Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

Debugging your GPU Workflow

Spencer Fricke
LunarG, Inc.

Some quick information about me

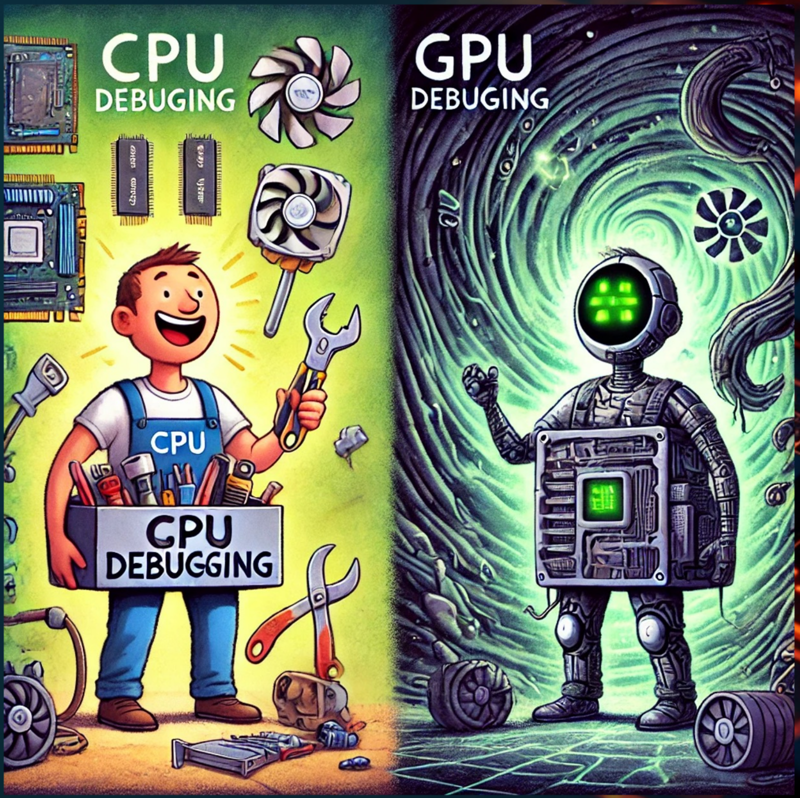
- Been working with Vulkan since 2017
- Been part of the Vulkan Working Group since 2019
- Been LunarG since October 2022
  - Technical Lead for the Validation Layer
  - Help maintain various SPIR-V tools
- Working on building GPU debugging tools over the last year
- Will talk to you forever afterwards about this presentation or anything Vulkan related
  - (personal disclaimer)

LUNARG

Table of Contents

Goals of this talk

- Showcase various ways to use Vulkan API to debug
- Show tools to help debug
- Show how it easy it is to use the tools!



(Generated by DALL-E)

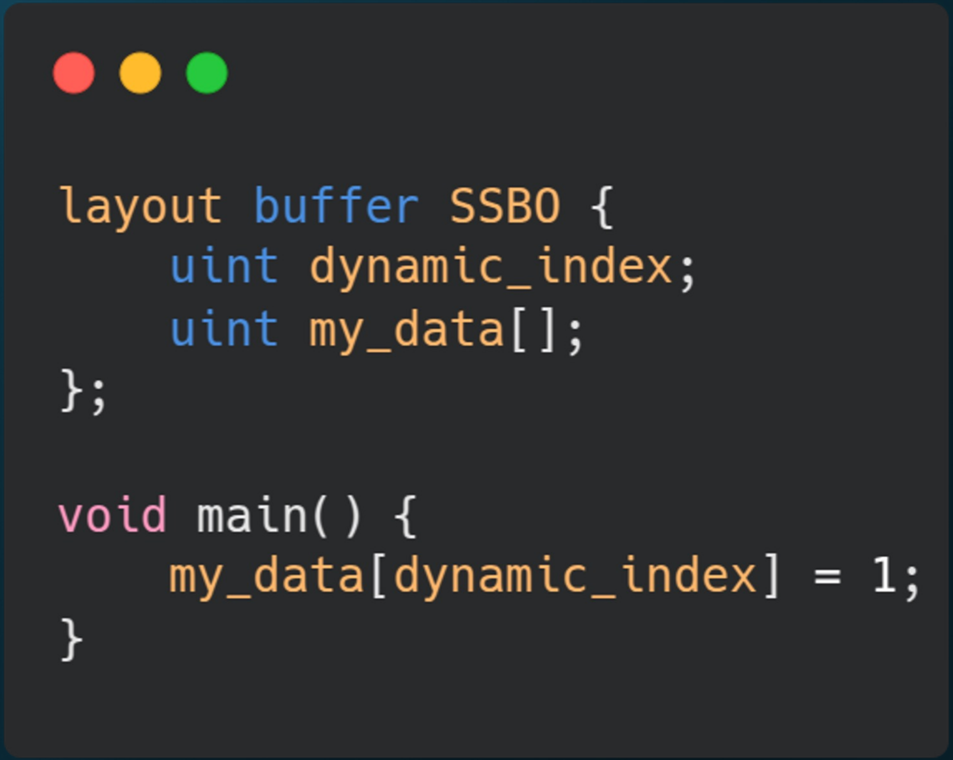
When debugging on the GPU, you want all the tools!

(Generated by DALL-E)

「Toolbox」 Robustness

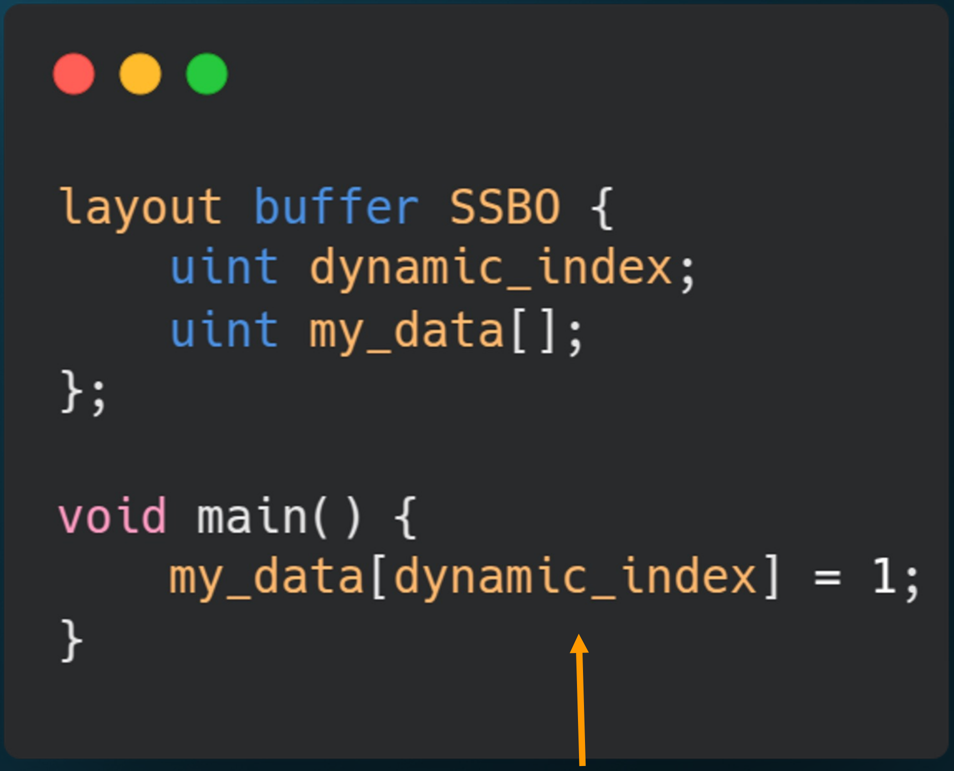
- Features that can be enabled at device creation time
  - Some implementations will have slight performance reduction when feature is enabled
- Will prevent various out-of-bounds from crashing
- More info: https://docs.vulkan.org/guide/latest/robustness.html

「Toolbox」 Robustness – example



```
layout buffer SSBO {
    uint dynamic_index;
    uint my_data[];
};

void main() {
    my_data[dynamic_index] = 1;
}
```

「Toolbox」 Robustness - example



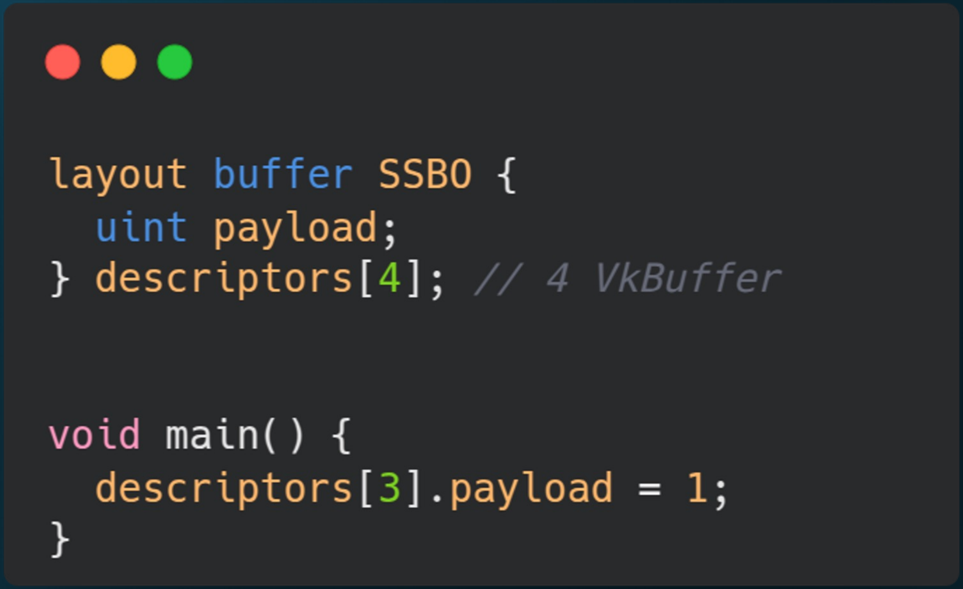
```
layout buffer SSBO {
    uint dynamic_index;
    uint my_data[];
};

void main() {
    my_data[dynamic_index] = 1;
}
```

Might be larger than the bound VkBuffer!

「Toolbox」 Robustness - example

- Enable VkPhysicalDeviceFeatures::robustBufferAccess
  - OOB writes are ignored
  - OOB loads return zero
- Did the crash/issue go away?
  - If it did, likely found source of issue
  - Can catch various OOB issue

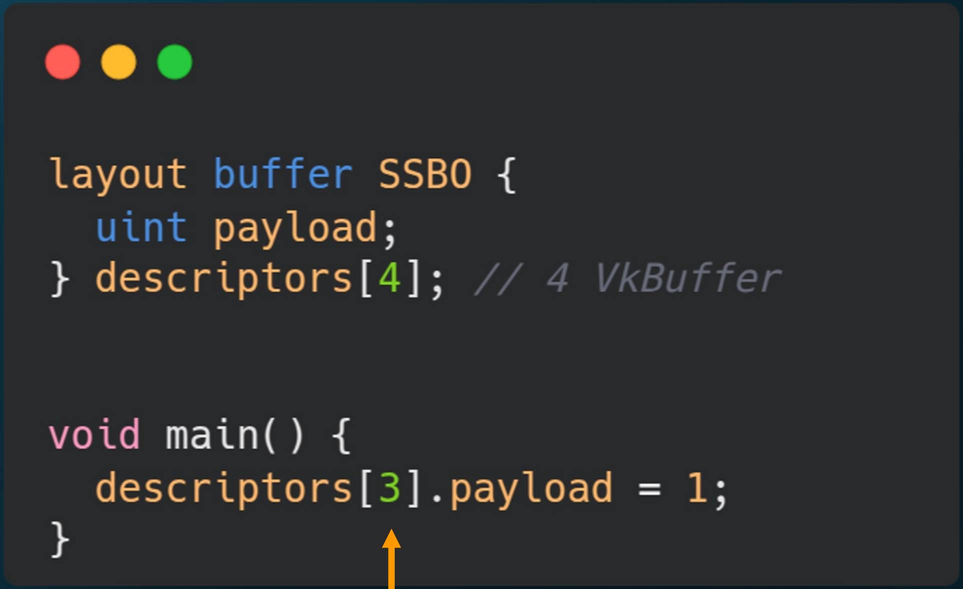
「Toolbox」 Robustness - example 2



```
layout buffer SSBO {
  uint payload;
} descriptors[4]; // 4 VkBuffer


void main() {
  descriptors[3].payload = 1;
}
```

LUNARG

「Toolbox」 Robustness - example 2



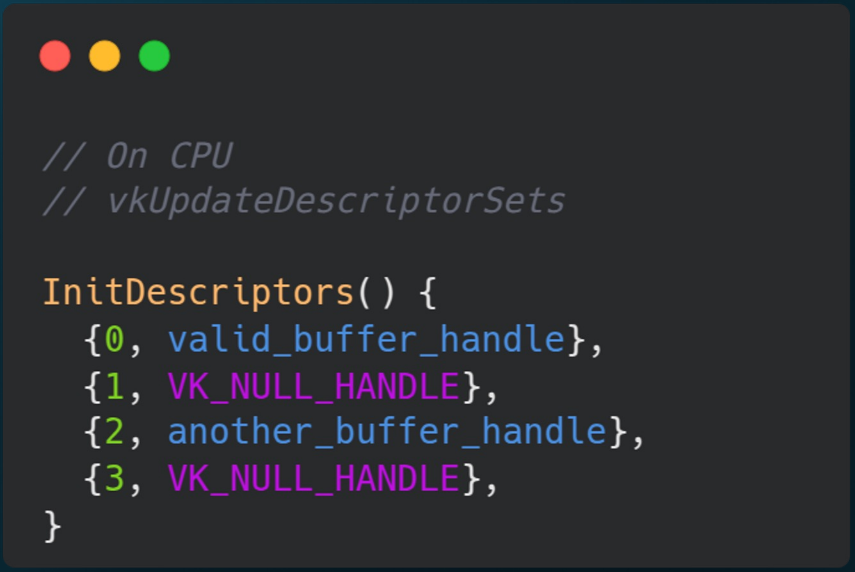
```
layout buffer SSBO {
  uint payload;
} descriptors[4]; // 4 VkBuffer


void main() {
  descriptors[3].payload = 1;
}
```

What if you forgot to bind a buffer?
(hint, it will hang most devices)

LUNARG

「Toolbox」 Robustness – example 2

- Enable VkPhysicalDeviceRobustness2FeaturesEXT::nullDescriptor
- Initialize everything with VK_NULL_HANDLE

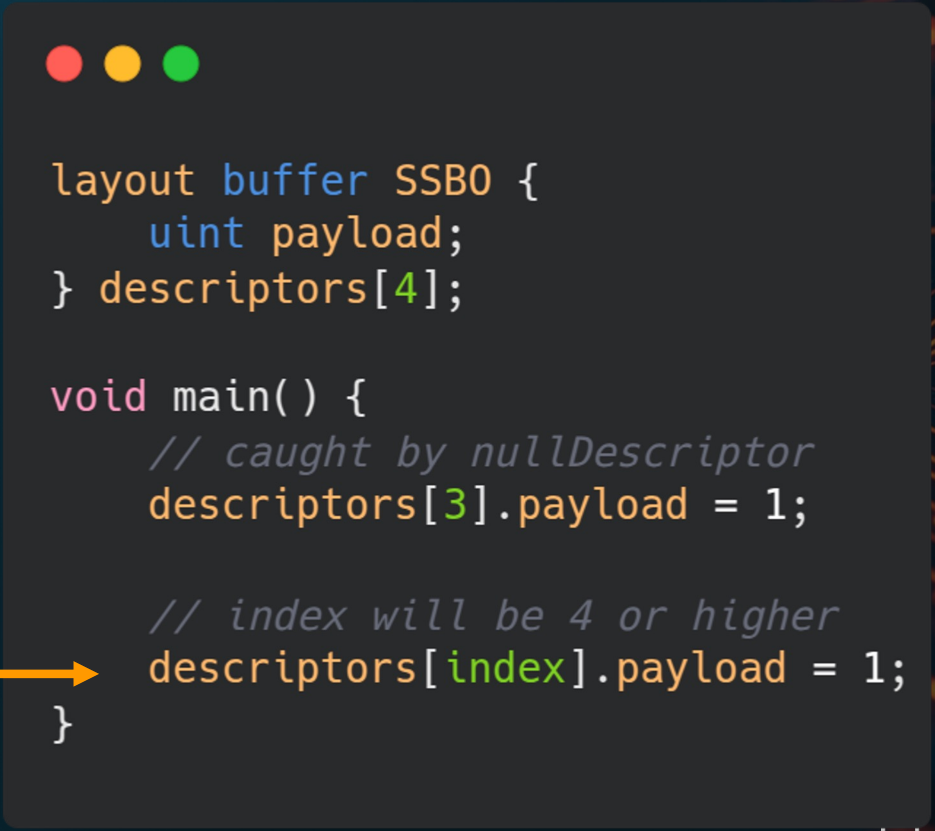


```
// On CPU
// vkUpdateDescriptorSets

InitDescriptors() {
  {0, valid_buffer_handle},
  {1, VK_NULL_HANDLE},
  {2, another_buffer_handle},
  {3, VK_NULL_HANDLE},
}
```

Robustness limitations

Very subtle difference!

Going to need another tool!



