL_KHR_parallel_shader_compile
  Bring native multi-threaded compile support to OpenGL ES
  Conformance coverage coming soon

Cross-process and cross-API interop extensions:
  GL_EXT_memory_object
  GL_EXT_memory_object_win32
  GL_EXT_memory_object_fd
  GL_EXT_semaphore
  GL_EXT_semaphore_win32
  GL_EXT_semaphore_fd
  GL_EXT_win32_keyed_mutex

New window extensions for GL_KHR_no_error:
  WGL_ARB_create_context_no_error and GLX_ARB_create_context_no_error
OpenGL Ecosystem Update

GLEW - The OpenGL Extension Wrangler
Updated with OpenGL 4.6 and the latest OpenGL extensions
http://glew.sourceforge.net/
Thanks Nigel Stewart!

OpenGL 4.6 reference card now available
Pick up a free copy here at the Khronos BOF!

OpenGL Conformance Test Suite (CTS) improvements:
Khronos investing in new coverage
New coverage inherited from OpenGL ES
Now open-source: https://github.com/KhronosGroup/VK-GL-CTS
OpenGL 4.6 CTS coming soon with lots of new coverage:
  Complete 4.6 coverage
  Additional 3.x - 4.x coverage
Conclusion

OpenGL 4.6 improves the baseline feature set in the core specification
OpenGL will continue to evolve to serve the needs of its customers
Will remain a viable 3D graphics API choice:
  Legacy 3D applications
  Higher-level API
  Innovation platform

Happy 25th Birthday!
25th Anniversary Trivia Prize!

OpenGL 25th Anniversary T-Shirt
Bonus 25th Anniversary Trivia Prize!

NVIDIA GeForce GTX...
Bonus 25th Anniversary Trivia Prize!

NVIDIA GeForce GTX USB thumb drive
Loaded with complete OpenGL-Registry
OpenGL ES Update

Tobias Hector, Imagination Technologies
OpenGL ES Working Group chair
OpenGL ES: Status

• OpenGL ES is extremely prevalent
  - 3.x has >60% market penetration*
  - 3.1 / 3.2 adoption still increasing

• No plan for new core version
  - Vulkan’s momentum is displacing it
  - Extensions still being developed
  - Continuing to watch market

• Focused on quality of life
  - Addressed the issue backlog
  - Looking to publish spec updates soon
  - GLSLang support for #version 320 es
  - Huge progress in CTS

* Sources:
  https://developer.android.com/about/dashboards/index.html
  http://hwstats.unity3d.com/mobile/gpu.html
OpenGL ES: Conformance

- Conformance was open sourced in January
  - Got there in the end!
  - One remaining part that is closed-source
    - ES is poised to remove that dependency soon

- 3 releases so far, more on the way
  - CTS still very actively maintained
  - Funding secured for further development
  - Addressing important holes in coverage
  - Working through backlog of issues
OpenGL ES: Extensions

- Many EXTs added over the last year
  - Members addressing market needs

- Various bits of new functionality
  - A number of minor features
  - Platform interactions
  - GL/ES and Vulkan content sharing
  - KHR_parallel_shader_compile

EXT_conservative_depth
EXT_clear_texture
EXT_draw_transform_feedback
EXT_multisampled_render_to_texture2
EXT_texture_compression_astc_decode_mode
EXT_texture_compression_astc_decode_mode_rgb9e5
EXT_EGL_image_array
EXT_memory_object
EXT_semaphore
EXT_memory_object_fd
EXT_semaphore_fd
EXT_memory_object_win32
EXT_semaphore_win32
EXT_win32_keyed_mutex
EXT_external_buffer
EXT_texture_compression_rgtc
EXT_texture_compression_bptc
KHR_parallel_shader_compile
Vulkan Update

Tom Olson, ARM
Vulkan Working Group chair
Vulkan

Design goals

- Clean, modern architecture
- Low overhead, explicit
- Portable across desktop and mobile
- Multi-thread / multi-core friendly
- Efficient, predictable performance

Emergent properties

- Community-facing and responsive
- Recognize central role of the ecosystem
- Strong commitment to open source
Vulkan at SIGGRAPH 2016

A typical six-month-old

- Loads of potential
- Getting a lot of attention
- Not really doing that much

Photo credit: Lou Haach

https://www.flickr.com/photos/lourdes_fisio/6877521944
Vulkan at SIGGRAPH 2017

At 18 months...

