## Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session Title</th>
<th>Speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:20</td>
<td>Vulkan: The State of the Union</td>
<td>Tom Olson (Arm), Karl Schultz (LunarG), Jeff Bolz (NVIDIA)</td>
</tr>
<tr>
<td>2:40</td>
<td>Making Use of New Vulkan Features</td>
<td>Matthäus Chajdas (AMD), Tiemo Jung (Gaijin), Lewis Gordon (Samsung)</td>
</tr>
<tr>
<td>4:00</td>
<td>Bringing ‘Fortnite’ to Mobile with Vulkan and OpenGL ES</td>
<td>Jack Porter (Epic Games) and Kostiantyn Drabeniuk (Samsung)</td>
</tr>
<tr>
<td>5:30</td>
<td>Ubisoft’s Experience Developing with Vulkan</td>
<td>Jean-Noe Morissette (Ubisoft)</td>
</tr>
</tbody>
</table>
Vulkan: The State of the Union

GDC 2019
#KhronosDevDay
Vulkan and OpenGL ES Updates

Vulkan: The State of the Union
- Tom Olson (Arm), Vulkan WG chair

GPU Assisted Validation: A Shallow Dive
- Karl Schultz (LunarG)

Buffer References
- Jeff Bolz (NVIDIA)
Vulkan: The State of the Union

Tom Olson (Arm), Vulkan WG Chair
Adoption news: Hardware

Support from all* major GPU vendors
- Standard in desktop driver drops

Still some legacy issues
- Driver updates needed for older HW
- Low-end mobile GPU limitations
Adoption news: Platforms

Supported on the major open Oses
- Best on Android, Linux, Win 10

Portability Initiative for closed platforms
- Linkable open source libraries
- Map Vulkan calls to native API
- Support only the efficient subset
Platform news: Q is my favorite letter!

Introducing Android Q Beta
13 March 2019

Posted by Dave Burke, VP of Engineering


Vulkan everywhere

We're continuing to expand the impact of Vulkan on Android, our implementation of the low-overhead, cross-platform API for high-performance 3D graphics. Our goal is to make Vulkan on Android a broadly supported and consistent developer API for graphics. We're working together with our device manufacturer partners to make Vulkan 1.1 a requirement on all 64-bit devices running Android Q and higher.
Platform news: Google’s new game platform

Welcome Stadia!
• AAA gaming in the cloud
• Custom AMD-based GPU
• Vulkan 1.1 under Linux

Wasn’t that an awesome keynote?
### Platform news: Portability Initiative

**MoltenVK** released in open source for macOS and iOS

- macOS SDK released by LunarG

**gfx-rs**
- GFX-rs DX11 backend in alpha
- joins DX12 and Metal backends

**RenderDoc**
- RenderDoc integrated with MoltenVK

**Vulkan**
- Early prototype of Portability Extension released on public GitHub and shipped in MoltenVK

### Future Milestones
- CTS 100% Pass for Reported Functionality + Completed DevSim/Validation Layers
- Portability Extension 1.0 Release
- Follow progress and give feedback at [https://github.com/KhronosGroup/Vulkan-Portability](https://github.com/KhronosGroup/Vulkan-Portability)

### Features added to MoltenVK since GDC 2018
- 24 new Vulkan extensions
  - Tessellation
  - Per-texture swizzling
- Pipeline cache load/store
- GPU switching
- 3 swapchain images
- Performance improvements

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDC 2018</td>
<td>August 2018</td>
</tr>
</tbody>
</table>
| September 2018 | January 2019| GDC 2019
Vulkan Apps Shipping On Apple with MoltenVK

**Forsaken Remastered** was just updated with **Vulkan** support! If you're on Linux, you're probably hitting 60fps with the existing OpenGL renderer, but it's good to be future proof. If you're on a Mac, though, you **definitely** want to switch. On my MacBook, the framerate goes from around 15 to a solid 60!