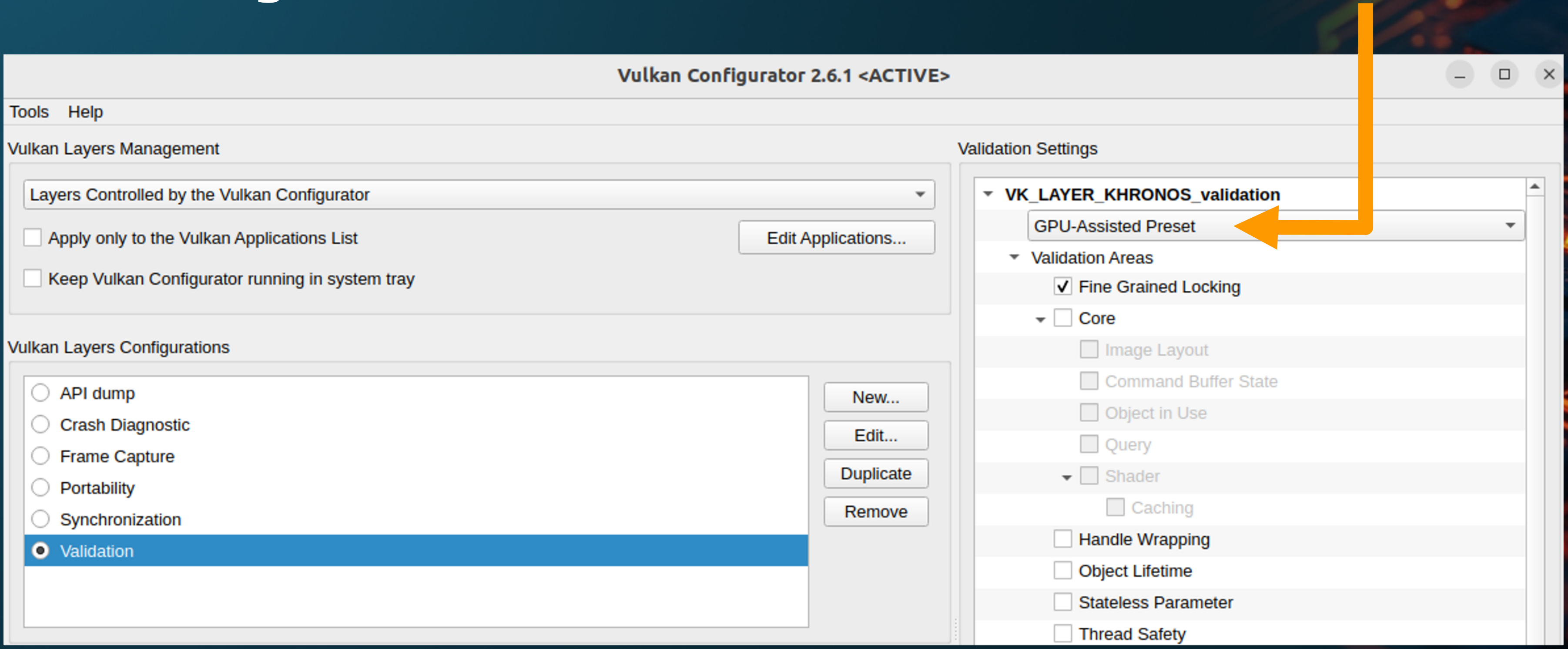
```
layout buffer SSBO {
    uint payload;
} descriptors[4];

void main() {
    // caught by nullDescriptor
    descriptors[3].payload = 1;

    // index will be 4 or higher
    descriptors[index].payload = 1;
}
```

「Toolbox」 GPU-AV

- GPU Assisted Validation
  - Optional setting in the Validation Layers
- Everything that can’t be detected on the CPU
- Add “hooks” to see what the GPU is doing at runtime and report back
- Has been top priority for us over the last year
  - we are only getting more and more GPU Centric world now.

「Toolbox」 GPU-AV – turning on*

- VkConfig (recommended)



Vulkan Configurator 2.6.1 <ACTIVE>

Tools Help

Vulkan Layers Management

Layers Controlled by the Vulkan Configurator

- [ ] Apply only to the Vulkan Applications List
- [ ] Keep Vulkan Configurator running in system tray

Edit Applications...

Vulkan Layers Configurations

- ○ API dump
- ○ Crash Diagnostic
- ○ Frame Capture
- ○ Portability
- ○ Synchronization
- ● Validation

New... Edit... Duplicate Remove

Validation Settings

VK_LAYER_KHRONOS_validation

GPU-Assisted Preset

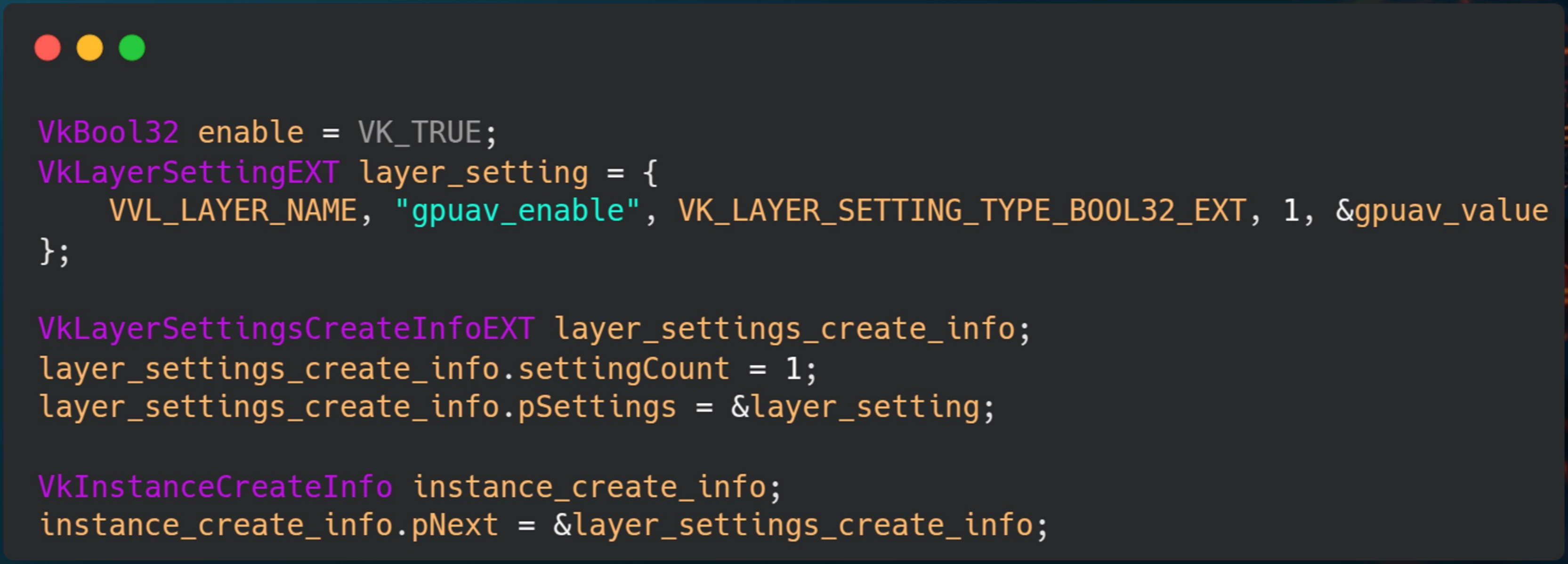
Validation Areas

- [✓] Fine Grained Locking
- [ ] Core
  - [ ] Image Layout
  - [ ] Command Buffer State
  - [ ] Object in Use
  - [ ] Query
  - [ ] Shader
    - [ ] Caching
- [ ] Handle Wrapping
- [ ] Object Lifetime
- [ ] Stateless Parameter
- [ ] Thread Safety

*Based on the 1.4.304 SDK

「Toolbox」 GPU-AV – turning on*

- VkConfig (recommended)
- VK_EXT_layer_settings



```
VkBool32 enable = VK_TRUE;
VkLayerSettingEXT layer_setting = {
    VVL_LAYER_NAME, "gpuav_enable", VK_LAYER_SETTING_TYPE_BOOL32_EXT, 1, &gpuav_value
};

VkLayerSettingsCreateInfoEXT layer_settings_create_info;
layer_settings_create_info.settingCount = 1;
layer_settings_create_info.pSettings = &layer_setting;

VkInstanceCreateInfo instance_create_info;
instance_create_info.pNext = &layer_settings_create_info;
```

*Based on the 1.4.304 SDK

「Toolbox」 GPU-AV – turning on*

- VkConfig (recommended)
- VK_EXT_layer_settings
- Environment variables
  - **Warning** – this will still have Core Validation and will be extra slow

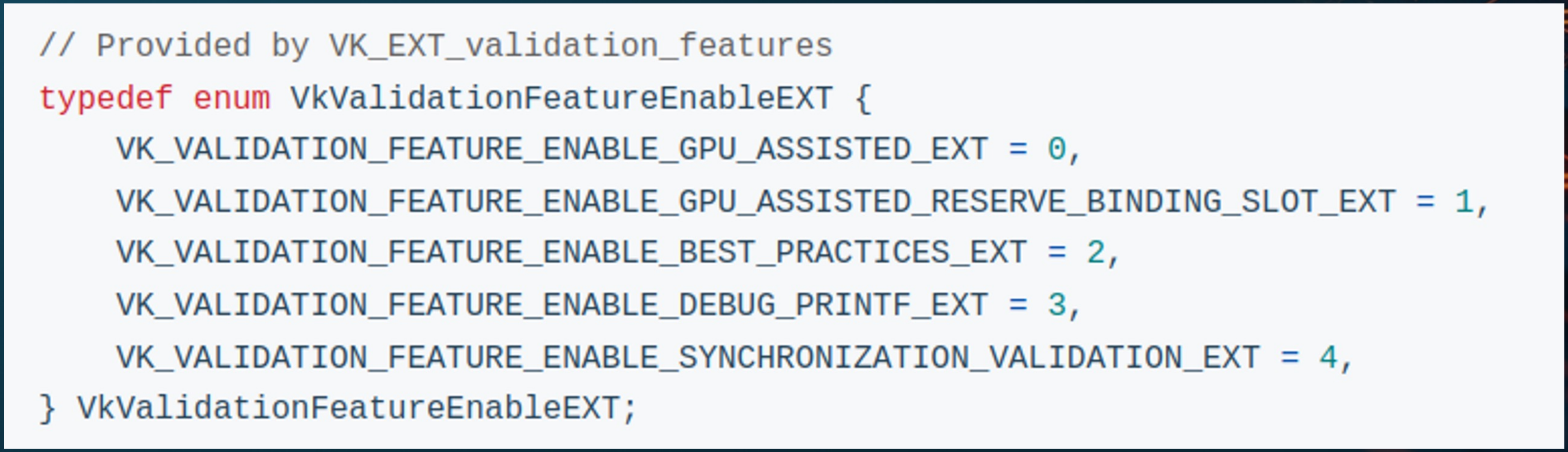
```
VK_LAYER_GPUAV_ENABLE=1 ./myApp
```

*Based on the 1.4.304 SDK

LUNARG

「Toolbox」 GPU-AV – turning on*

- VkConfig (recommended)
- VK_EXT_layer_settings
- Environment variables
- VK_EXT_validation_features (deprecated)

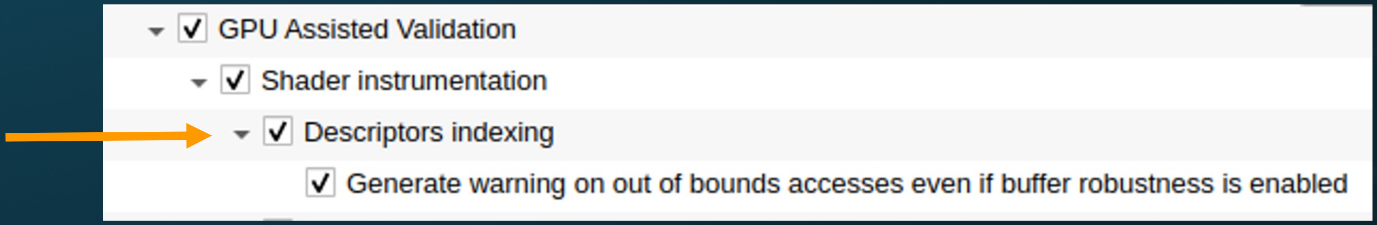


```
// Provided by VK_EXT_validation_features
typedef enum VkValidationFeatureEnableEXT {
    VK_VALIDATION_FEATURE_ENABLE_GPU_ASSISTED_EXT = 0,
    VK_VALIDATION_FEATURE_ENABLE_GPU_ASSISTED_RESERVE_BINDING_SLOT_EXT = 1,
    VK_VALIDATION_FEATURE_ENABLE_BEST_PRACTICES_EXT = 2,
    VK_VALIDATION_FEATURE_ENABLE_DEBUG_PRINTF_EXT = 3,
    VK_VALIDATION_FEATURE_ENABLE_SYNCHRONIZATION_VALIDATION_EXT = 4,
} VkValidationFeatureEnableEXT;
```

*Based on the 1.4.304 SDK

「Toolbox」 GPU-AV - Descriptor Indexing

- Detects OOB descriptor index accesses
- Detects if descriptor is uninitialized or destroyed
- Detects if the descriptor itself is valid
  - Ex. Storage buffer is not accessed OOB
  - Ex. A 3D image accessed is bound to a VkImage with VK_IMAGE_TYPE_3D

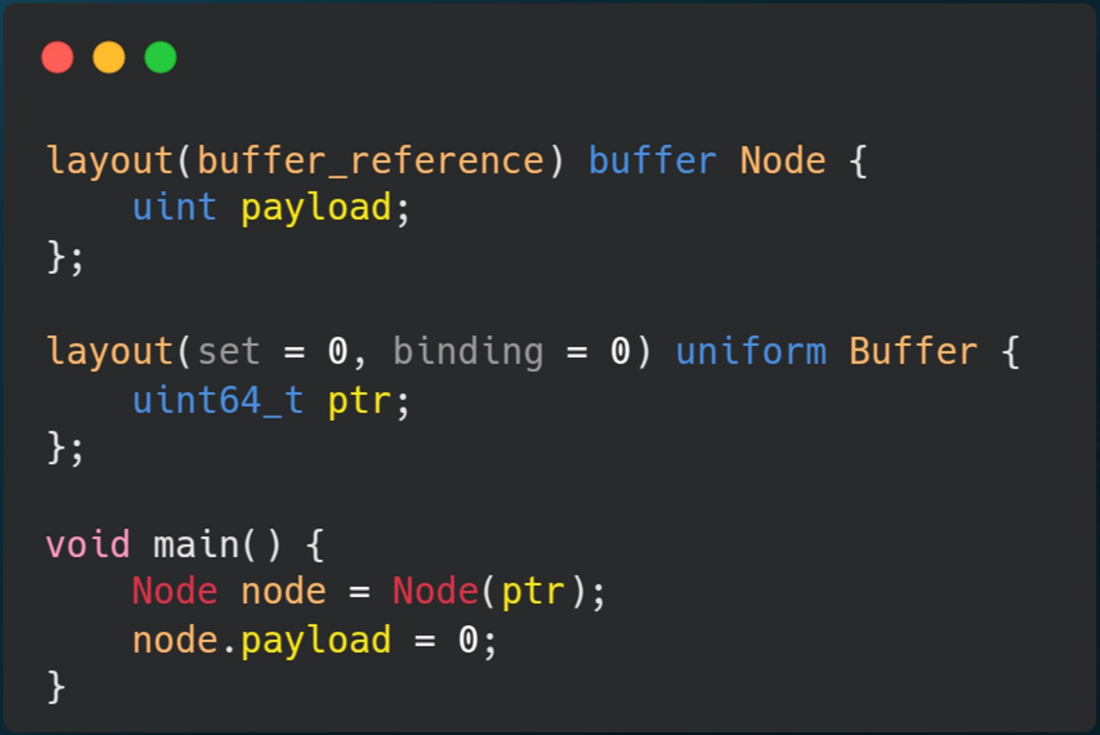

- ✓ GPU Assisted Validation
  - ✓ Shader instrumentation
    - ✓ Descriptors indexing
      - ✓ Generate warning on out of bounds accesses even if buffer robustness is enabled

(Options from VkConfig)