- Still a work in progress
- But, enormously more capable!
- Growing and changing in all directions
- A bit chaotic, but a lot more fun

Photo credit: Johnathan Nightingale

https://www.flickr.com/photos/johnath/5358512977
Availability

Production drivers from all three desktop GPU vendors
- No more betas!
- *some assembly required

Platforms
- Linux, Windows, Steam / SteamVR
- Standard interface exposed in Android 7.x

Mobile
- Phones and tablets from Google, Huawei, Samsung, Sony, Xiaomi,…
- Both premium and mid-range devices
- Nintendo Switch, NVIDIA Shield / Shield TV

For the latest, see http://vulkan.gpuinfo.org/
Games and Game Engines

• At SIGGRAPH 2016
  - The Talos Principle
  - Dota2, UE4, Doom

• Today
  - UE4, Unity 5.6
  - Serious Engine
  - Oculus SDK
  - Mad Max (beta)
  - CryEngine 5.4 (beta)

• Rumors
  - Quake Champions, Ashes of the Singularity, Wolfenstein II, ...
Mobile too!

- UE4
- Unity 5.6
- Galaxy on Fire 3 - Manticore
- Lineage2 Revolution
- Heroes of Incredible Tales
- GRID Autosport
- Score! Hero
- Dream League Soccer
- ...the list goes on
Developer Interest

- LunarG SDK download rate has more than doubled since launch
- Available at LunarXchange: http://vulkan.lunarg.com
GitHub Activity

At SIGGRAPH 2016

We’ve found 431 repository results

1,254 repository results

glfw/glfw
A multi-platform library for OpenGL, OpenGL ES, Vulkan, window and input

Updated a day ago
Recent Examples
Khronos / Working Group Activity

- 30 new KHR extensions
  - Bug fixes and new tech

- GLSLang
  - Extensive HLSL support

- Many SDK improvements

- Conformance Test progress
  - Current release has 198K test cases
  - Up from 107K last year

- Specification is now accepting pull requests!
## Up Next...

<table>
<thead>
<tr>
<th>Time</th>
<th>Session Title</th>
<th>Speaker(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:00</td>
<td>Working Group Status Updates</td>
<td>Piers Daniell, NVIDIA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tobias Hector, Imagination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tom Olson, ARM</td>
</tr>
<tr>
<td>4:45</td>
<td>New Features in Vulkan</td>
<td>Jan-Harald Fredriksen, ARM</td>
</tr>
<tr>
<td>4:40</td>
<td>Vulkan Portability Initiative</td>
<td>Neil Trevett, NVIDIA</td>
</tr>
<tr>
<td>4:55</td>
<td>Vulkan Compute: Porting OpenCL C to Vulkan</td>
<td>Ralph Potter, Codeplay</td>
</tr>
<tr>
<td>5:05</td>
<td>HLSL in Vulkan</td>
<td>Hai Nguyen, Google</td>
</tr>
<tr>
<td>5:15</td>
<td>LunarG Vulkan Ecosystem Update</td>
<td>Karen Ghavam, LunarG</td>
</tr>
<tr>
<td>5:25</td>
<td>Vulkan on UE4: Summer 2017</td>
<td>Rolando Caloca, Epic Games</td>
</tr>
<tr>
<td>5:35</td>
<td>Q&amp;A</td>
<td>You!</td>
</tr>
<tr>
<td>5:45</td>
<td>Party Time!</td>
<td>Everyone</td>
</tr>
</tbody>
</table>
New Features in Vulkan

Jan-Harald Fredriksen, ARM
New features

- Vulkan Next in active development
  - Core spec in definition
  - Many features available as extensions