*Initial Vulkan Performance On macOS With Dota 2 Is Looking Very Good*

Written by Michael Larabel in Valve on 1 June 2018 at 05:37 PM EDT. 34 Comments

*Yesterday Valve released Vulkan support for Dota 2 on macOS. Indeed, this first major game relying upon MoltenVK for mapping Vulkan over the Apple Metal drivers is delivering performance gains.*

**Production Dota 2 on Mac**

- Ships - up to 50% more perf than Apple's OpenGL
- First iOS Apps using MoltenVK ship through app store

**Multiple iOS and macOS apps organically ported**

- Google Filament PBR Renderer on Mac
- Initial ports of Wine games in progress using Vulkan on Mac

**Qt Running on Mac through MoltenVK**

**Artifact from Steam ships on MoltenVK on macOS - first Vulkan-only Valve app on Mac**

**RPCS3 PlayStation 3 Emulator on Mac**

**Dolphin GameCube and Wii Emulator**

**Diligent Engine**

**Qt Running on Mac through MoltenVK**

**June 2018**

**September 2018**

**November 2018**

**January 2019**
Game Engine News

Most modern engines have Vulkan rendering paths

Editor support from EPIC, Unity
Vulkan is the default UE4 RHI under Linux
Vulkan AAA Content Shipping on Desktop...

Vulkan AAA Titles on PC

AAA titles on Linux

macOS, SteamOS and PC
...and Mobile

Plus....
Lineage 2 Revolution
Heroes of Incredible Tales
Dream League Soccer...
Developer Interest - SDK Downloads

Windows

MacOS

Linux
Developer Interest - GitHub Projects

SIGGRAPH 2016

SIGGRAPH 2017

GDC 2019

Trend over time

SIGGRAPH 2016

GDC 2017

SIGGRAPH 2017

GDC 2018

SIGGRAPH 2018

GDC 2019

1,254 repository results

2,295 repository results

2,925 repository results

SaschaWillems/Vulkan

Examples and demos for the new Vulkan API
What we’ve been working on

New functionality - API and shading language extensions
- Driven by developer and ecosystem priorities
- Larger / longer-range efforts happen in subgroups (TSGs)

Strengthening the ecosystem
- SPIR-V and HLL toolchain
- Improving implementation quality / reliability
- Developer tools, documentation, support...
- Generally making Vulkan a better target for developers
Functionality news (highlights)

**VK_EXT_scalar_block_layout**
- Structures and arrays can be aligned based on their largest component
- Full (we hope?) support for HLSL buffer layouts

**VK_EXT_memory_budget, VK_EXT_memory_priority (experimental)**
- Query heap capacity and usage
- Prioritize memory allocations that should stay in device_local memory

**VK_KHR_driver_properties**
- Query driver ID (e.g. AMD vs RADV) and conformance status

**VK_EXT_pipeline_creation_feedback**
- Determine whether new pipelines were found in the API pipeline cache
Functionality news (more highlights)

VK_KHR_shader_float16_int8, VK_KHR_shader_float_controls
- Shader arithmetic on reduced precision types
- Query and override rounding mode, denorm handling, etc

VK_EXT_descriptor_indexing
- Very large descriptor sets (at least 64k)
- Reference with dynamically non-uniform indices
- Update after bind is OK
- Updating while in flight is OK too! (if you know what you’re doing)
- Invalid descriptors are OK (VK_DESCRIPTOR_BINDING_PARTIALLY_BOUND_BIT_EXT)

VK_EXT_buffer_device_address
(Yeah, there’s more)

VK_KHR_8bit_storage
VK_KHR_create_renderpass2
VK_KHR_depth_stencil_resolve
VK_KHR_draw_indirect_count
VK_KHR_get_display_properties2
VK_KHR_shader_atomic_int64
VK_KHR_surface_protected_capabilities
VK_KHR_swapchainmutable_format
VK_KHR_vulkan_memory_model

VK_EXT_depth_clip_enable
VK_EXT_filter_cubic
VK_EXT_fragment_density_map
VK_EXT_full_screen_exclusive
VK_EXT_host_query_reset
VK_EXT_separate_stencil_usage
VK_EXT_transform_feedback
VK_EXT_validation_features
VK_EXT_vertex_attribute_divisor
VK_EXT_ycbcr_image_arrays
Larger efforts in flight

Portability
- Support Vulkan on closed platforms without native drivers

Ray Tracing
- The rendering technology of the future!