「Toolbox」 GPU-AV - Buffer Device Address

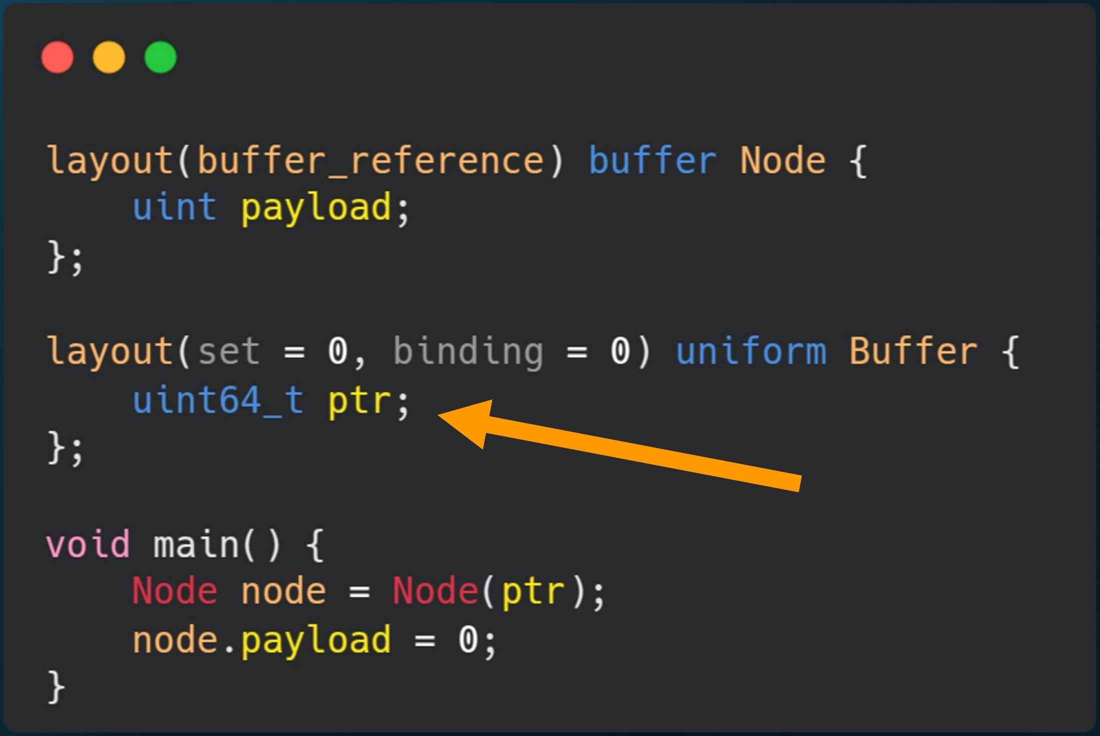
- aka – buffer reference
- aka – PhysicalStorageBuffer (SPV_KHR_physical_storage_buffer)
- aka – pointers in your shader

「Toolbox」 GPU-AV - Buffer Device Address



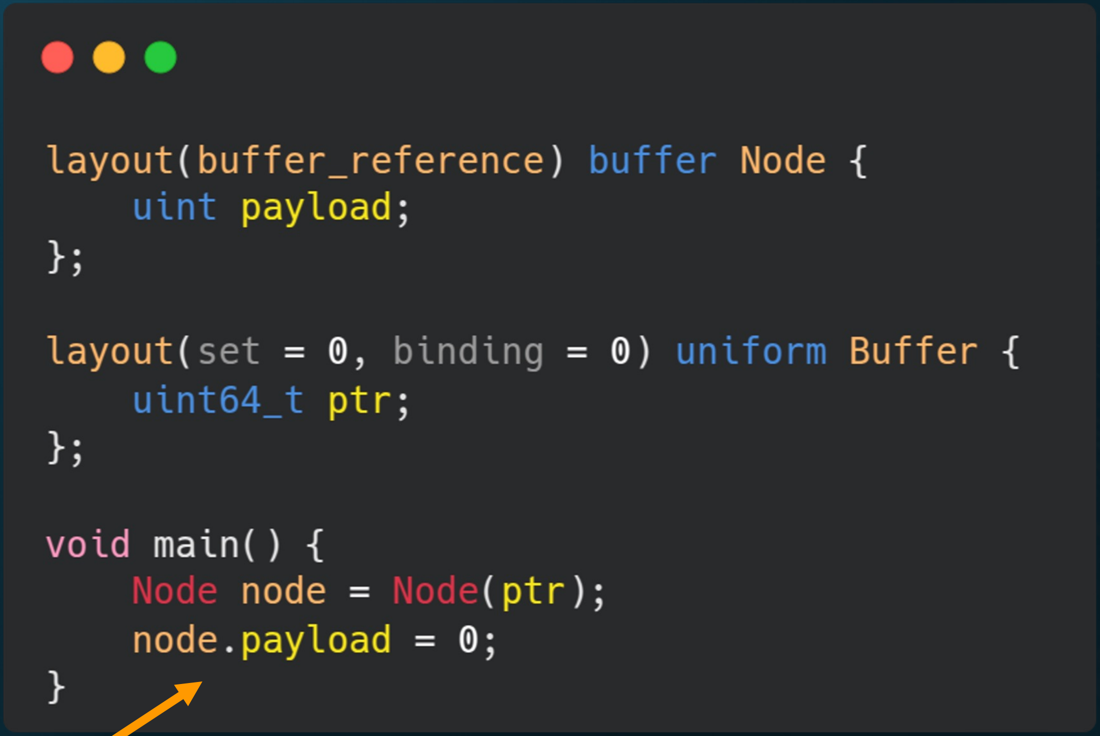
```glsl
layout(buffer_reference) buffer Node {
    uint payload;
};

layout(set = 0, binding = 0) uniform Buffer {
    uint64_t ptr;
};

void main() {
    Node node = Node(ptr);
    node.payload = 0;
}
```

「Toolbox」 GPU-AV - Buffer Device Address



```
layout(buffer_reference) buffer Node {
    uint payload;
};

layout(set = 0, binding = 0) uniform Buffer {
    uint64_t ptr;
};

void main() {
    Node node = Node(ptr);
    node.payload = 0;
}
```

「Toolbox」 GPU-AV - Buffer Device Address

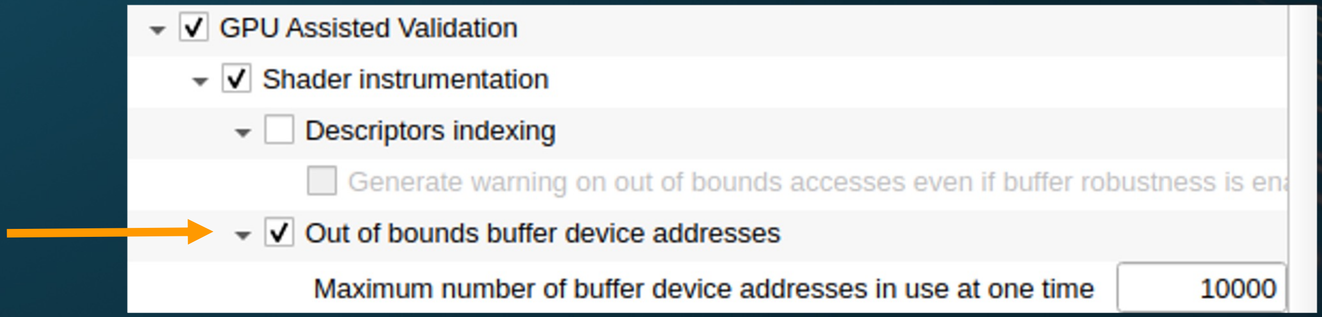


```
layout(buffer_reference) buffer Node {
    uint payload;
};

layout(set = 0, binding = 0) uniform Buffer {
    uint64_t ptr;
};

void main() {
    Node node = Node(ptr);
    node.payload = 0;
}
```

“dereferencing” the pointer

「Toolbox」 GPU-AV - Buffer Device Address

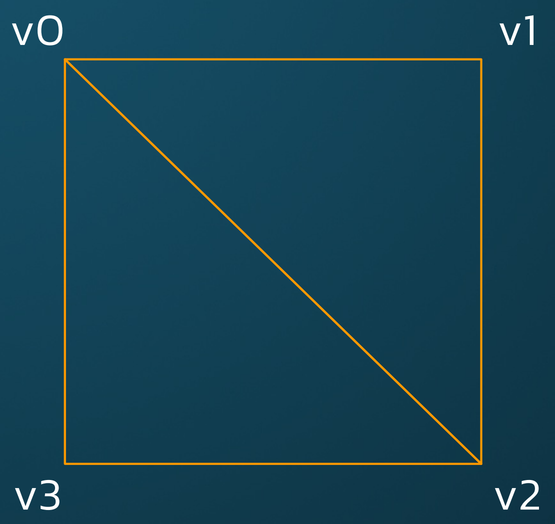
- Detects if address is not properly aligned
- Detects if address is not inside a valid VkBuffer range



- [✓] GPU Assisted Validation
  - [✓] Shader instrumentation
    - [ ] Descriptors indexing
      - [ ] Generate warning on out of bounds accesses even if buffer robustness is ena
    - [✓] Out of bounds buffer device addresses
      - Maximum number of buffer device addresses in use at one time: 10000

(Options from VkConfig)

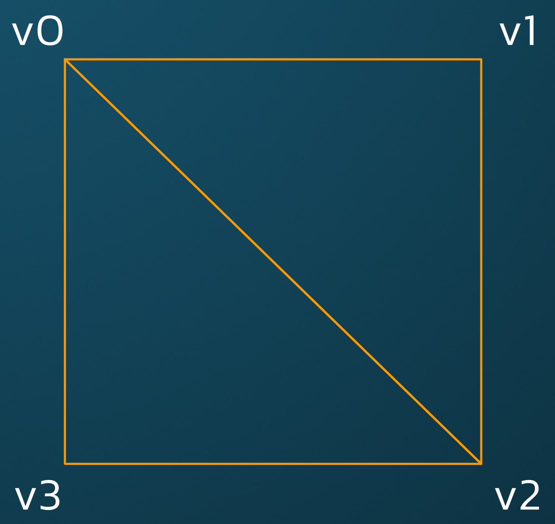
Index Buffer gone wrong

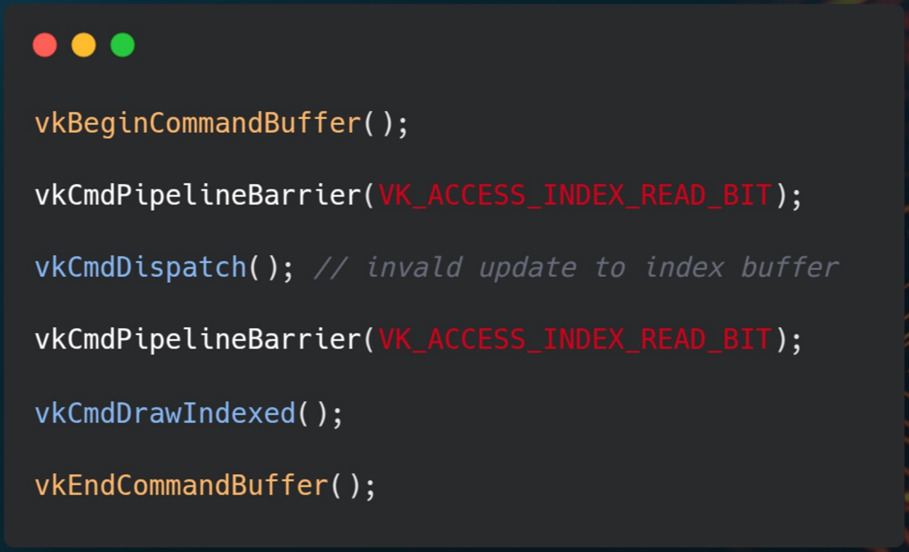


Index Buffer

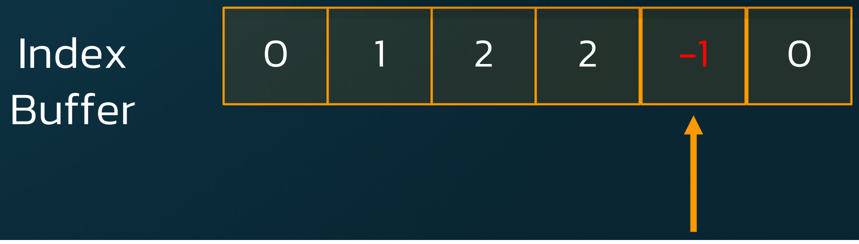
0	1	2	2	3	0

Index Buffer gone wrong



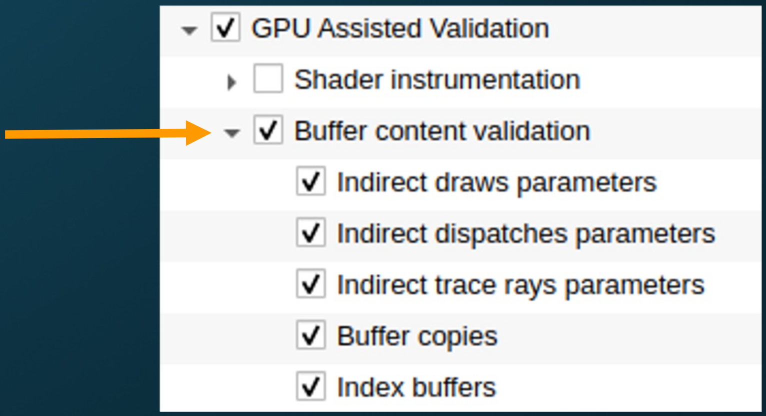


```
vkBeginCommandBuffer();

vkCmdPipelineBarrier(VK_ACCESS_INDEX_READ_BIT);

vkCmdDispatch(); // invald update to index buffer

vkCmdPipelineBarrier(VK_ACCESS_INDEX_READ_BIT);

vkCmdDrawIndexed();

vkEndCommandBuffer();
```



「Toolbox」 GPU-AV - Buffer Contents

- Detects invalid indirect draw/dispatch/traceRay parameters
- Detects if depth/stencil buffer copies have invalid contents
- Detects invalid values inside index buffer



- [✓] GPU Assisted Validation
  - [ ] Shader instrumentation
  - [✓] Buffer content validation
    - [✓] Indirect draws parameters
    - [✓] Indirect dispatches parameters
    - [✓] Indirect trace rays parameters
    - [✓] Buffer copies
    - [✓] Index buffers

(Options from VkConfig)

「Toolbox」 GPU-AV is still growing!

- Many new things planned for 2025
  - Making it faster!
  - Better error messages!
  - VK_EXT_descriptor_buffer
  - VK_EXT_device_generated_commands
  - Ray Tracing
  - Mesh Shaders
- Please contact me (after the talk, Github issue, email, knocking on my front door) with GPU Validation you would find helpful!
  - Important to know which things to focus on first

GPU-AV (and other tools) limitations

- Can’t read/examine your source code
    - Can use your variable names

GPU-AV (and other tools) limitations

- Can't read/examine your source code
  - Can use your variable names
- Can't read your mind
  - Don't know why you are trying to do what you are doing

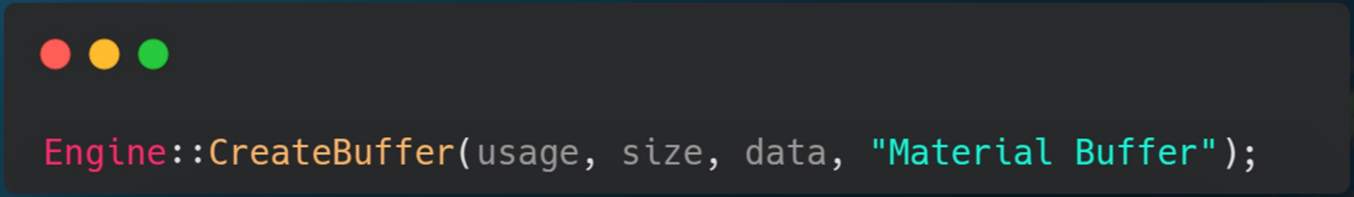
「Toolbox」 VK_EXT_debug_utils

- Give names to Vulkan handles
- Give names to section of command buffer
- Give name to VkQueue
- Will print out in Validation Layer errors (and other tools!)