- 38 Khronos ratified extensions (KHR)
- 3 Khronos ratified experimental extensions (KHX)
  - NOT recommended for use in production code
- 15 cross-vendor extensions (EXT)
- >30 vendor extensions
The first few

- **VK_KHR_maintenance1**
  - Render to slices of 3D image
  - `vkCmdCopyImage` between 3D slice to 2D array layer
  - Negative viewport height to support left handed NDC
  - `VK_FORMAT_FEATURE_TRANSFER_*_BIT_KHR` for staging only resources
  - `vkCmdFillBuffer` on transfer-only queues
  - `vkTrimCommandPoolKHR` to return command pool memory to the system

- **VK_KHR_shader_draw_parameters**
  - New built-in shader variables
  - `BaseInstance`, `BaseVertex`, and `DrawIndex`

- **Making structures extendable - used by other extensions**
  - `VK_KHR_get_physical_device_properties2`
  - `VK_KHR_get_memory_requirements2`
  - `VK_KHR_get_surface_capabilities2`
Sharing memory

- Needed for compositors and other system integration
  - Resource sharing at memory object level
  - Works across logical devices, process, and API boundaries
  - No longer KHX

- Platform independent core
  - VK_KHR_external_memory
  - VK_KHR_external_memory_capabilities

- Platform specific types
  - VK_KHR_external_memory_fd
  - VK_KHR_external_memory_win32

- Support for backing data resources with single memory allocations
  - VK_KHR_dedicated_allocation
  - May be required for sharing in some circumstances
Sharing synchronization primitives

• Also need to synchronize access to shared memory

• Semaphores
  - VK_KHR_external_semaphore
  - VK_KHR_external_semaphore_capabilities
  - VK_KHR_external_semaphore_win32
  - VK_KHR_external_semaphore_fd
  - VK_KHR_win32_keyed_mutex (DX11)

• Fences
  - VK_KHR_external_fence
  - VK_KHR_external_fence_capabilities
  - VK_KHR_external_fence_win32
  - VK_KHR_external_fence_fd
Cross API sharing

- Related set of GL / GLES extensions to import Vulkan memory
  - GL_EXT_memory_object
  - GL_EXT_semaphore
  - GL_EXT_memory_object_fd
  - GL_EXT_semaphore_fd
  - GL_EXT_memory_object_win32
  - GL_EXT_semaphore_win32
  - GL_EXT_win32_keyed_mutex
Multi-GPU

• Native multi-GPU support for NVIDIA SLI and AMD Crossfire platforms
  - VK_KHR_device_group
  - VK_KHR_device_group_creation

• Supports explicit AFR, SFR and VR rendering algorithms
• Device mask to select which physical device to use
VR and Display

- **VK_KHR_multiview**
  - For stereo rendering
  - One command to multiple views
  - Extends render pass
  - View mask, offset, correlation

- **VK_KHR_shared_presentable_image**
  - Application and presentation engine can access an image at the same time
  - Reduced latency

- **VK_KHR_incremental_present**
  - Provide damage regions in vkQueuePresentKHR
Updating descriptor sets

- **VK_KHR_descriptor_update_template**
  - Use to updating same set of descriptors in many descriptor sets with same layout

  ![Diagram](image)

  - `VkWriteDescriptorSet`
  - `vkUpdateDescriptorSets`
  - `VkDescriptorUpdateTemplateCreateInfoKHR`
  - `vkCreateDescriptorUpdateTemplateKHR`
  - `VkDescriptorUpdateTemplateKHR`
  - `vkUpdateDescriptorSetWithTemplateKHR`

- **VK_KHR_push_descriptor**
  - Update small number of descriptors from the command buffer
  - Driver managed instead of descriptor sets
  - Can make it easier to port existing code
Compute and shading language

- **VK_KHR_16bit_storage**
  - 16-bit types in shader input and output interfaces, and push constant blocks

- **VK_KHR_variable_pointers**
  - Invocation-private pointers into uniform and/or storage buffers

- **See next presentation!**

- **VK_KHR_storage_buffer_storage_class**
  - New SPIR-V StorageBuffer storage class
  - Distinguishes Uniform and StorageBuffers without extra decorations
  - Used to describe constraints - HW treats these storage classes differently