Video
- Expose video encode/decode through Vulkan

Machine Learning
- Expose ML compute capabilities of modern GPUs

Safety Critical (external)
- Vulkan for safety critical systems
Language and tools news

SPIR-V enhancements in flight
- Make SPIR-V a better IR for compilers, tools, IHV GPU back ends

dxc / SPIREGG - SPIR-V back end for Microsoft’s HLSL compiler
- fp16 support considered stable, 8- and 64-bit support in development

clspv - compile OpenCL kernels to run under Vulkan
- Initially developed to support Adobe Project Rush
- Work continues; added experimental fp16 / int8 support

Internal tooling improvements
- Using ClusterFuzz to generate many test cases from a single shader
- Using Amber to capture minimal CTS test cases from complex applications
Conformance test news

Vulkan CTS investments

- Our largest engineering project
- Adding >100K test cases / year
- Pre-2018 tests now deprecated

You can help!

- Raise issues and PRs at https://github.com/KhronosGroup/VK-GL-CTS
Validation News

Debian Linux packages for Vulkan SDK

- In addition to the old SDK “tarball”

Vulkan WG is now handling routine new feature validation

- Frees up LunarG resources to work on more advanced validation tooling

GPU Assisted Validation

- (for example)
SDK and toolchain news

Maybe our toolchain could be a bit better organized?

- VS -

We’re working on it...
First steps

Welcome Kris Rose!
• Dedicated DevRel resource for Vulkan / OpenXR
• Mission: professionalize our developer outreach
• Initial planning and developer survey complete
• (Thanks to those who participated!)

Work in (very early) progress
• An official Vulkan code samples repository
• To be developed in collaboration with leading community contributors

More work in (very early) progress
• An official Vulkan SDK
Steps toward a Vulkan SDK

- LunarG donates Vulkan SDK to Khronos

- **What’s changing**
  - LunarG will open desktop SDK build and packaging scripts to Vulkan WG
  - LunarG Vulkan SDK rebranded as “The Vulkan SDK”
  - New look for Vulkan SDK download site: [https://vulkan.lunarg.com/](https://vulkan.lunarg.com/)

- **Benefits**
  - Strengthened collaboration between LunarG & Khronos Vulkan WG
  - Evolution toward a more coherent, unified ecosystem

- **What’s not changing**
  - LunarG’s leadership role in maintaining / developing / delivering the SDK
  - Valve’s sponsorship and support
New Look for the Vulkan SDK Download Site
What else should we be doing?

This is where you come in...

- What (if anything) makes it hard for you to use Vulkan successfully?
- What are the most important problems for us to focus on?

We’ll take feedback anywhere, any time

- General issues: https://github.com/KhronosGroup/Vulkan-Ecosystem
- Spec issues: https://github.com/KhronosGroup/Vulkan-Docs
- SDK issues: LunarXchange (http://vulkan.lunarg.com)
- Here, on the street, slack, reddit, in the bars, ...

We look forward to hearing from you!
GPU-Assisted Validation: A Shallow Dive

GDC 2019 - Khronos Developer Day
LunarG - Karl Schultz
What is GPU-Assisted Validation?

GPU-assisted validation consists of using the GPU to perform certain validation checks at shader execution time.

- Built into existing validation layers
- Additional checking is applied to all relevant shaders
- Activated easily with the flip of a switch
  - Better than various manual and targeted shader debug approaches
Validating Bindless Descriptor Accesses

- “Bindless” not in the spec!
- One of the inspirations for GPU-assisted validation.
- Descriptor from the array is not bound until run time.

1. Not bindless, bound at compile time.
2. Not bindless, compile time error.

```cpp
layout (set = 0, binding = 1) uniform sampler2D tex[6];
...
1) uFragColor = light * texture(tex[4], texcoord.xy);
...
2) uFragColor = light * texture(tex[10], texcoord.xy);
...
3) uFragColor = light * texture(tex[tex_ind],
   texcoord.xy);
```
Bindless Checking Use Case

- Each cube face drawn with a different texture (\texttt{tex[6]}).
- Texture chosen by \texttt{VertexIndex->texture map in uniform buffer}.
- But one face isn’t supposed to be black.
- Could be worse!
  - \texttt{VK_ERROR_DEVICE_LOST}
- What is causing this and where is it happening?
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment. Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43: uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: **UNASSIGNED**-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CmdBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment.  Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43:   uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment. Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43: uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 | Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.

Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.

Fragment coord (x,y) = (190.5, 138.5).

Shader validation error occurred in file: /home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43:   uFragColor = light * texture(tex[tex_ind], texcoord.xy);

Name of command buffer set by application with vkSetDebugUtilsObjectNameEXT

Handle of command buffer containing the offending Draw
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).

Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment. Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43:   uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 | Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.

**Pipeline (Pipeline #0)(0x41).**

Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment. Fragment coord = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43: uFragColor = light * texture(tex[tex_ind], texcoord.xy);

Name and handle of the pipeline bound at the time of the Draw.
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: UNASSIGNED - Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).

**Shader Module (Cube Fragment Shader)(0x3f).**

Shader Instruction Index = 116.
Stage = Fragment. Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file: /home/user/src/Vulkan-Tools/cube/cube.frag at line 43:
43:  uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: UNASSIGNED - Descriptor index out of bounds
Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf#1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment. Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43: uFragColor = light * texture(tex[tex_ind], texcoord.xy);

SPIR-V instruction that caused the error
47: 46(ptr) AccessChain 42(tex) 45
48: 37 Load 47
52: 32(fvec4) Load 50(texcoord)
53: 51(fvec2) VectorShuffle 52 52 0 1
54: 32(fvec4) ImageSampleImplicitLod 48 53
55: 32(fvec4) VectorTimesScalar 54 35
    Store 34(uFragColor) 55
56: 32(fvec4) Load 34(uFragColor)
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 |
Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 0 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment.  Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file:
/home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43:  uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Turn on GPU-Assisted Validation to See:

ERROR : VALIDATION - Message Id Number: 0 | Message Id Name: UNASSIGNED-Descriptor index out of bounds

Index of 6 used to index descriptor array of length 6.
Command buffer (CubeDrawCommandBuf #1)(0xbc6c80).
Draw Index 0.
Pipeline (Pipeline #0)(0x41).
Shader Module (Cube Fragment Shader)(0x3f).
Shader Instruction Index = 116.
Stage = Fragment. Fragment coord (x,y) = (190.5, 138.5).
Shader validation error occurred in file: /home/user/src/Vulkan-Tools/cube/cube.frag at line 43.
43: uFragColor = light * texture(tex[tex_ind], texcoord.xy);
Real World Use Case - Source 2 (Dan Ginsburg)

• Experiencing VK_ERROR_DEVICE_LOST in workload using bindless textures

• Tried the usual things:
  - Could not capture in RenderDoc due to device lost error
  - Enabling VK_NV_device_diagnostic_checkpoints masked the problem
  - Spent a couple of days investigating bug

• Tested with work-in-progress GPU Validation layer
  - Reported accessing descriptor array with negative value in specific shader/line
  - Immediately able to trace back to push constant index value being set to -1. Found and fixed CPU side code with the error.
How to Activate GPU-Assisted Validation

- Need SDK 1.1.97 or better
- Not on by default
  - Uses resources
  - ~10% performance penalty (very shader-dependent)
- Programmatic activation
  The VK_EXT_validation_features extension can be used to enable GPU-Assisted Validation at CreateInstance time.

Here is sample code illustrating how to enable it:

```c
VkValidationFeatureEnableEXT enables[] = {VK_VALIDATION_FEATURE_ENABLE_GPU_ASSISTED_EXT};
VkValidationFeaturesEXT features = {};  
features.sType = VK_STRUCTURE_TYPE_VALIDATION_FEATURES_EXT;
features.enabledValidationFeatureCount = 1;
features.pEnabledValidationFeatures = enables;

VkInstanceCreateInfo info = {};
info.pNext = &features;
```
How to Activate GPU-Assisted Validation

- **Layer configuration file activation**
  The existing layer configuration file mechanism can be used to enable GPU-Assisted Validation. This mechanism is described on the LunarXchange website, in the "Layers Overview and Configuration" document.

To turn on GPU validation, add the following to your layer settings file, which is often named `vk_layer_settings.txt`.

```
lunarg_core_validation.gpu_validation = all
```
How GPU-Assisted Validation Works

Shader Module

SPIR-V code

spirv-opt
Perform instrumentation pass
(at CreateShaderModule)

instrumented SPIR-V code

Command Buffer

Queue Submit

Device Memory

Queue WaitIdle
Map and analyze debug buffer

Descriptor

Debug Buffer

Debug Report
Limitations

- **Vulkan 1.1 device required**
  - Instrumentation uses SPIR-V 1.3 features

- **Needs free descriptor set slot in Pipeline Layout**
  - Uses `maxBoundDescriptorSets - 1` slot if possible
  - Option to “reserve” a slot by faking `maxBoundDescriptorSets`

- **Other resource allocations count towards max**
  - device memory
  - descriptors

- **Limited to image/texel descriptors (for now)**
Future Work

- Validation of `VK_EXT_descriptor_indexing` extension usage
  - This extension relaxes restrictions on descriptor initialization
    - Bindless OOB checking of descriptor arrays of unknown size (`tex[]`)
    - Validating that descriptors in a descriptor array used by a shader have been updated by the application
    - Validate that application has updated descriptors in “Update after bind” case
  - Timeframe - April?