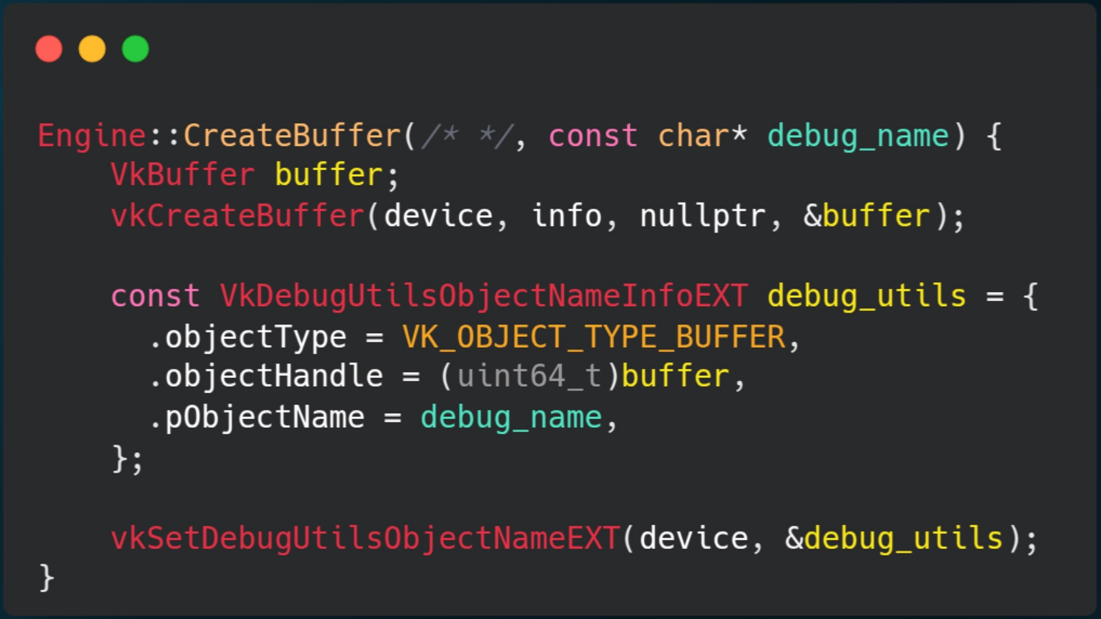
「Toolbox」 VK_EXT_debug_utils

```
Engine::CreateBuffer(usage, size, data, "Material Buffer");
```

「Toolbox」 VK_EXT_debug_utils

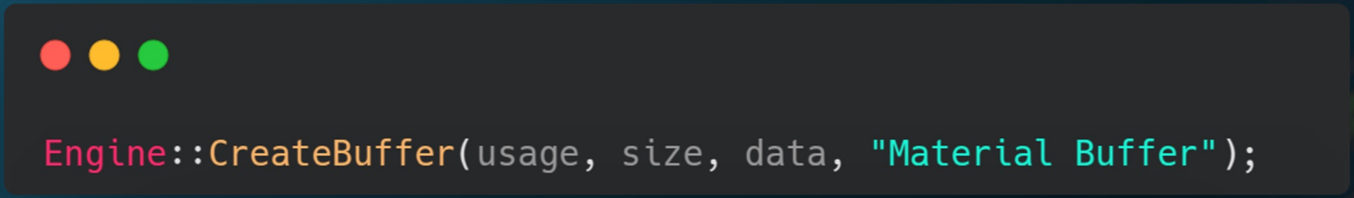


```
Engine::CreateBuffer(usage, size, data, "Material Buffer");
```

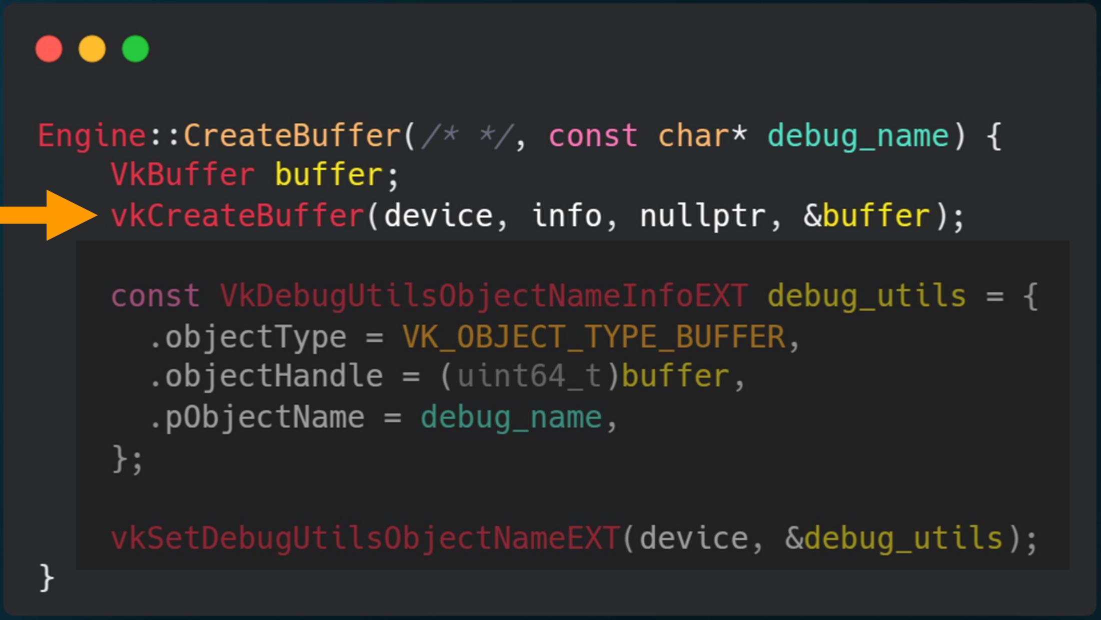


```
Engine::CreateBuffer(/* */, const char* debug_name) {
    VkBuffer buffer;
    vkCreateBuffer(device, info, nullptr, &buffer);

    const VkDebugUtilsObjectNameInfoEXT debug_utils = {
      .objectType = VK_OBJECT_TYPE_BUFFER,
      .objectHandle = (uint64_t)buffer,
      .pObjectName = debug_name,
    };

    vkSetDebugUtilsObjectNameEXT(device, &debug_utils);
}
```

「Toolbox」 VK_EXT_debug_utils

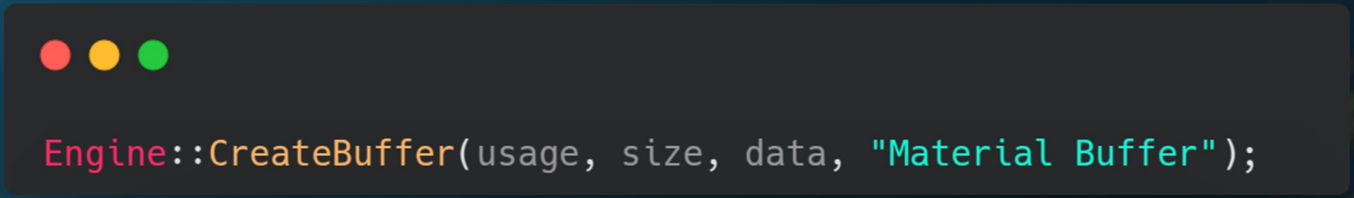


```
Engine::CreateBuffer(usage, size, data, "Material Buffer");
```

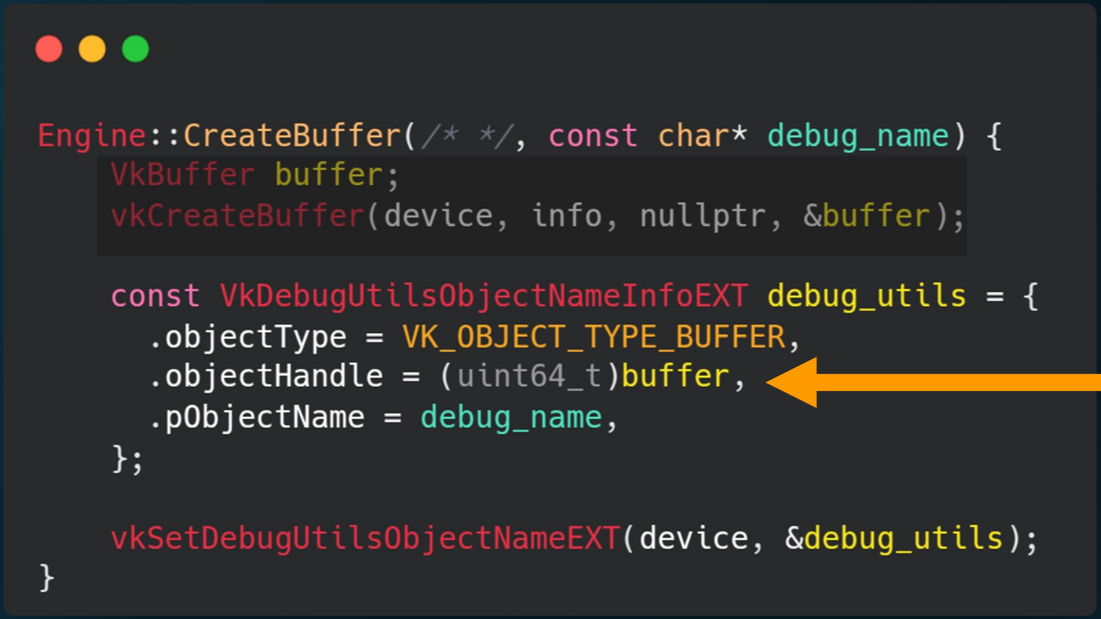


```
Engine::CreateBuffer(/* */, const char* debug_name) {
    VkBuffer buffer;
    vkCreateBuffer(device, info, nullptr, &buffer);

    const VkDebugUtilsObjectNameInfoEXT debug_utils = {
      .objectType = VK_OBJECT_TYPE_BUFFER,
      .objectHandle = (uint64_t)buffer,
      .pObjectName = debug_name,
    };

    vkSetDebugUtilsObjectNameEXT(device, &debug_utils);
}
```

「Toolbox」 VK_EXT_debug_utils



```
Engine::CreateBuffer(usage, size, data, "Material Buffer");
```



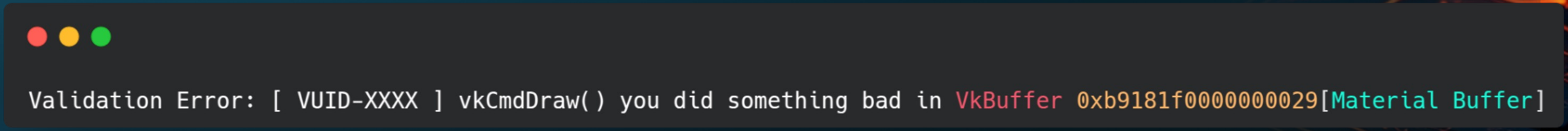
```
Engine::CreateBuffer(/* */, const char* debug_name) {
    VkBuffer buffer;
    vkCreateBuffer(device, info, nullptr, &buffer);

    const VkDebugUtilsObjectNameInfoEXT debug_utils = {
      .objectType = VK_OBJECT_TYPE_BUFFER,
      .objectHandle = (uint64_t)buffer,
      .pObjectName = debug_name,
    };

    vkSetDebugUtilsObjectNameEXT(device, &debug_utils);
}
```

「Toolbox」 VK_EXT_debug_utils

```
Engine::CreateBuffer(usage, size, data, "Material Buffer");
```

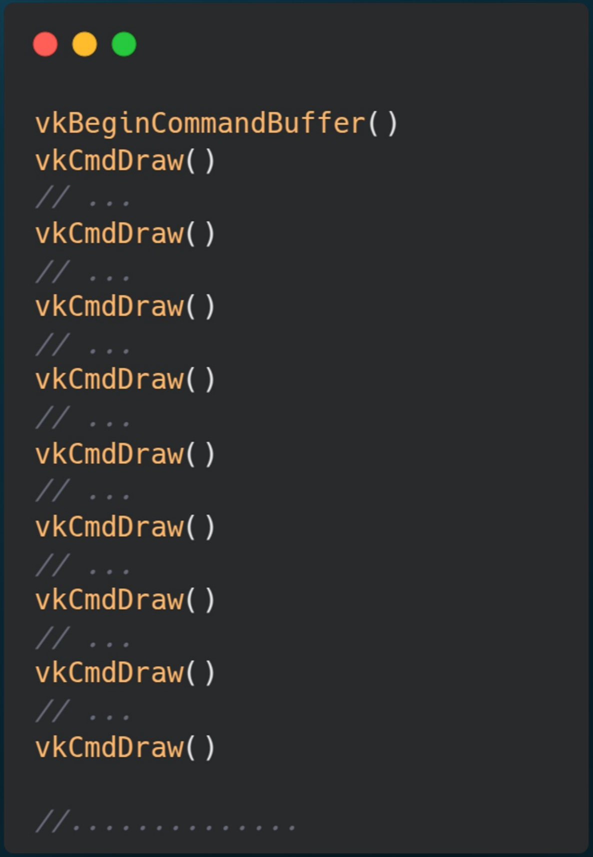
```
Engine::CreateBuffer(/* */, const char* debug_name) {
    VkBuffer buffer;
    vkCreateBuffer(device, info, nullptr, &buffer);

    const VkDebugUtilsObjectNameInfoEXT debug_utils = {
      .objectType = VK_OBJECT_TYPE_BUFFER,
      .objectHandle = (uint64_t)buffer,
      .pObjectName = debug_name,
    };

    vkSetDebugUtilsObjectNameEXT(device, &debug_utils);
}
```

「Toolbox」 VK_EXT_debug_utils



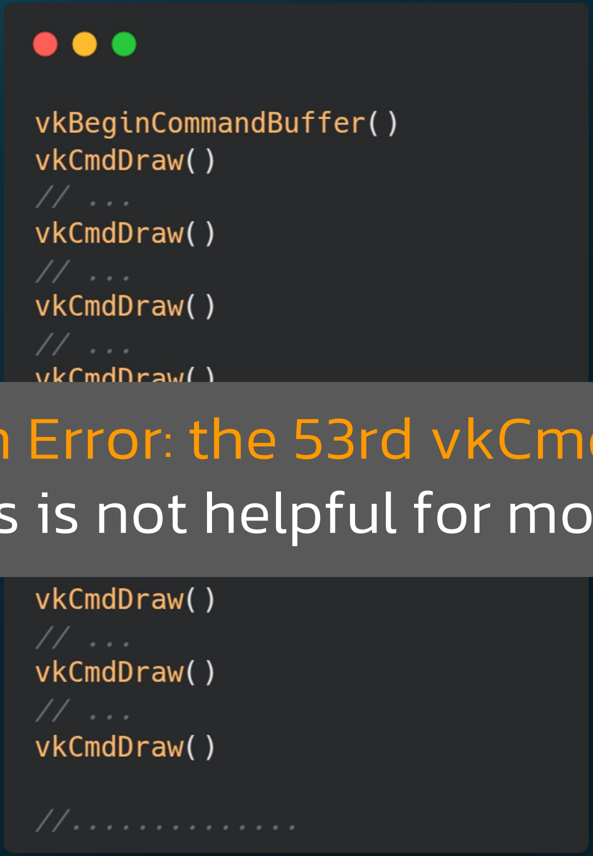
```
Validation Error: [ VUID-XXXX ] vkCmdDraw() you did something bad in VkBuffer 0xb9181f0000000029[Material Buffer]
```

Some people do LOTS of draws



```
vkBeginCommandBuffer()
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()

//..............
```

Some people do LOTS of draws

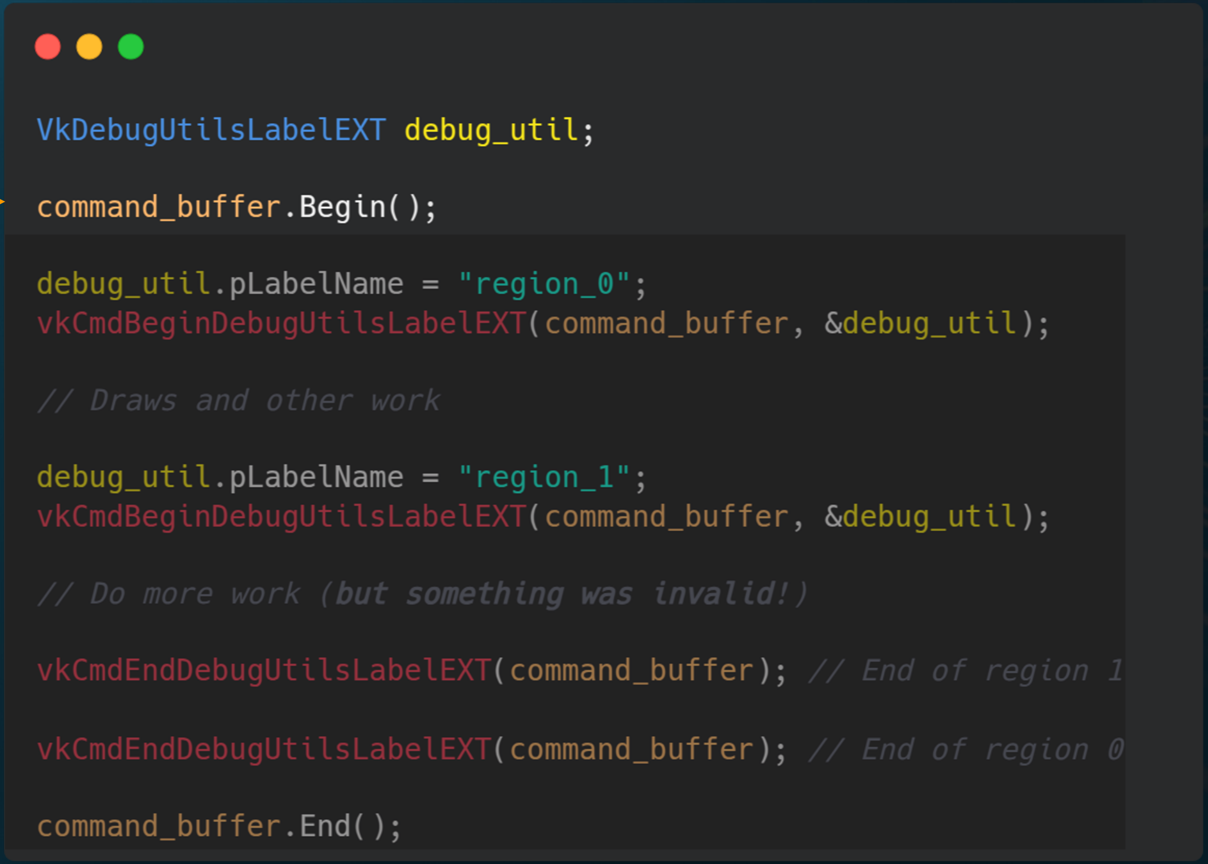


```
vkBeginCommandBuffer()
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()
```

```
vkCmdDraw()
// ...
vkCmdDraw()
// ...
vkCmdDraw()

//..............
```