- **VK_KHR_relaxed_block_layout**
  - Relax restrictions on offset decorations - for HLSL compatibility
In the pipeline

- **Maintenance2**
  - Allow depth-stencil images be read-only / writeable per aspect
  - View compressed image formats as integers
  - Fix tessellatiion domain origin
  - Describe the clipping behavior of points

- **Subgroup operations**
  - Expose cross-lane/warp operations

- **Enabling features like VR cinema**
  - Protected memory to display DRM protected content
  - YCbCr formats with color space conversions
Vulkan Portability Initiative

Neil Trevett, NVIDIA
Khronos President / Vulkan Portability TSG chair
Market Demand for Universal 3D Portability

Community Outreach at GDC 2017
Create a hybrid Portability API?

Feedback - AVOID CREATING A FOURTH API!!!
Would need new specification, CTS, Documentation.
Additional developer learning curve.
A whole new specification to name, brand, promote.
Would INCREASE industry fragmentation
Vulkan Portability TSG Process

Open source project with identical goals already underway - come and help! https://github.com/gfx-rs/gfx

Vulkan Portability Deliverables
1. Vulkan Subset Diff Spec
2. Vulkan Subset Development Layer
3. Vulkan Subset API Library over DX12/Metal
4. SPIRV-Cross Translator
5. Vulkan Subset Conformance Tests

Layers, APIs, Translators and Tests all to be developed and released in open source

Possible proposals for Vulkan extensions for enhanced portability (and possibly Web robustness) sent to Vulkan WG

Identify Vulkan features not directly mappable to DX12 and Metal

New Vulkan functionality may affect the overlap analysis

Expand/test existing open source SPIRV-Cross Tool
OpenCL and Vulkan

Single source C++ programming. Great for supporting C++ apps, libraries and frameworks.

Industry working to bring Heterogeneous compute to standard ISO C++
C++17 Parallel STL hosted by Khronos
Executors - for scheduling work
“Managed pointers” or “channels” - for sharing data

OpenCL 1.2
OpenCL C Kernel Language

SYCL 1.2
C++11 Single source programming

2011

OpenCL 2.1
SPIR-V in Core

SYCL 2.2
C++14 Single source programming

2015

OpenCL 2.2
C++ Kernel Language

SYCL 2.2
C++14 Single source programming

2017

OpenCL for DSPs
- Embedded imaging, vision and inferencing
- Flexible reduced precision
- Conformance without IEEE 32 Floating Point
- Explicit DMA

Help bring OpenCL-class compute to Vulkan
Vulkan Long Term Goal

And a great first step...

Clspv open-source OpenCL C to Vulkan Compiler Project
Adobe has ported 200K lines of OpenCL C to Vulkan
Proof-of-concept that OpenCL compute can be brought seamlessly to Vulkan
Vulkan Compute
Porting OpenCL C to Vulkan

Ralph Potter, Codeplay
Introduction

Experimental work bringing a large OpenCL C codebase to Vulkan compute
Collaboration between Google, Codeplay, and Adobe
Evaluating over 200K lines of production code selected from Adobe products
Compiler implementation driven by real world needs

Need to resolve differences between Vulkan’s SPIR-V execution environment
and OpenCL C’s requirements
Alternatively, OpenCL C’s programming model, compared to GLSL

Required a prototype compiler, and new extensions
VK_KHR_16bit_storage/SPV_KHR_16bit_storage
VK_KHR_variable_pointers/SPV_KHR_variable_pointers

Proof-of-concept for other pointer-based languages
Vulkan Adoption

All Major GPU Companies shipping Vulkan Drivers - for Desktop and Mobile Platforms

Mobile, Embedded and Console Platforms Supporting Vulkan

Android 7.0
Nintendo Switch
Android TV
Embedded Linux

Cross Platform
SteamOS
ubuntu
redHat
TIZEN
Windows 7
Windows 8
Windows 10
16-bit Storage