- Relaxing need for descriptor set binding slot
  - “Pipeline adaptive”
    - Find a descriptor binding based on pipeline layout
    - Requires recompilation of shaders
  - Use `VK_EXT_buffer_address` extension
    - Removes need for descriptor entirely
    - May not always be available
Credits

- **Greg Fischer (LunarG)**
  - Shader instrumentation
  - Overall design
- **Karl Schultz / Tony Barbour (LunarG)**
  - Layer design and implementation
- **Dan Ginsburg (Valve)**
  - Early design feedback and testing
More Information

- Vulkan SDK website (aka LunarXchange) https://vulkan.lunarg.com
- White Paper
  - https://www.lunarg.com/new-vulkan-gpu-assisted-validation-white-paper/
- Design Notes
  - GitHub: KhronosGroup/Vulkan-ValidationLayers/docs
  - https://github.com/KhronosGroup/Vulkan-ValidationLayers/blob/master/docs/gpu_validation.md
- Code
  - Shader Instrumentation
    - GitHub: KhronosGroup/SPIRV-Tools
  - Layer Implementation
    - GitHub: KhronosGroup/Vulkan-ValidationLayers
Buffer References

Jeff Bolz - NVIDIA
March 2019
Motivation

- Feature: “Access buffer memory through a 64-bit GPU address”

- Bindless buffer access
  - “Pointers” in shading language still a common feature request
  - Similar to GL_NV_shader_buffer_load/store
  - VK_EXT_descriptor_indexing isn’t a perfect substitute for this

- Conceptual compatibility with D3D12 root-level CBV/SRV/UAV
  - Query GPU address from the driver, pass it through to the shader, no descriptor sets involved

- Useful with VK_NV_ray_tracing
  - To store buffer addresses in the Shader Binding Table
Existing SSBO syntax

```glsl
#version 450

layout(set=0, binding=0, std430) buffer BufType0 {
    vec4 f[];
} b0;

layout(set=0, binding=1, std430) buffer BufType1 {
    uvec4 i[];
} b1;

void main() {
    int id = gl_LocalInvocationIndex;
    b0.f[id] = vec4(b1.i[id]);
}
```

Buffers all accessed through set/binding numbers

But what if you want to select the source and destination buffer dynamically?
Existing descriptor_indexing solution

```
#version 450
#extension GL_EXT_nonuniform_qualifier : enable

layout(set=0, binding=0, std430) buffer BufType0 {
    vec4 f[];
} b0[];

layout(set=0, binding=0, std430) buffer BufType1 {
    uvec4 i[];
} b1[];

layout(set=0, binding=1, std430) buffer Inputs {
    uint idx0;
    uint idx1;
} inputs;

void main() {
    int id = gl_LocalInvocationIndex;
    b0[idx0].f[id] = vec4(b1[idx1].i[id]);
}
```

Enable GL_EXT_nonuniform_qualifier

Runtime-sized array of buffers, all set to the same binding

New “input” buffer containing indices of the buffers we want to use

Select buffers by index
Buffer reference solution (new)

```glsl
#version 450
#extension GL_EXT_buffer_reference : enable

layout(buffer_reference, std430) buffer BufType0 {
  vec4 f[];
};

layout(buffer_reference, std430) buffer BufType1 {
  uvec4 i[];
};

layout(set=0, binding=1, std430) buffer Inputs {
  BufType0 b0;
  BufType1 b1;
} inputs;

void main() {
  int id = gl_LocalInvocationIndex;
  inputs.b0.f[id] = vec4(inputs.b1.i[id]);
}
```

- **buffer_reference** defines a reference type

- No variable name on block declaration means this is only a type declaration

- New “input” buffer containing references to the buffers we want to use

- Select buffers by reference
Approach

- Expressed in GLSL similarly to C++ reference types
  - Using the name “buffer reference” to describe them