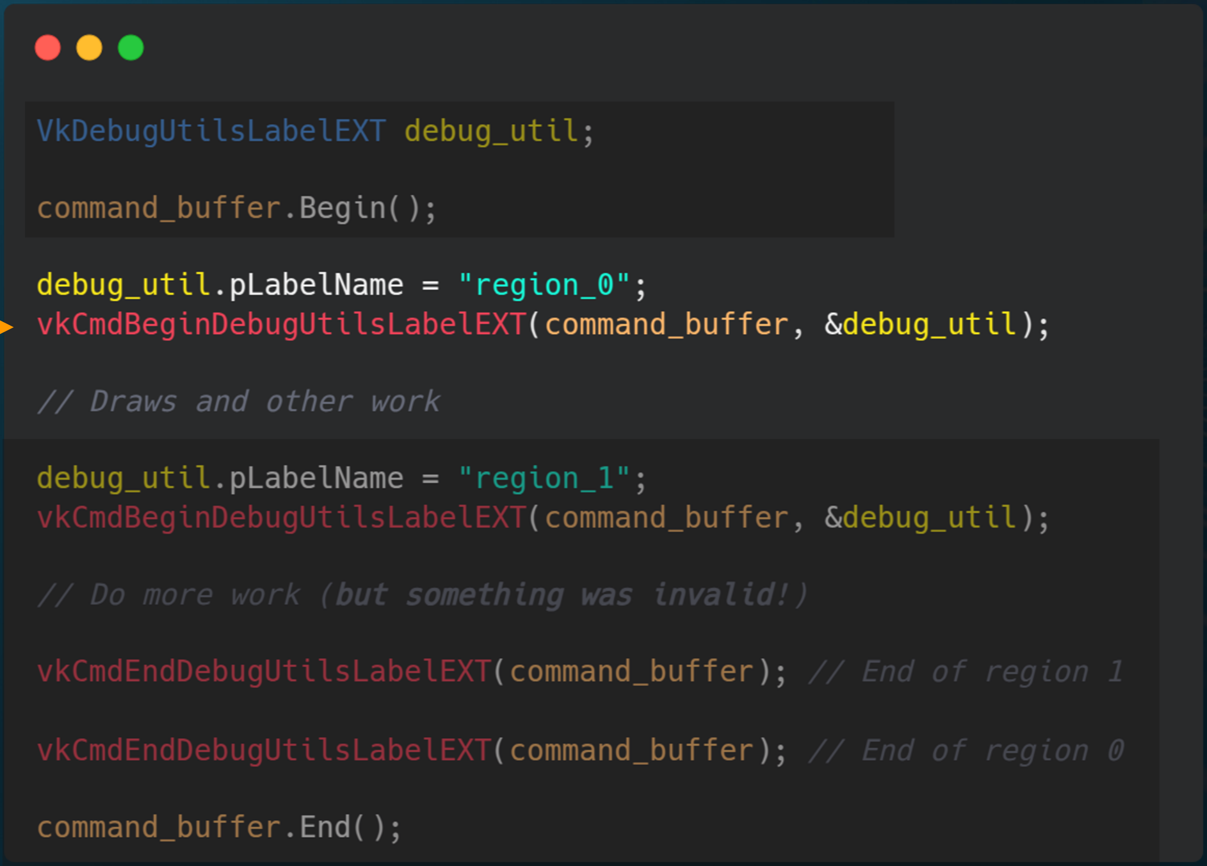
Validation Error: the 53rd vkCmdDraw() was bad
(this is not helpful for most people)

「Toolbox」 VK_EXT_debug_utils



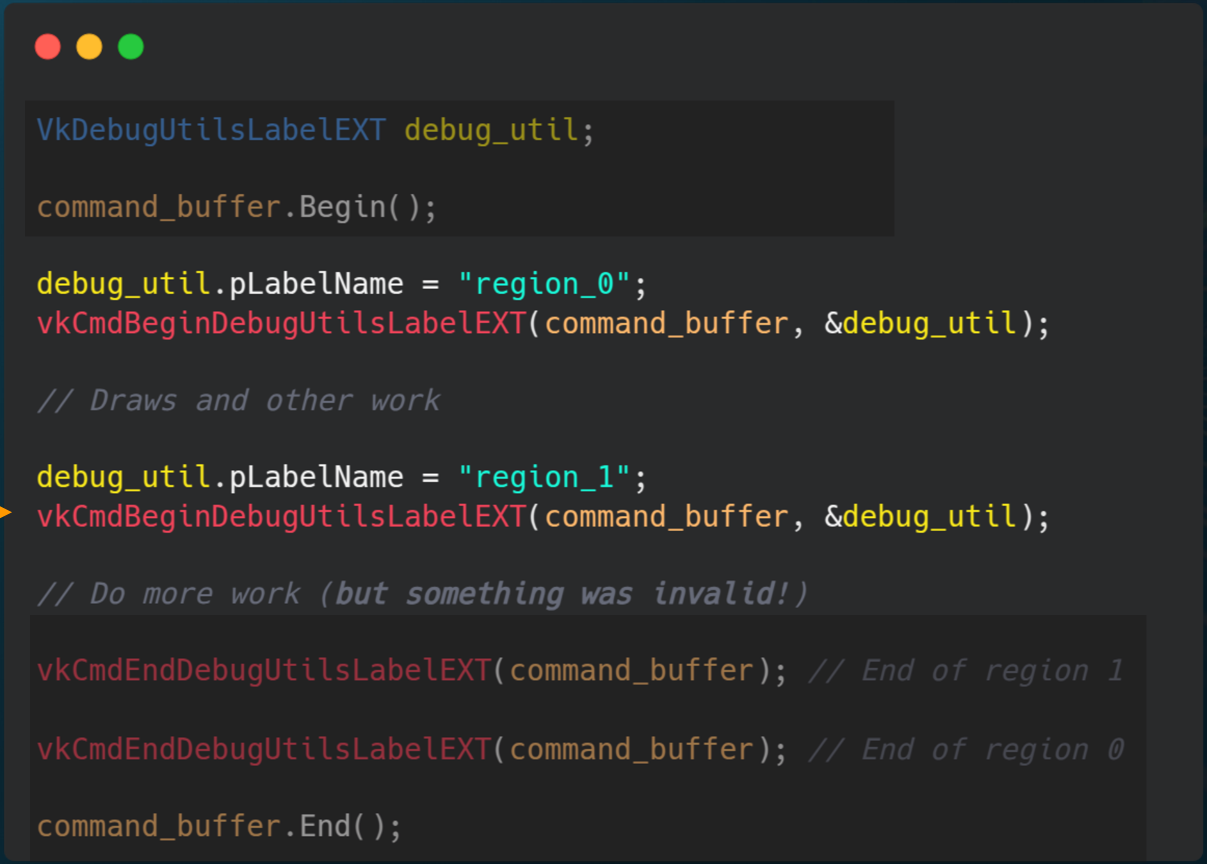
```
VkDebugUtilsLabelEXT debug_util;

command_buffer.Begin();

debug_util.pLabelName = "region_0";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Draws and other work

debug_util.pLabelName = "region_1";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Do more work (but something was invalid!)

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 1

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 0

command_buffer.End();
```

「Toolbox」 VK_EXT_debug_utils



```
VkDebugUtilsLabelEXT debug_util;

command_buffer.Begin();

debug_util.pLabelName = "region_0";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Draws and other work

debug_util.pLabelName = "region_1";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Do more work (but something was invalid!)

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 1

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 0

command_buffer.End();
```

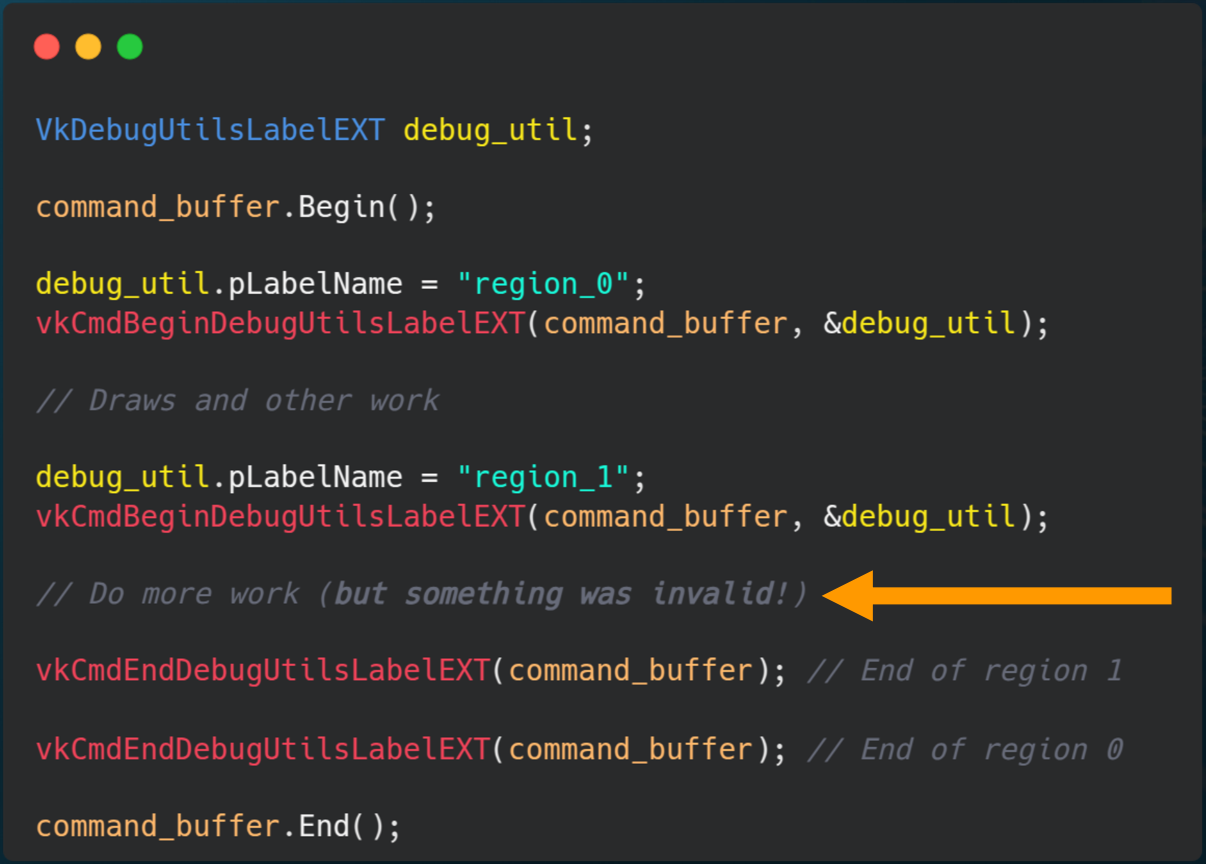
「Toolbox」 VK_EXT_debug_utils



```
VkDebugUtilsLabelEXT debug_util;

command_buffer.Begin();

debug_util.pLabelName = "region_0";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Draws and other work

debug_util.pLabelName = "region_1";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Do more work (but something was invalid!)

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 1

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 0

command_buffer.End();
```

LUNARG

「Toolbox」 VK_EXT_debug_utils



```
VkDebugUtilsLabelEXT debug_util;

command_buffer.Begin();

debug_util.pLabelName = "region_0";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Draws and other work

debug_util.pLabelName = "region_1";
vkCmdBeginDebugUtilsLabelEXT(command_buffer, &debug_util);

// Do more work (but something was invalid!)

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 1

vkCmdEndDebugUtilsLabelEXT(command_buffer); // End of region 0

command_buffer.End();
```

LUNARG

「Toolbox」 VK_EXT_debug_utils

```
Validation Error: [ VUID-XXXX ] [ Debug region: region_0::region_1 ] vkCmdDraw() you did something bad
```

Quick Question

Who here debugs their CPU stack traces with release builds? (instead of using debug or release-with-debug-info)

Quick Question

Who here debugs their CPU stack traces with release builds? (instead of using debug or release-with-debug-info)

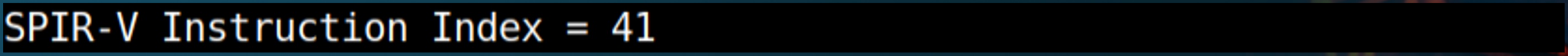
The GPU should **not** be any different!

LUNARG

「Toolbox」 NonSemantic Shader DebugInfo

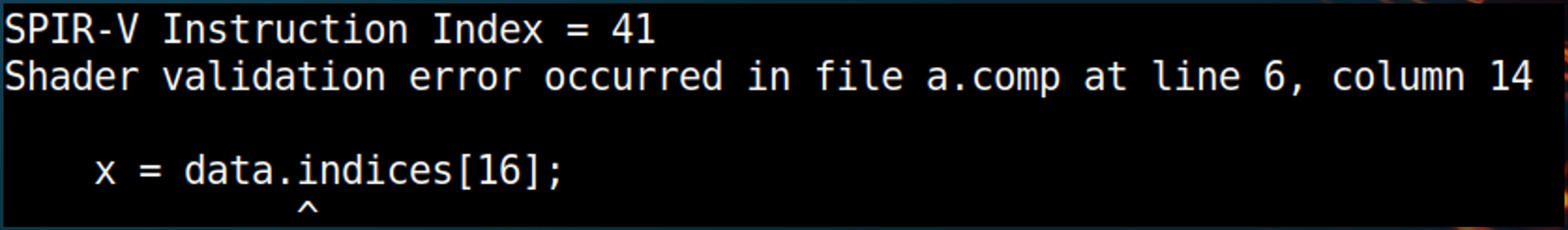
- Allows SPIR-V to be mapped back out to source language
  - This is what allows RenderDoc to let you step through a shader
  - Allows Validation Layers to report source code in error message

「Toolbox」 NonSemantic Shader DebugInfo



```
SPIR-V Instruction Index = 41
```

VS



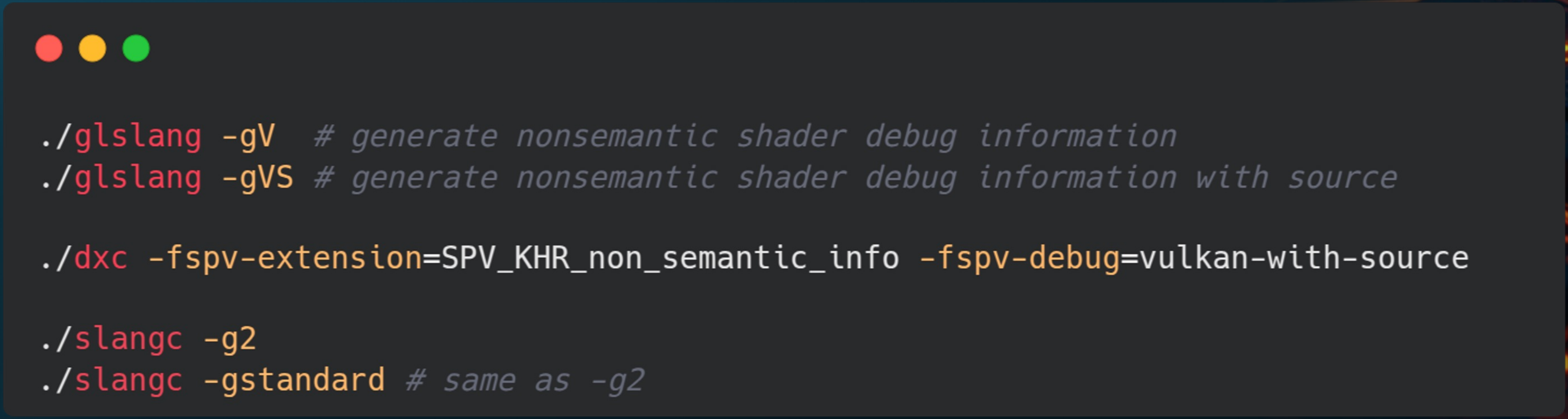
```
SPIR-V Instruction Index = 41
Shader validation error occurred in file a.comp at line 6, column 14

    x = data.indices[16];
             ^
```

How to get this to work?

- **App** – make sure device supports VK_KHR_shader_non_semantic_info
  - Promoted in Vulkan 1.3
- **Shading Language** – Produce the debug info
  - They likely do already!
- **Tools** – Consume the debug info

「Toolbox」 NonSemantic Shader DebugInfo

- Simple way to turn add to your SPIR-V
- Create a “debug” build of your shaders
- Same idea of relwithdebinfo



```
./glslang -gV  # generate nonsemantic shader debug information
./glslang -gVS # generate nonsemantic shader debug information with source

./dxc -fspv-extension=SPV_KHR_non_semantic_info -fspv-debug=vulkan-with-source

./slangc -g2
./slangc -gstandard # same as -g2
```

2025 is the year of better shader debugging

- Thank Baldur!
- Many tools still need to improve usage of this
- Created SPIR-V Guide article to help https://github.com/KhronosGroup/SPIRV-Guide/blob/main/chapters/shader_debug_info.md
- File bug reports on your tools if they use incorrectly!