VK_KHR_16bit_storage enables the SPV_KHR_16bit_storage SPIR-V extension

Enables the use of 16-bit types in shader interfaces
   16-bit types in shader input and output interfaces, storage buffers and push constant blocks
   Potential bandwidth reductions from smaller types
   Also helps us tackle OpenCL C’s 16-bit types

Supports OpLoad, OpStore, and conversion to/from 32-bit types
Variable Pointers

VK_KHR_variable_pointers enables the SPV_KHR_variable_pointers SPIR-V extension

Enables per-invocation dynamic pointers into storage buffers and optionally work-group storage

More constrained than “generic” pointers
  Provides pointers to externally visible storage
  Without the potential performance impact of more general form

Two variants and corresponding capabilities/feature flags
  VariablePointers - Addresses all storage buffers and work-group storage
  VariablePointersStorageBuffer - Constrained to a single interface block
CLSPV Compiler

Prototype OpenCL C 1.2 to Vulkan compiler
Tracks top-of-tree LLVM and clang, not a fork
Open-sourced: https://github.com/google/clspv
Map OpenCL address spaces to SPIR-V storage classes
Translate OpenCL C builtins to GLSL.std.450 extended instruction set
Map pointer arithmetic to VariablePointers
kernel
void interleave(global float *dst,
    global float *src_a,
    global float *src_b)
{
    int id = get_global_id(0);
    global float *src =
        (id % 2) ? src_a : src_b;
    dst[id] = src[id / 2];
}

// Pointers to StorageBuffer src_a, src_b
%28 =   OpAccessChain %2 %24 %14 %14
%29 =   OpAccessChain %2 %25 %14 %14
// Load GlobalInvocationId
%30 =   OpAccessChain %11 %17 %14
%31 =   OpLoad %6 %30
// Src = (GlobalInvocationId & 1 == 0) ?
//     src_b : src_a
%32 =   OpBitwiseAnd %6 %31 %15
%33 =   OpIEqual %12 %32 %14
// Dynamically select between two pointers
%34 =   OpSelect %2 %33 %29 %28
// Load Src[GlobalInvocationId / 2]
%35 =   OpSDiv %6 %31 %16
%36 =   OpPtrAccessChain %2 %34 %35
%37 =   OpLoad %1 %36
// Store Dst[GlobalInvocationId]
%38 =   OpAccessChain %2 %23 %14 %31
OpStore %38 %37
OpReturn
Limitations

OpenCL builtins without Vulkan/GLSL equivalents are not supported
  bitselect, nextafter, prefetch, printf, async_work_group_copy...
8/16-wide vectors
Numerical precision matches Vulkan’s SPIR-V environment
  OpenCL has strict precision rules for builtin functions
Anything that relies on pointer sizes
Byte-addressable data types
Despite these limitations, we only need to modify ~30 lines out of > 200K LOC
https://github.com/google/clspv/blob/master/docs/OpenCLCOnVulkan.md
Acknowledgements

David Neto
John Kessenich

Eric Berdahl

Neil Henning
JinGu Kang
HLSL in Vulkan

Hai Nguyen, Google
Overview

- How Does HLSL Work in Vulkan?
- HLSL Compilers for Vulkan
  - Glslang
  - Shaderc
  - DXC
How Does HLSL Work in Vulkan?

- By compiling to SPIR-V of course!

- Vulkan had the necessary bits to support most of HLSL
  - Most of required plumbing had a direct mappings of concepts
  - Some other concepts required a bit of fitting to work

- Changes in Vulkan to accommodate HLSL
  - Added HLSL-style unaligned buffer access via extension
    - Enables [float, float3] layouts within a 16 byte boundary for StructuredBuffers

- Ongoing work to add more coverage of HLSL in tools
Glslang (Khronos/Google/LunarG)

- First compiler to support HLSL in Vulkan
- HLSL support is complete enough for real world projects
  - DOTA 2 (Valve)
  - Ashes of Singularity (Oxide Games)
- What shader models are supported?
  - Mostly SM5.0 and some SM5.1
    - Largely driven by community asks
Glslang HLSL (1/2)