- Limitations are similar to C++ references
  - No “pointer math” in shaders (*more on this later)
  - No unary ‘&’ operator
  - Always point to a complete block type (for layout information)
  - No pointers to intermediate or basic types
    - But you could always wrap those types in their own block definition

- More flexible than D3D12 root-level views
  - References can be stored anywhere in memory (e.g. build a linked list structure)
  - Supported in all shader stages
GLSL Extension

• New layout enables a reference type declaration
  - `layout(buffer_reference) buffer blockname { member-list } instancename_opt;`
  - Defines a reference type named “blockname” that acts like a reference to a structure type in buffer memory containing `member-list`
    - If there is no `instancename`, it is only a type declaration
    - If there is an `instancename`, it also declares a block as usual

• Forward declarations allowed
  - `layout(buffer_reference) buffer blockname; // no { member-list }`
  - Needed for a struct to include a pointer to itself (e.g. linked list)

• `layout(buffer_reference_align = N)` indicates the minimum alignment the block will be aligned to
  - Defaults to 16B
GLSL Extension

- Which accesses are affected by a qualifier is confusing (and maybe underspecified)
- Call qualifiers “Qi”, where each can be “coherent”, “volatile”, etc.

```glsl
layout(buffer_reference, set=1) Q1 buffer T0 {
    vec4 x;
};
layout(buffer_reference, binding=1) Q2 buffer T1 {
    Q3 T0 y;
} b1;

void main() {
    b1.y.x = ...;
    Q4 T0 y = b1.y;
    y.x = ...;
}
```

- Advice: If there is any ambiguity, use local variables to make the actual qualifiers used obvious!
Vulkan Extension

- **VK_EXT_buffer_device_address**
- Query a 64-bit device address from a VkBuffer

```c
bufferCreateInfo.usage |= VK_BUFFER_USAGE_SHADER_DEVICE_ADDRESS_BIT_EXT;
...
VkBufferDeviceAddressInfoEXT info = {
    VK_STRUCTURE_TYPE_BUFFER_DEVICE_ADDRESS_INFO_EXT,
    NULL,
    buffer
};
VkDeviceAddress addr = vkGetBufferDeviceAddressEXT(device, &info);
```

- **Trace capture/replay supported (optional feature)**
  - Replay tool can request a buffer be created with a particular address, saved from a previous run on the same hardware/driver
  - Drivers (probably) carve out a region of the address space to use for these buffers, to avoid the address being already in use
  - Most frequent feature request we had for GL_NV_shader_buffer_load
Pointer Math

• Current extension:
  - Can convert reference types to and from uint64_t
  - And can do arbitrary “pointer math” on the integer address values

• Upcoming extension:
  - Support “reference + integer” operations with C++-pointer-like result
    - \( b + i = (B)((\text{uint}64\_t)b + i \times \text{sizeof}(B)) \)
  - Support bracket array indexing
    - \( b[i] == b + i \)
  - Goal is to make it easier to use a reference that points to an array of blocks
    - Warning: \( b[i][j] == b[i+j] \)
      - We don’t have a distinction between “pointer” and “reference” types, so indexing the reference type returns the same reference type
  - Can implement \( \text{sizeof}(T) \) as a macro
    - #define \( \text{sizeof}(T) (\text{uint}64\_t(T(\text{uint}64\_t(0))+1)) \)
  - No new API/SPIR-V extension for this, entirely implemented in glslang
SPIR-V Extension

- **SPV_EXT_physical_storage_buffer**
  - Add new storage class `PhysicalStorageBufferEXT`
  - Pull in some existing SPIR-V instructions from OpenCL

- No new “reference” types, all implemented in terms of existing pointers

- New decorations for aliased/restrict on pointer variables

- (These details probably don’t affect you)
Status

- Specs/compiler/validation released in January
  - Links available from https://github.com/KhronosGroup/Vulkan-Docs/issues/890
  - Array indexing extension (WIP): https://github.com/KhronosGroup/GLSL/pull/61

- NVIDIA implementation available for all GPUs in Vulkan developer driver branch (https://developer.nvidia.com/vulkan-driver)
  - Will be in mainline drivers soon

- Intel
  - Linux: Available in mesa git master, will be in the 19.1 release
  - Windows: Work in progress to be available soon

- AMD
  - Working with developers to expose it as they move to use it