「Toolbox」 Debug Printf

- Lets you use printf() inside your shader
- Great to find values inside your shader
- Great to know if hit Ray Tracing stages
- Can produce a **LOT** of data
  - Would suggest wrapping with things such as if (gl_VertexIndex == 0)

LUNARG

「Toolbox」 Debug Printf

- Simple idea
    - Store values in a buffer, read afterwards and use sprintf()
- Standardized with SPV_KHR_non_semantic_info
    - Will work the same in Validation Layers, RenderDoc, etc
    - More info found in SPIR-V Guide

「Toolbox」 Debug Printf

- Simple idea
  - Store values in a buffer, read afterwards and use sprintf()
- Standardized with SPV_KHR_non_semantic_info
  - Will work the same in Validation Layers, RenderDoc, etc
  - More info found in SPIR-V Guide

- Need 2 things

1. Add to your shader
2. Have a tool consume it (and display the results)

LUNARG

「Toolbox」 Debug Printf Example

GLSL

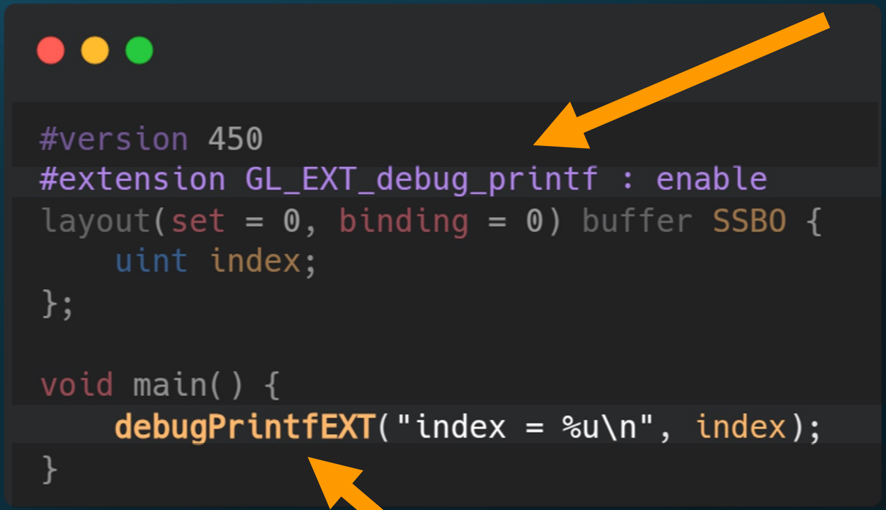
```
#version 450
#extension GL_EXT_debug_printf : enable
layout(set = 0, binding = 0) buffer SSBO {
    uint index;
};

void main() {
    debugPrintfEXT("index = %u\n", index);
}
```

HLSL / Slang

```
RWStructuredBuffer<uint> SSBO;

[shader("compute")]
[numthreads(1, 1, 1)]
void main() {
    uint index = SSBO[0];
    printf("index = %u\n", index);
}
```

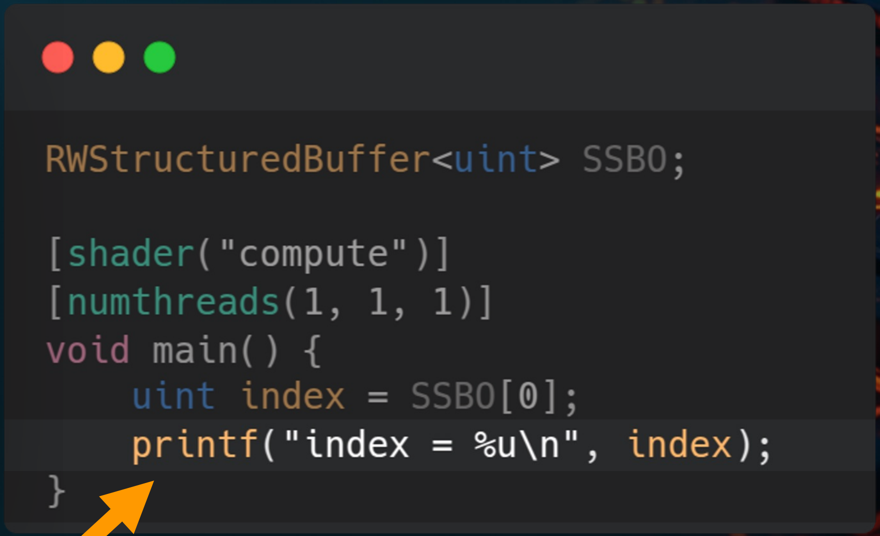
「Toolbox」 Debug Printf Example

GLSL



```glsl
#version 450
#extension GL_EXT_debug_printf : enable
layout(set = 0, binding = 0) buffer SSBO {
    uint index;
};

void main() {
    debugPrintfEXT("index = %u\n", index);
}
```

HLSL / Slang

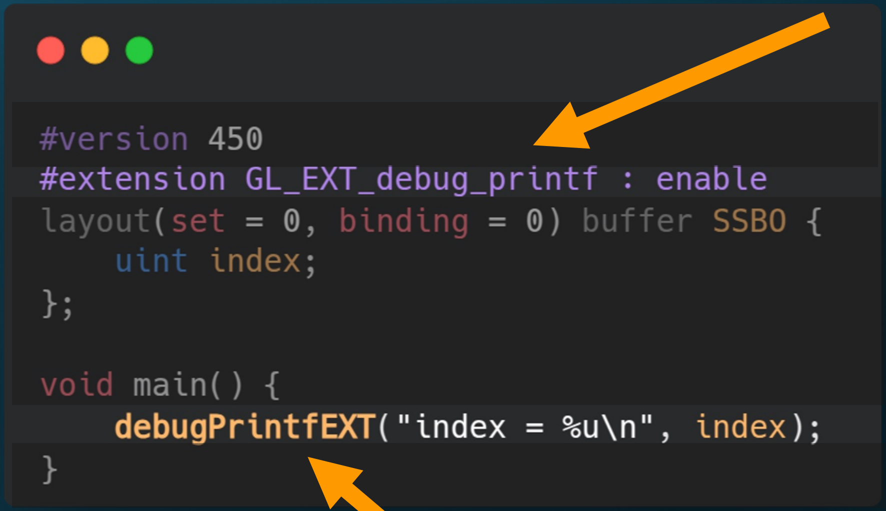


```hlsl
RWStructuredBuffer<uint> SSBO;

[shader("compute")]
[numthreads(1, 1, 1)]
void main() {
    uint index = SSBO[0];
    printf("index = %u\n", index);
}
```

LUNARG

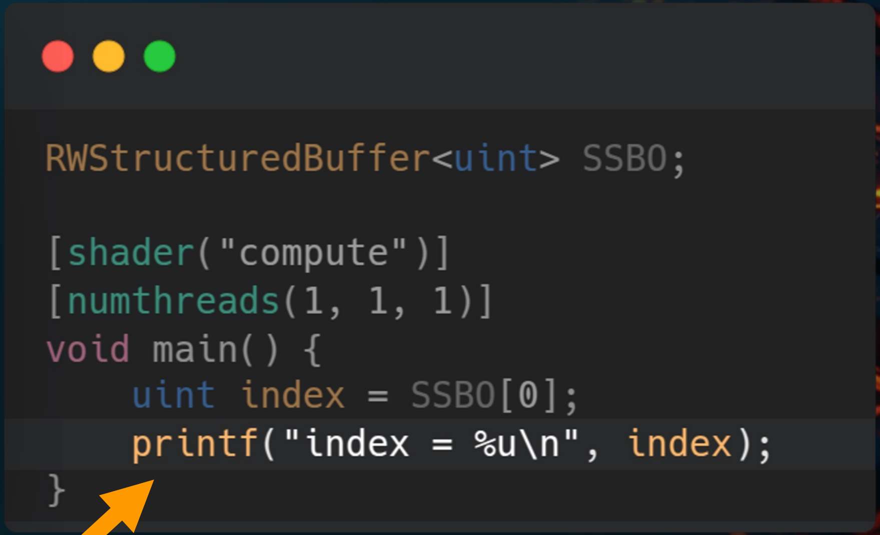
「Toolbox」 Debug Printf Example

GLSL



```
#version 450
#extension GL_EXT_debug_printf : enable
layout(set = 0, binding = 0) buffer SSBO {
    uint index;
};

void main() {
    debugPrintfEXT("index = %u\n", index);
}
```

HLSL / Slang



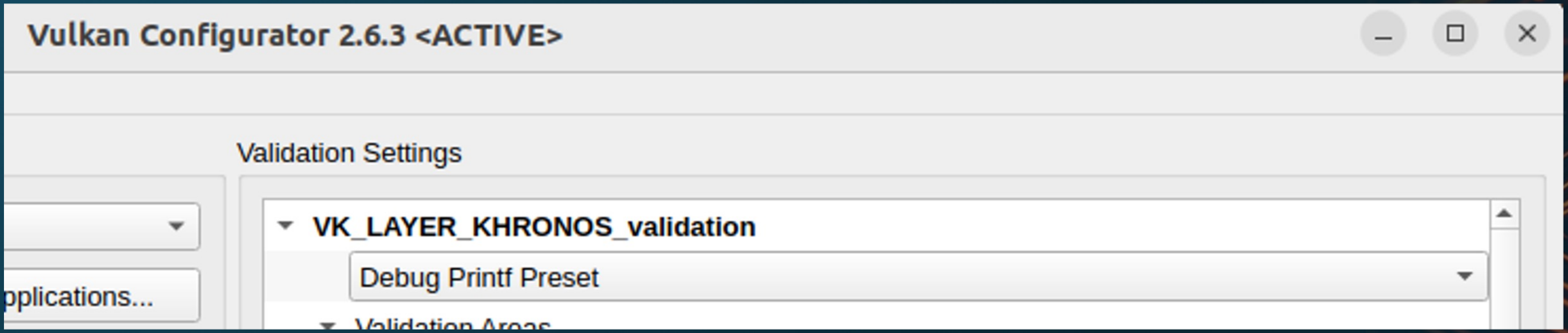
```
RWStructuredBuffer<uint> SSBO;

[shader("compute")]
[numthreads(1, 1, 1)]
void main() {
    uint index = SSBO[0];
    printf("index = %u\n", index);
}
```

It is really this simple to add to your shader!

LUNARG

「Toolbox」 Debug Printf Example

- Super fast to get going with in VkConfig



Validation Settings

VK_LAYER_KHRONOS_validation

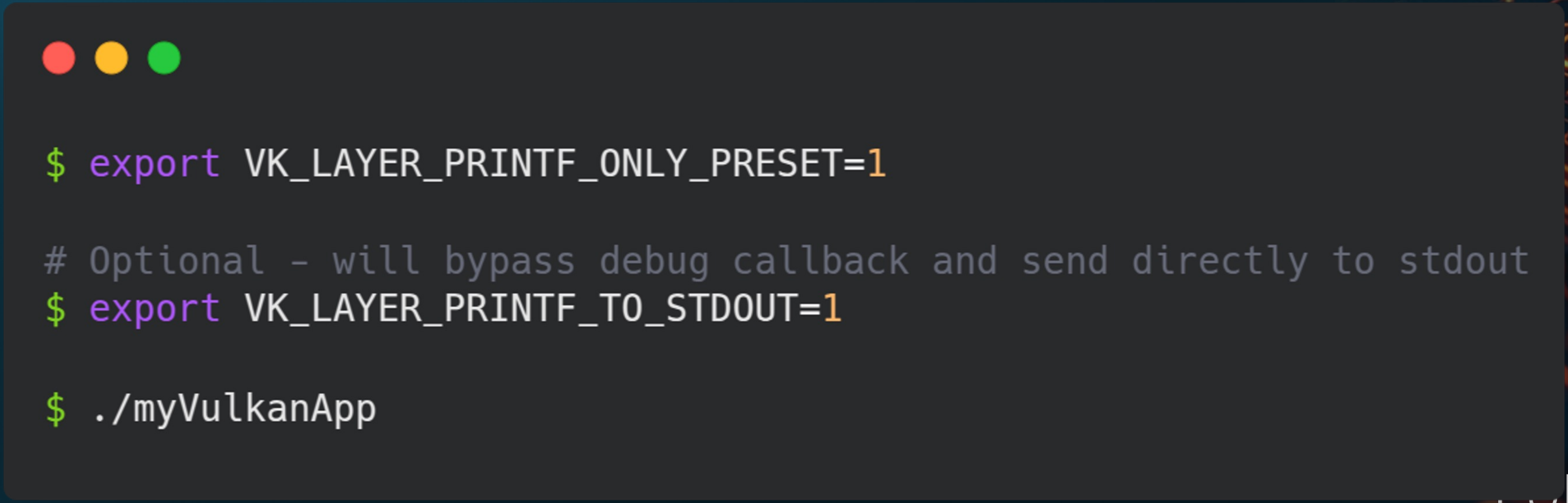
Debug Printf Preset

Validation Areas

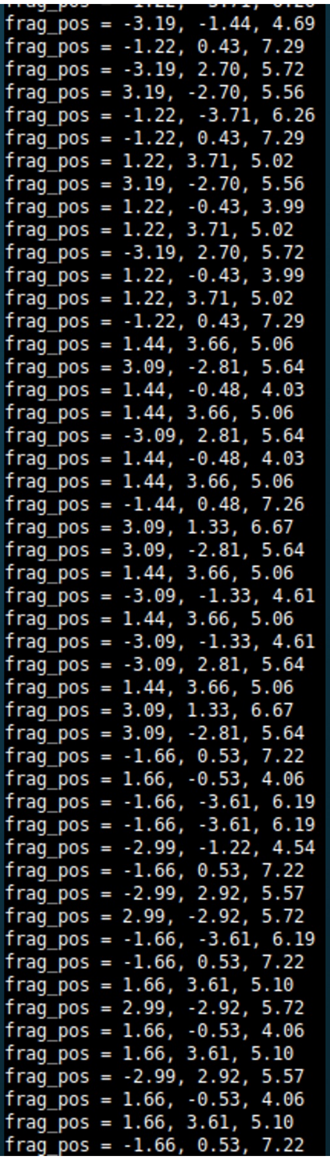
printf() results in same spot as normal validation layer error messages

「Toolbox」 Debug Printf Example

- New way to quickly turn on Debug Printf
  - Need 1.4.304 SDK (or later)
  - Will disable the rest of Validation Layers for you
  - Still recommend using VkConfig if you can instead



```
$ export VK_LAYER_PRINTF_ONLY_PRESET=1

# Optional - will bypass debug callback and send directly to stdout
$ export VK_LAYER_PRINTF_TO_STDOUT=1

$ ./myVulkanApp
```



```
frag_pos = -3.19, -1.44, 4.69
frag_pos = -1.22, 0.43, 7.29
frag_pos = -3.19, 2.70, 5.72
frag_pos = 3.19, -2.70, 5.56
frag_pos = -1.22, -3.71, 6.26
frag_pos = -1.22, 0.43, 7.29
frag_pos = 1.22, 3.71, 5.02
frag_pos = 3.19, -2.70, 5.56
frag_pos = 1.22, -0.43, 3.99
frag_pos = 1.22, 3.71, 5.02
frag_pos = -3.19, 2.70, 5.72
frag_pos = 1.22, -0.43, 3.99
frag_pos = 1.22, 3.71, 5.02
frag_pos = -1.22, 0.43, 7.29
frag_pos = 1.44, 3.66, 5.06
frag_pos = 3.09, -2.81, 5.64
frag_pos = 1.44, -0.48, 4.03
frag_pos = 1.44, 3.66, 5.06
frag_pos = -3.09, 2.81, 5.64
frag_pos = 1.44, -0.48, 4.03
frag_pos = 1.44, 3.66, 5.06
frag_pos = -1.44, 0.48, 7.26
frag_pos = 3.09, 1.33, 6.67
frag_pos = 3.09, -2.81, 5.64
frag_pos = 1.44, 3.66, 5.06
frag_pos = -3.09, -1.33, 4.61
frag_pos = 1.44, 3.66, 5.06
frag_pos = -3.09, -1.33, 4.61
frag_pos = -3.09, 2.81, 5.64
frag_pos = 1.44, 3.66, 5.06
frag_pos = 3.09, 1.33, 6.67
frag_pos = 3.09, -2.81, 5.64
frag_pos = -1.66, 0.53, 7.22
frag_pos = 1.66, -0.53, 4.06
frag_pos = -1.66, -3.61, 6.19
frag_pos = -1.66, -3.61, 6.19
frag_pos = -2.99, -1.22, 4.54
frag_pos = -1.66, 0.53, 7.22
frag_pos = -2.99, 2.92, 5.57
frag_pos = 2.99, -2.92, 5.72
frag_pos = -1.66, -3.61, 6.19
frag_pos = -1.66, 0.53, 7.22
frag_pos = 1.66, 3.61, 5.10
frag_pos = 2.99, -2.92, 5.72
frag_pos = 1.66, -0.53, 4.06
frag_pos = 1.66, 3.61, 5.10
frag_pos = -2.99, 2.92, 5.57
frag_pos = 1.66, -0.53, 4.06
frag_pos = 1.66, 3.61, 5.10
frag_pos = -1.66, 0.53, 7.22
```

Links examples of Debug Printf

- Compiler Explorer (Godbolt) links to play with online
  - GLSL – https://godbolt.org/z/4fafn75Wq
  - HLSL – https://godbolt.org/z/d84qs4rca
  - Slang – https://godbolt.org/z/xz8v9hnK1
- Shameless plug – did you know SPIR-V and GPU Shading languages are now on Compiler Explorer!

「Toolbox」 Debug Printf - final note

- As of the 1.4.304 SDK can now be simultaneously used with GPU-AV

Help me!