- All shader stages work
  - VS=vert, HS=tesc, DS=tese, GS=geom, PS=frag, CS=comp
- For supported features HLSL source can be compiled unmodified
- HLSL registers map to binding numbers
  - Normally descriptor set 0, but can override
    - `--resource-set-binding`
    - GLSL syntax or HLSL `spaceN` parameter in `register()`
- Location for I/O variables is based on declaration order
Glslang HLSL (2/2)

- Supports all CBV/SRV/UAV types
  - UAVs that have counters will consume 2 binding slots
    - 1 for resource
    - 1 for counter buffer (hidden and not referenced in HLSL source)
  - Mapping HLSL resource types to Vulkan resource types can be tricky
    - Samplers -> Samplers
    - Textures -> Images
    - cbuffer/ConstantBuffer -> Uniform Buffer
Glslang

- Working with HLSL in Vulkan
  - Command options to shift binding number offsets for Vulkan
    - `--shift-sampler-binding <value>`
    - `--shift-texture-binding <value>`
    - `--shift-cbuffer-binding <value>`
    - `--shift-uav-binding <value>`
  - Resolves overlap in binding numbers translated from `register`
  - Binding number offsets can also be auto assigned
Shaderc (Google)

- Shaderc depends on glslang so HLSL support is roughly the same
  - There's a bit of lag since Shaderc uses Google's glslang repo instead of the Khronos repo
- Can optionally execute spirv-opt as part of the build process
- Working with HLSL in Vulkan
  - Command line options for binding number offsets is different
    - `-ftexture-binding-base [stage] <value>`
    - `-fsampler-binding-base [stage] <value>`
    - `-fubo-binding-base/-fcbuffer-binding-base [stage] <value>`
Spiregg in dxc (Google/Microsoft)

- dxc
  - Based on LLVM and Clang 3.7
  - Only supports HLSL
  - Targets SM6.0 and higher

- Google contributing SPIR-V codegen (spiregg)
  - Actively developing
  - Actively merged into dxc mainline on official repo

- SPIR-V progress
LunarG Vulkan Ecosystem Update

Karen Ghavam,
LunarG, Inc.
CEO/Engineering Director
LunarG Vulkan Ecosystem Update

VK_LAYER_LUNARG_device_simulation
New SPIR-V Optimizations

For more information, email info@lunarg.com
VK_LAYER_LUNARG_device_simulation

Without Device Simulation

Application

Loader

ICD

GPU

All actual device capabilities

Actual device capabilities are exposed

With Device Simulation

Application

Loader

DevSim

ICD

GPU

Fewer capabilities, filtered from configuration

All actual device capabilities

Simulated capabilities are exposed
VK_LAYER_LUNARG_device_simulation

• Test application without requiring all actual devices
  - Modifies results from Vulkan queries
  - Device configuration defined by JSON file

• Use cases
  - Exercise fall-back code paths, when a capability isn’t available.
  - Find unintentional assumptions (triggers validation errors)
  - Test application behavior under severe resource constraints

• Simulation, NOT Emulation
  - Simulation: Changes query results from more-capable device to simulate less-capable device
  - Not emulation: Does not remove (enforce) capabilities that are actually present on actual device
  - Not emulation: Doesn’t add more capabilities not already present in actual device
Device Simulation Layer Resources

- JSON schema for validating configuration files
  - Verify configuration files are correct
  - [https://schema.khronos.org/vulkan/devsim_1_0_0.json#](https://schema.khronos.org/vulkan/devsim_1_0_0.json#)

- Integrated with Sascha Willems database
  - [https://vulkan.gpuinfo.org/](https://vulkan.gpuinfo.org/)
  - Device data is already accessible in DevSim schema-compliant JSON format

- Development continues, more features to implement:
  - Extensions, Formats
  - Memory, Queues
  - Others? Suggestions?