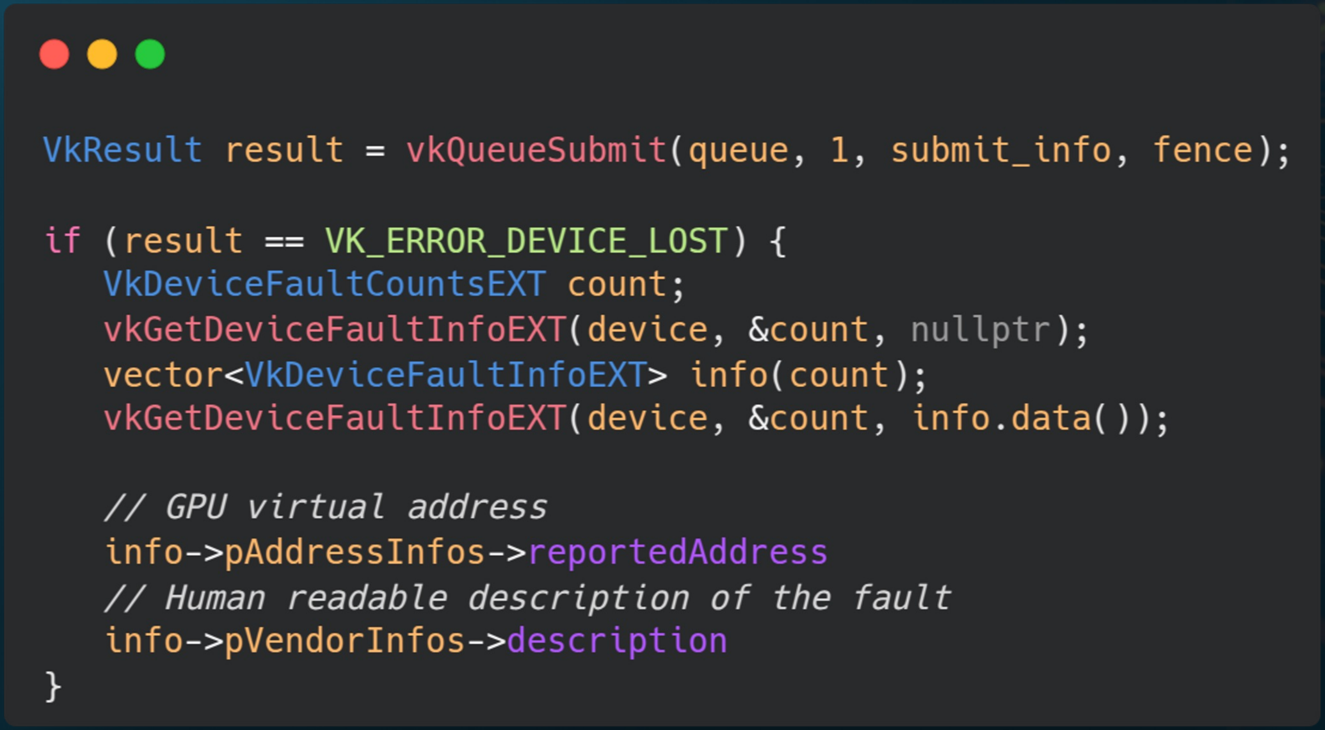
I am getting

VK_ERROR_DEVICE_LOST

and don't know what to do!

「Toolbox」 VK_EXT_device_fault

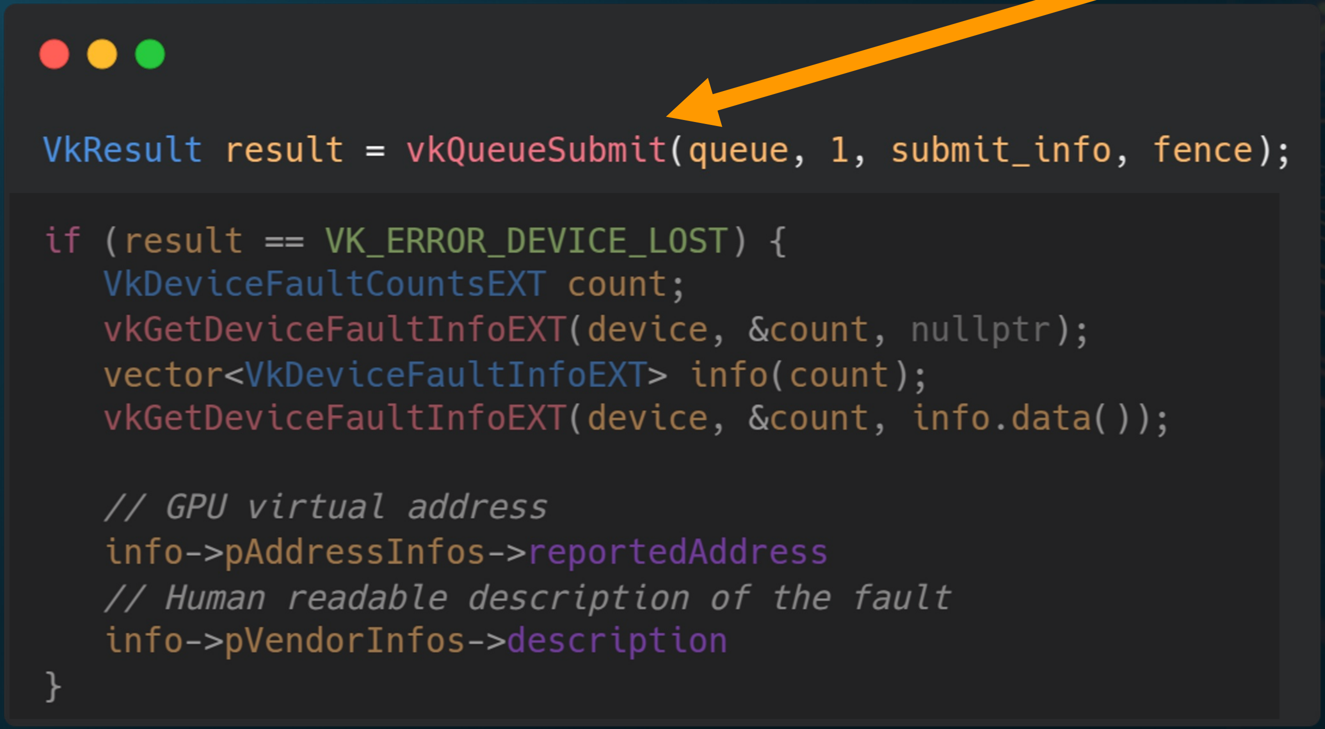
- Can provide great information on VK_ERROR_DEVICE_LOST
  - Unfortunately still not supported everywhere



```
VkResult result = vkQueueSubmit(queue, 1, submit_info, fence);

if (result == VK_ERROR_DEVICE_LOST) {
    VkDeviceFaultCountsEXT count;
    vkGetDeviceFaultInfoEXT(device, &count, nullptr);
    vector<VkDeviceFaultInfoEXT> info(count);
    vkGetDeviceFaultInfoEXT(device, &count, info.data());

    // GPU virtual address
    info->pAddressInfos->reportedAddress
    // Human readable description of the fault
    info->pVendorInfos->description
}
```

「Toolbox」 VK_EXT_device_fault

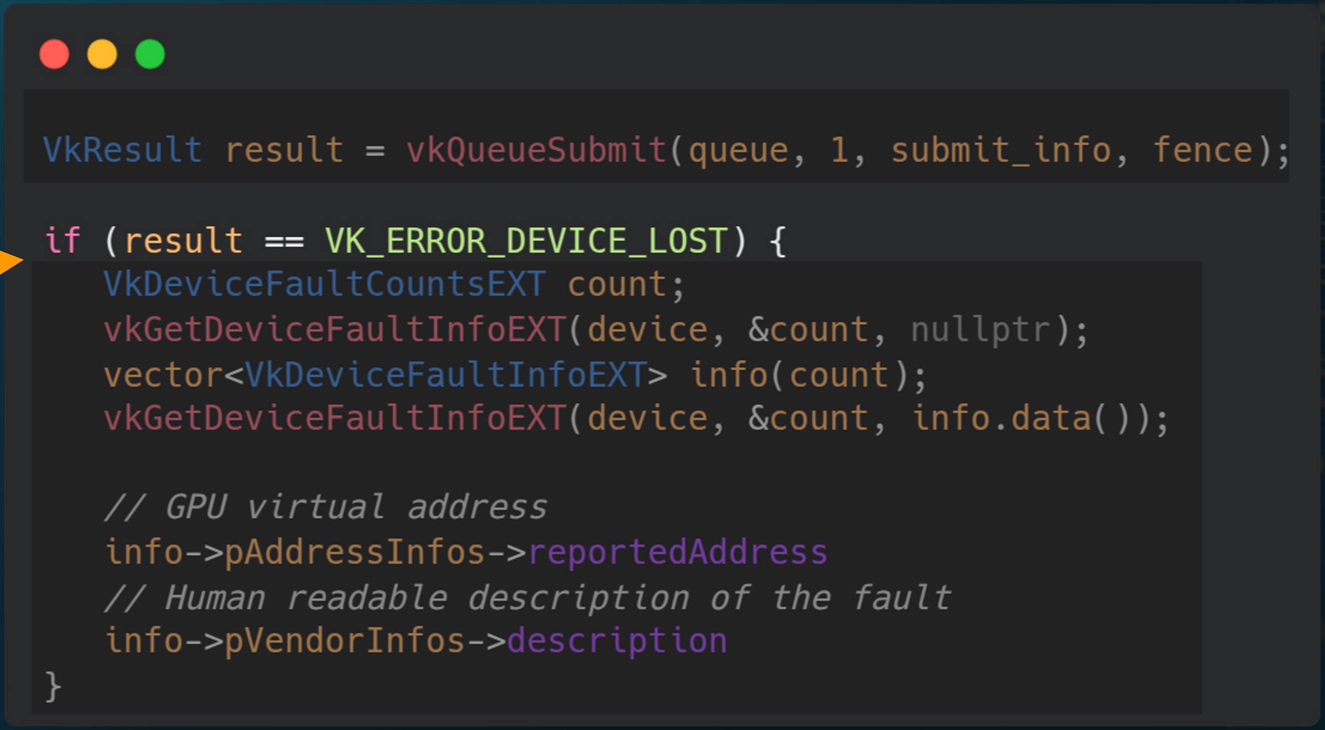
- Can provide great information on VK_ERROR_DEVICE_LOST
  - Unfortunately still not supported everywhere



```
VkResult result = vkQueueSubmit(queue, 1, submit_info, fence);

if (result == VK_ERROR_DEVICE_LOST) {
   VkDeviceFaultCountsEXT count;
   vkGetDeviceFaultInfoEXT(device, &count, nullptr);
   vector<VkDeviceFaultInfoEXT> info(count);
   vkGetDeviceFaultInfoEXT(device, &count, info.data());

   // GPU virtual address
   info->pAddressInfos->reportedAddress
   // Human readable description of the fault
   info->pVendorInfos->description
}
```

「Toolbox」 VK_EXT_device_fault

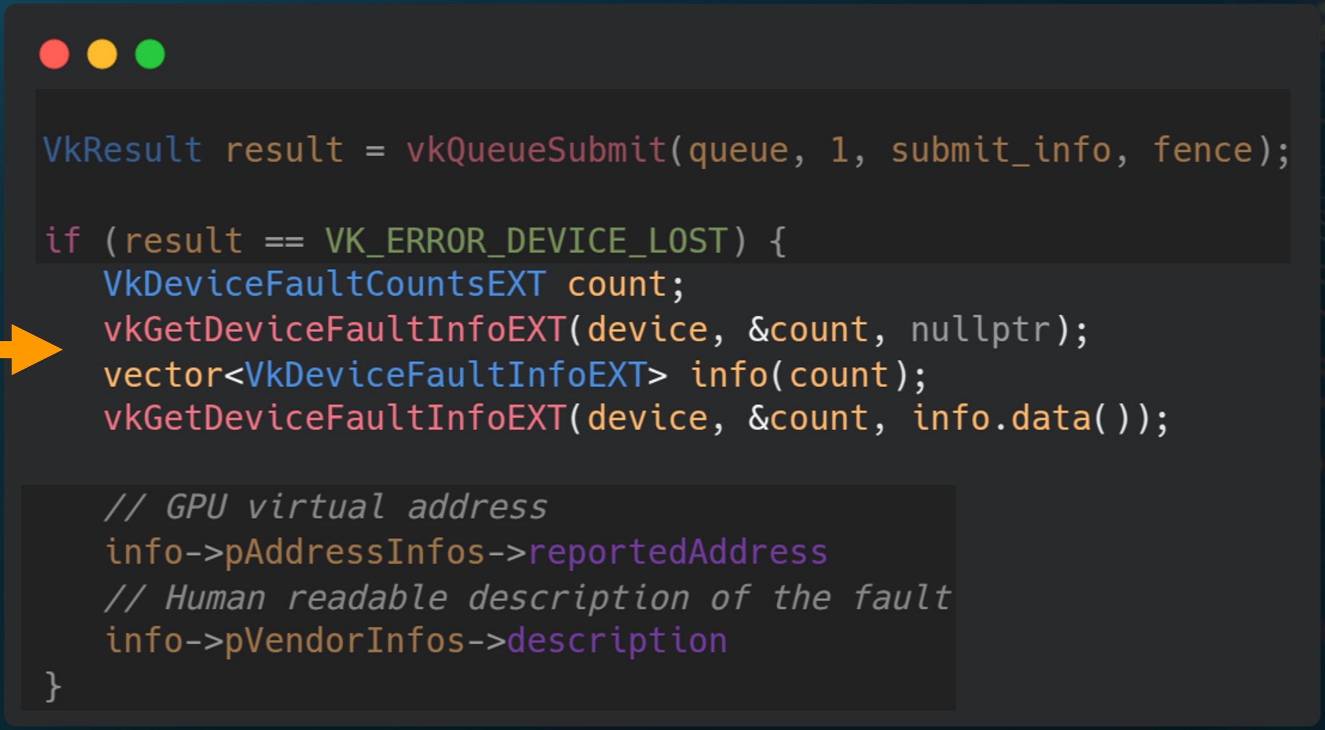
- Can provide great information on VK_ERROR_DEVICE_LOST
  - Unfortunately still not supported everywhere



```
VkResult result = vkQueueSubmit(queue, 1, submit_info, fence);

if (result == VK_ERROR_DEVICE_LOST) {
    VkDeviceFaultCountsEXT count;
    vkGetDeviceFaultInfoEXT(device, &count, nullptr);
    vector<VkDeviceFaultInfoEXT> info(count);
    vkGetDeviceFaultInfoEXT(device, &count, info.data());

    // GPU virtual address
    info->pAddressInfos->reportedAddress
    // Human readable description of the fault
    info->pVendorInfos->description
}
```

「Toolbox」 VK_EXT_device_fault

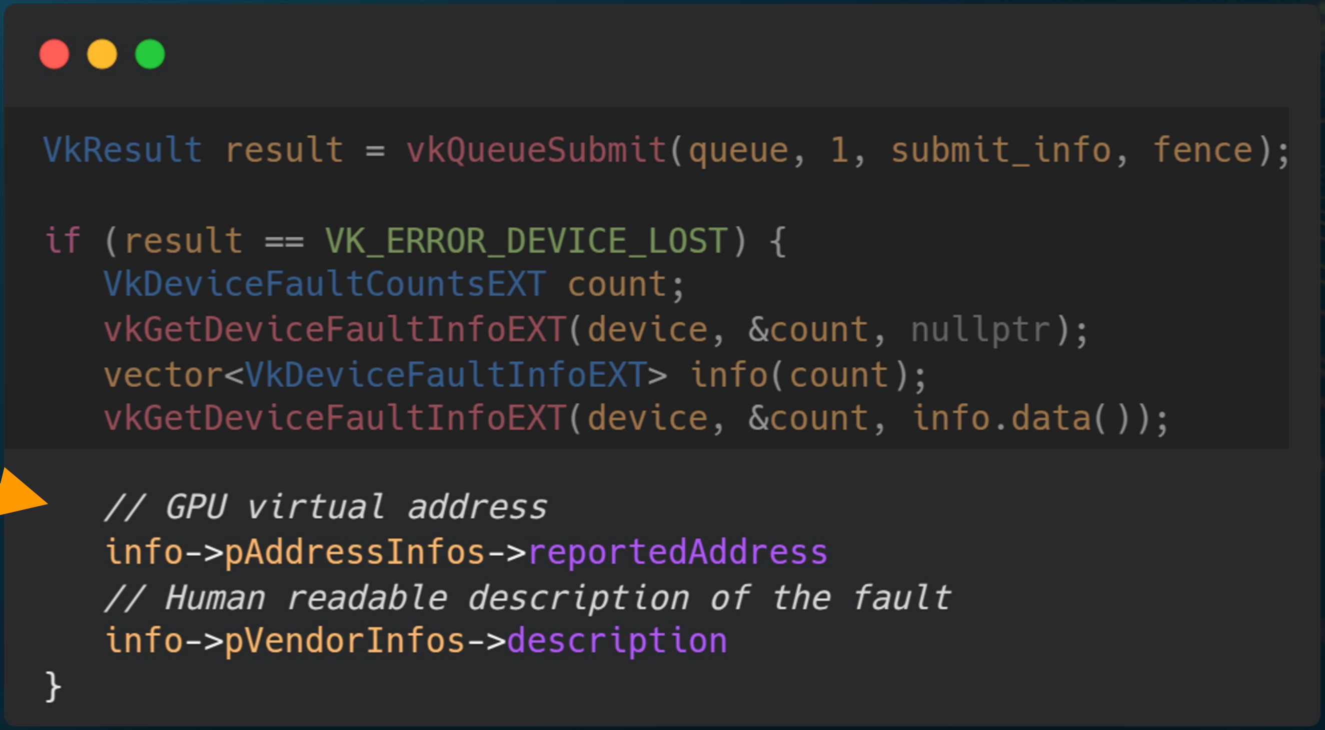
- Can provide great information on VK_ERROR_DEVICE_LOST
  - Unfortunately still not supported everywhere



```
VkResult result = vkQueueSubmit(queue, 1, submit_info, fence);

if (result == VK_ERROR_DEVICE_LOST) {
    VkDeviceFaultCountsEXT count;
    vkGetDeviceFaultInfoEXT(device, &count, nullptr);
    vector<VkDeviceFaultInfoEXT> info(count);
    vkGetDeviceFaultInfoEXT(device, &count, info.data());

    // GPU virtual address
    info->pAddressInfos->reportedAddress
    // Human readable description of the fault
    info->pVendorInfos->description
}
```

「Toolbox」 VK_EXT_device_fault

- Can provide great information on VK_ERROR_DEVICE_LOST
  - Unfortunately still not supported everywhere



```
VkResult result = vkQueueSubmit(queue, 1, submit_info, fence);

if (result == VK_ERROR_DEVICE_LOST) {
   VkDeviceFaultCountsEXT count;
   vkGetDeviceFaultInfoEXT(device, &count, nullptr);
   vector<VkDeviceFaultInfoEXT> info(count);
   vkGetDeviceFaultInfoEXT(device, &count, info.data());

   // GPU virtual address
   info->pAddressInfos->reportedAddress
   // Human readable description of the fault
   info->pVendorInfos->description
}
```

LUNARG

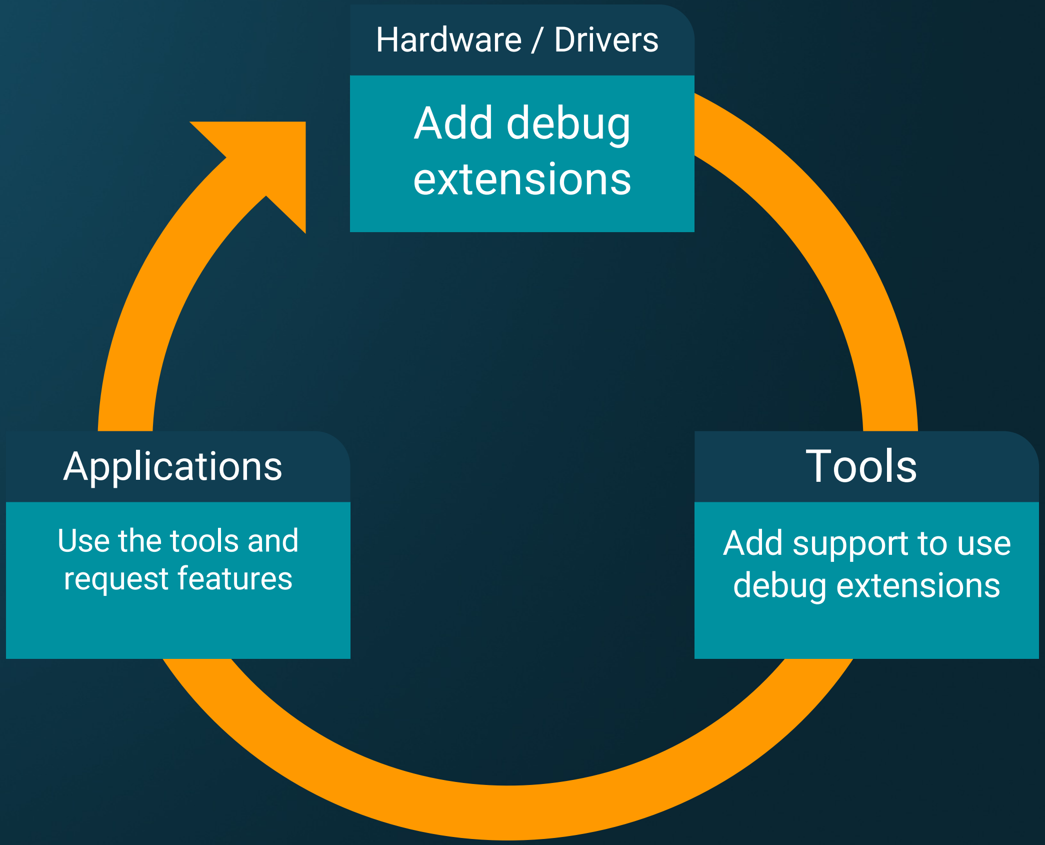
「Toolbox」 VK_EXT_device_fault

- Even simpler options – just use Crash Diagnostic Layer
  - See Jeremy’s talk after me

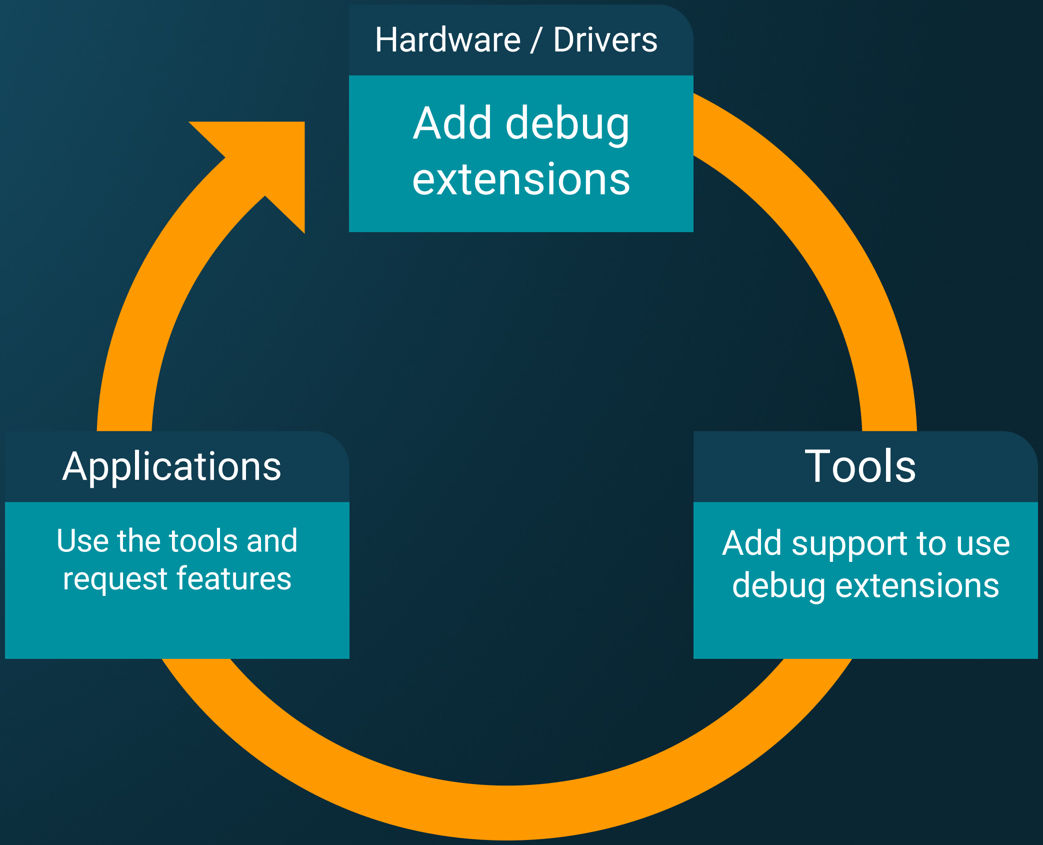
The toolbox is large

- Many other tools I didn't have time to mention
  - RenderDoc
  - Crash Diagnostic Layer
  - GPU Reshape
  - Vendor specific tools
    - Nsight
    - RGP
    - Etc
  - Platform specific tools
    - Android GPU Inspector (AGI)

Takes a village to raise good debugging

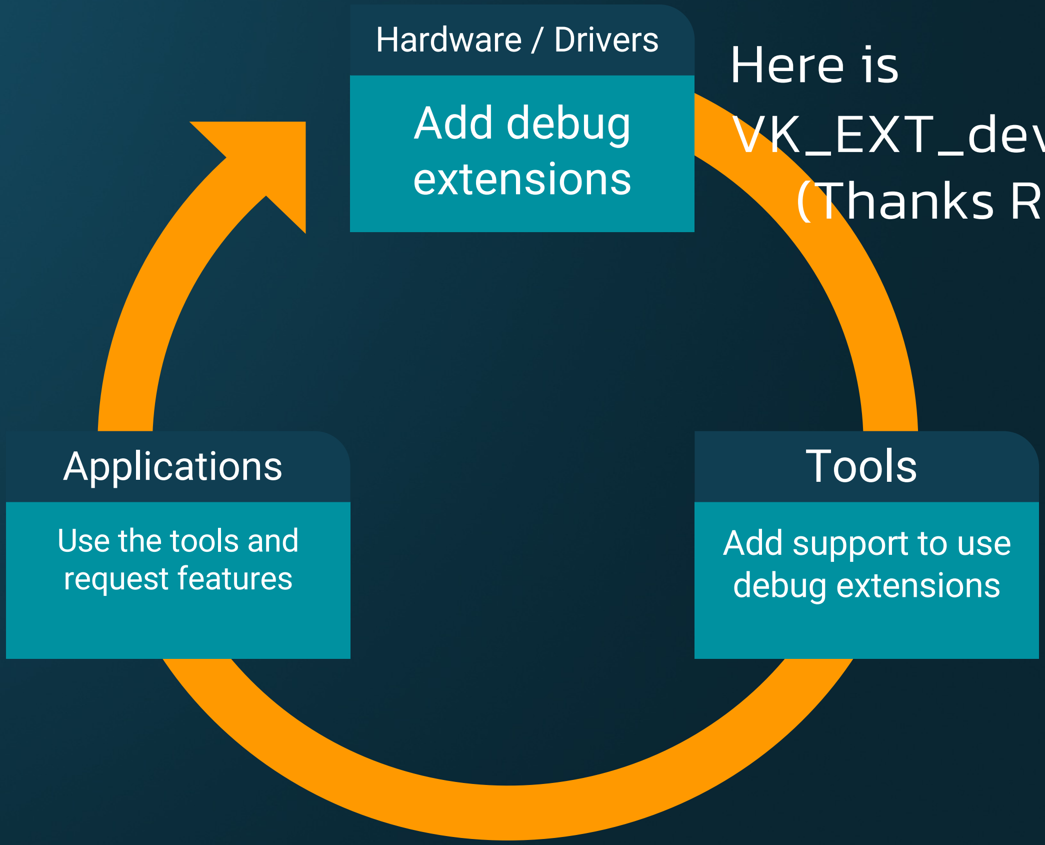

Hardware / Drivers
Add debug extensions
Tools
Add support to use debug extensions
Applications
Use the tools and request features

Takes a village to raise good debugging


Hardware / Drivers
Add debug extensions
Applications
Use the tools and request features
Tools
Add support to use debug extensions

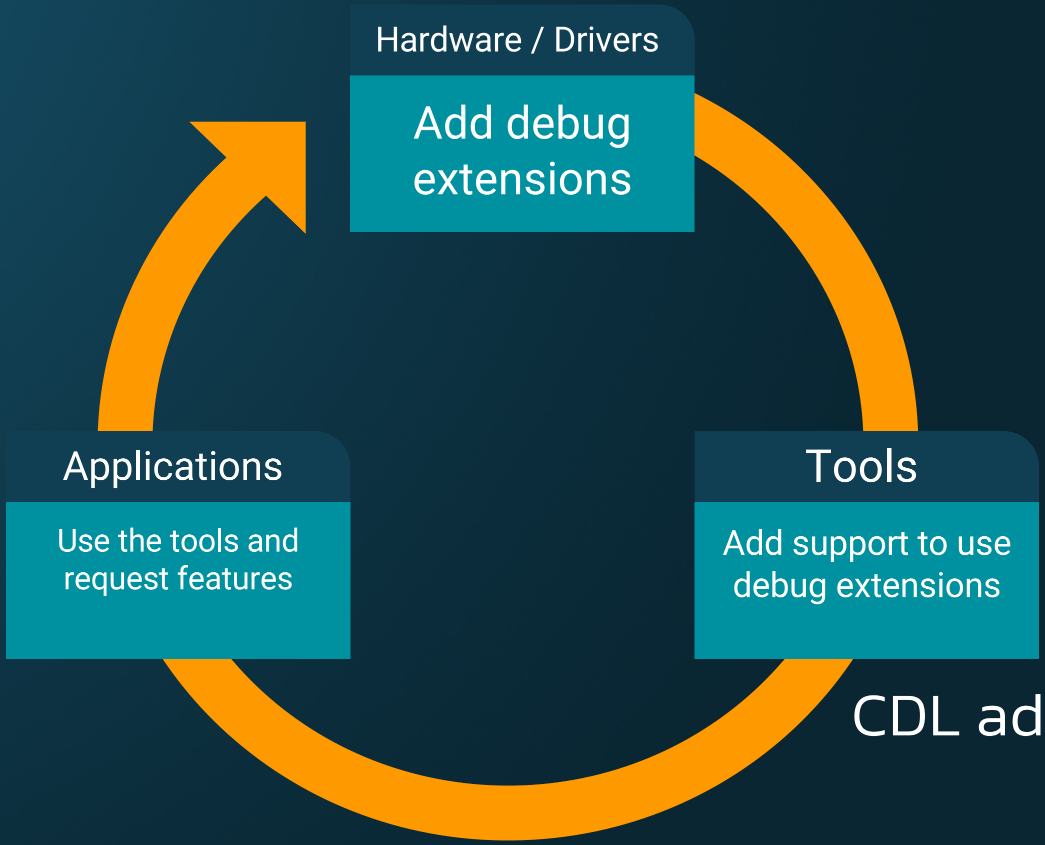
"We want better message on device lost!"

Takes a village to raise good debugging


Hardware / Drivers
Add debug extensions
Tools
Add support to use debug extensions
Applications
Use the tools and request features

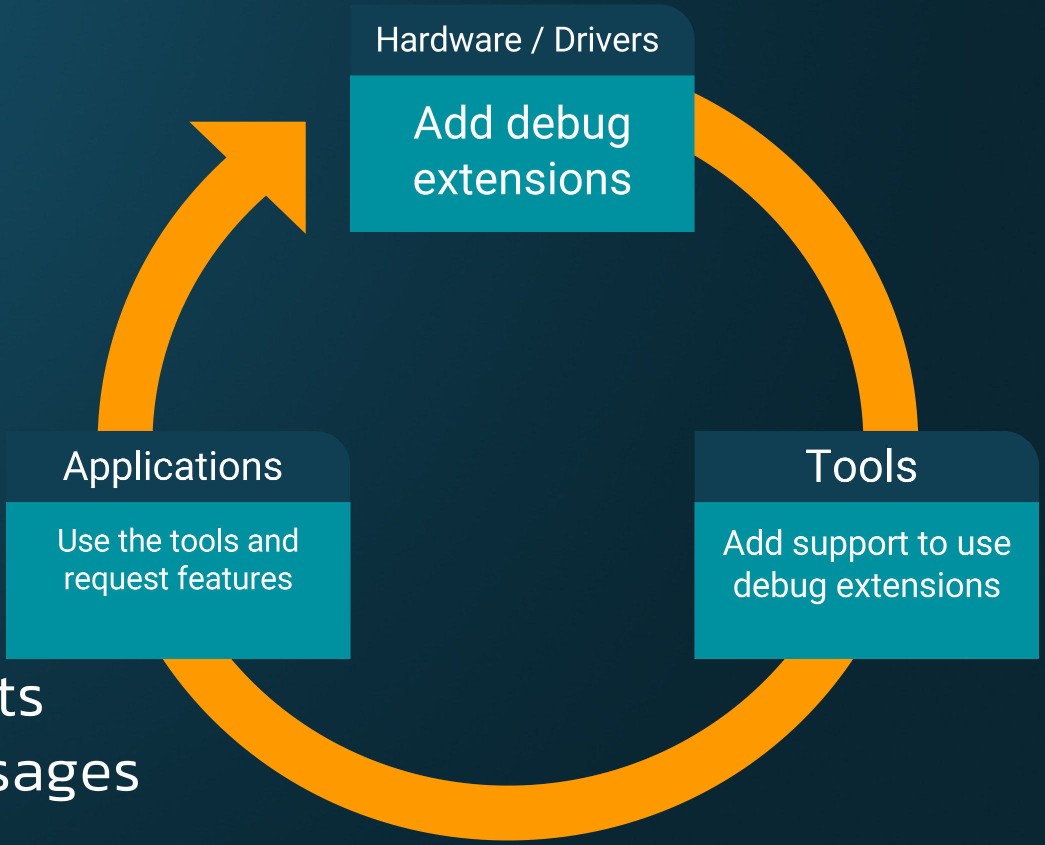
Here is VK_EXT_device_fault (Thanks Ralph!)

Takes a village to raise good debugging


Hardware / Drivers
Add debug extensions
Tools
Add support to use debug extensions
Applications
Use the tools and request features

CDL adds support

Takes a village to raise good debugging


Hardware / Drivers
Add debug extensions
Tools
Add support to use debug extensions
Applications
Use the tools and request features

Uses CDL and gets better error messages

Final Reminder

- Tell me your feedback what tools you want!
  - After the talk
  - Online
  - Over coffee if you are in the Raleigh, NC area!
- Community feedback helps drives a lot of decisions

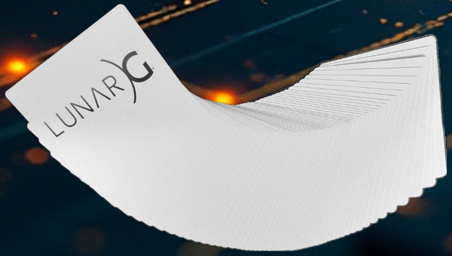
Thank you!

Actions

Download this Presentation

https://khr.io/1cr

Talk to us and get Swag!



Visit the
LunarG Sponsor Table

Take the Annual Developers Survey

https://khr.io/1cq

Your Feedback Matters!

Survey Results
- ➔ Are shared with the Khronos Vulkan Working Group
- ➔ Are used to drive development priorities throughout 2025

Survey Closes
Wednesday, Feb. 19, 2025
(GMT—7)