- Available now
  - Source at [https://github.com/LunarG/VulkanTools](https://github.com/LunarG/VulkanTools)
  - Please submit issues
  - Binaries in the next Vulkan SDK release
  - Developed by Mike Weiblen: mikew@lunarg.com
Announcing New SPIR-V Optimizations

Currently Supports:
Shaders with Logical Addressing
Entry Point Functions

Optimizations include:
Inlining (exhaustive)
Store/Load Elimination
Dead Code Elimination (aggressive)
Dead Branch Elimination
Common Uniform Elimination (PR pending)

Now < 40% the size of original SPIR-V*
Less than 40% larger than DX Byte Code*

*Your mileage may vary
New SPIR-V Optimizations - What’s next

- Inlining (no growth)
- Optimization Time Improvements
- Loop Unrolling (performance)

Future Exploration:
- Constant Folding
- Common Subexpression Elimination

Please submit your issues on github (copy @greg-lunarg)

For more information contact:
Contact Greg Fischer
greg@lunarg.com
Last season, on UE4...

- Feb 2016: Vulkan SDK publicly released
Last season, on UE4...

- Feb 2016: Vulkan SDK publicly released
- Protostar!
  - Samsung S7 launch event
  - Mobile Renderer
    - Feature Level ES3.1
Last season, on UE4...

- Feb 2016: Vulkan SDK publicly released
- Protostar!
- Lineage 2 Revolution
Today

- ShooterGame
Today

- InfiltratorDemo
Today

- Unreal Tournament
Today

- Editor
Today

MUCH AWESOME

SO WOW

Tappy Chicken

UNREAL ENGINE
Today

- Shader Model 5 is **the** default renderer/RHI for Vulkan desktop
  - Previously was SM4 - D3D10 (no compute techniques)
  - Run it today! UE4Editor -vulkan
Today

- Shader Model 5 is the default renderer/RHI for Vulkan desktop
  - Previously was SM4 - D3D10 (no compute techniques)
  - Run it today! UE4Editor -vulkan
    - Caveat emptor: Still some bugs
    - So please report them :)
Today

- Shader Model 5 is the default renderer/RHI for Vulkan desktop
- RHI API Update
  - More compliant with modern style APIs
  - Renderer tells more information upfront to the RHI
    - Explicit transitions

```cpp
RHICmdList.TransitionResource( EResourceTransitionAccess::EReadable, SceneContext.GetSceneDepthSurface() );
```
Today

- Shader Model 5 is the default renderer/RHI for Vulkan desktop
- RHI API Update
  - More compliant with modern style APIs
  - Renderer tells more information upfront to the RHI
    - Explicit transitions
    - Pipeline states are now first-class citizens of the Renderer and RHIs
Today

- Shader Model 5 is the default renderer/RHI for Vulkan desktop
- RHI API Update
  - More compliant with modern style APIs
  - Renderer tells more information upfront to the RHI
    - Explicit transitions
    - Pipeline state

```cpp
// Set the graphic pipeline state.
FGraphicsPipelineStateInitializer GraphicsPSOInit;
RHICmdList.ApplyCachedRenderTargets(GraphicsPSOInit);
GraphicsPSOInit.DepthStencilState = TStaticDepthStencilState::GetRHI();
GraphicsPSOInit.BlendState = TStaticBlendState::GetRHI();
GraphicsPSOInit.RasterizerState = TStaticRasterizerState::GetRHI();
GraphicsPSOInit.PrimitiveType = PT_TriangleList;
GraphicsPSOInit.BoundShaderState.VertexDeclarationRHI = GetVertexDeclarationFVector4();
GraphicsPSOInit.BoundShaderState.VertexShaderRHI = GETSAFERHISHADE_VERTEX(*VertexShader);
GraphicsPSOInit.BoundShaderState.PixelShaderRHI = GETSAFERHISHADE_PIXEL(*PixelShader);
SetGraphicsPipelineState(RHICmdList, GraphicsPSOInit);
```
Today

- Shader Model 5 is the default renderer/RHI for Vulkan desktop
- RHI API Update
- Focus on stability and visual parity with D3D11
Today

- Shader Model 5 is the default renderer/RHI for Vulkan desktop
- RHI API Update
- Focus on stability and visual parity with D3D11
- Tons of fixes for Vulkan on 4.17
  - Refactored descriptor set management
  - Fixed a lot of gfx issues
  - Validation warning messages drastically down
  - Ongoing work! More fixes coming to main/github
Shader Model 5 is the default renderer/RHI for Vulkan desktop
RHI API Update
Focus on stability and visual parity with D3D11
Tons of fixes for Vulkan on 4.17
Goal: Default RHI on Linux
#todo

- CPU
  - Descriptor Sets
    - Improve layouts

```csh
layout(set=3, binding=4, std140) uniform HLSLCC_CBh
{
    vec4 pu_h[12];
};

layout(set=3, binding=0) uniform sampler2D ShadowDepthTexture;
layout(set=3, binding=1) uniform sampler2D SceneDepthTexture;
layout(set=3, binding=2) uniform sampler2D GBuffers_GBufferDTexture;
layout(set=3, binding=3) uniform sampler2D GBuffers_GBufferBTexture;
layout(location=0) out vec4 out_Target0;
```
#todo

- **CPU**
  - **Descriptor Sets**
    - Improve layouts
    - Optimize run-time updates
#todo

- CPU
  - Descriptor Sets
  - Parallel RHI threads
    - Generate command buffers going wide
    - Intra-thread layout/barrier tracking

Render → RHI
#todo

- CPU
  - Descriptor Sets
  - Parallel RHI threads
    - Generate command buffers going wide
    - Intra-thread layout/barrier tracking
#todo

- **CPU**
- **GPU**
  - Some missing features (eg DFAO)
  - Deep dive with Radeon GPU Profiler & RenderDoc!
    - Redundant transitions/barriers
    - Redundant/empty render passes
    - Harness multiple/async queues
#todo-next

- Render Passes as first-class citizen of the RHI
  - Will allow the RHI to stop guessing what the Renderer wants to do
  - Less tracking
  - Also helps with transitions!
#todo-next

- Render Passes as first-class citizen of the RHI
  - Will allow the RHI to stop guessing what the Renderer wants to do
  - Less tracking
  - Also helps with transitions!
#todo-next

- Render Passes as first-class citizen of the RHI
- Offline/Cooked PSOs
  - Conservative shader compilation
    - ‘Dynamically spawn point light with atmospheric fog for a skeletal mesh that has morph targets using a blueprint’
#todo-next

- Render Passes as first-class citizen of the RHI
- Offline/Cooked PSOs
  - Conservative shader compilation
  - Plan
    - Reduce # vertex formats using dynamic vertex fetch
    - Mark pipelines (vertex/pixel pairings) ahead of time
    - Gather possible render target formats
    - + we know material state (blend, depth) ahead of time...
    - => Can pre-create PSOs at cook time

### Vertex Inputs
### Shaders
### RT Formats
### Material State
#todo-next

- Render Passes as first-class citizen of the RHI
- Offline/Cooked PSOs
  - Conservative shader compilation
  - Plan
  - Side Gain: Reduces total # of shaders compiled!
#todo-next

- Render Passes as first-class citizen of the RHI
- Offline/Cooked PSOs
  - Conservative shader compilation
  - Plan
  - Side Gain: Reduces total # of shaders compiled!
  - Helps with hitches creating PSOs at runtime
    - (Meanwhile we still have the save pipeline cache to disk solution)
Longer Term...

- Tessellation
- Multi-GPU support
Debugging Tips

- Use validation layers
- Use RenderDoc
- Use Radeon Graphics Profiler
- Add debug modes to submit command lists:
  - After every EndRenderPass
  - After every Dispatch
  - After every Blit/Copy
- Add debug mode to WaitForIdle after every submit
  - Great for tracking GPU hangs!
- Keep shader source at runtime to cross-reference
Thanks!

- RenderDoc/BaldurK
- LunarG & glslang teams
- AMD for Radeon Graphics Profiler
- Vulkan Working Group
Come Back at 5:45 for the After-Party

Sponsored by:

CESIUM

NVIDIA

WWW.KHRONOS.ORG