DEVELOPER DAY

Descriptor Indexing

Hai Nguyen, Google

KHRONOS™ GROUP

UBISOFT

MONTRÉAL
APRIL 2018

Agenda

- Overview
- Descriptors Refresher
- Descriptor Indexing
- More On Unbounded
- Non-Uniform Indexing

Overview

- **Descriptor indexing adds two awesome features to Vulkan**
  - Extremely large descriptor sets (unbounded in DX parlance)
    - Orders of magnitude larger in most cases
  - Dynamic non-uniform resource indexing
    - HLSL shaders can use **NonUniformResourceIndex**
- **Descriptor indexing is an EXT extension**
  - EXT means that IHVs are not required to implement this extension
  - EXT extensions may or may not become a part of the core
- **Please provide feedback**
  - Your feedback helps shape the extensions should it become KHR or core

Descriptors Refresher

- **Limits & Organization**
  - Descriptors in Vulkan have both per stage and per set limits
  - Descriptor sets are collections of bindings described by set layouts
- **Allocation**
  - Sets are allocated from descriptor pools
    - Pools have both a set count and descriptor count limit
- **Updating**
  - Updates must happen outside of command record and execution
    - E.g. No updating after **vkCmdBindDescriptorSets**
- **Shader Bindings**
  - Each descriptor binding maps to a single resource binding in a shader
    - Including descriptors that are arrays (this differs from DX)

Descriptors Refresher: Updating

```
// OK! cmd is in initial state
vkUpdateDescriptorSets([set0, set1, set2]);

// Record cmd
vkBeginCommandBuffer(cmd);
vkCmdBindDescriptorSets(cmd, ...,[set0, set1, set2], ...);
vkEndCommandBuffer(cmd);

// NO! BAD! UNDEFINED BEHAVIOR!
vkUpdateDescriptorSets([set0, set1, set2]);

// Submit cmd
vkQueueSubmit(...)

// NO! BAD! UNDEFINED BEHAVIOR!
vkUpdateDescriptorSets([set0, set1, set2]);

// ...synchronization to tell us work is done for cmd...

// OK! cmd is in executable state
vkUpdateDescriptorSets([set0, set1, set2]);
```

Descriptors Refresher: Arrayness

```
// HLSL
Texture2D Textures[10] : register(t1);
SamplerState Sampler : register(s0);
float4 main() : SV_Target {
  float2 tc = (float2)0;
  return Textures[9].Sample(Sampler, tc);
}

// FXC
// Name             Type  Format         Dim      ID      HLSL Bind  Count
// --------- ---------- ------- ----------- ------- -------------- ------
// Sampler      sampler      NA          NA      S0             s0      1
// Textures     texture  float4          2d      T0             t1     10
//
dcl_resource_texture2d (float,float,float,float) T0[1:10], space=0

; SPIR-V
       OpName %4 "Textures"
       OpDecorate %4 DescriptorSet 0
       OpDecorate %4 Binding 1
  %3 = OpTypeImage %8 2D 0 0 0 1 Unknown
 %10 = OpConstant %9 10
 %11 = OpTypeArray %3 %10
 %12 = OpTypePointer UniformConstant %11
  %4 = OpVariable %12 UniformConstant
```

Descriptors Refresher: Arrayness

- Indexing behaves the same in HLSL on Vulkan and DirectX
- HLSL on DirectX flattens out the arrays start from the base register
- HLSL on Vulkan keeps arrays in the same binding
- Don’t panic when inspecting IR or bytecode 

Descriptor Indexing: Basics

- **VK_EXT_descriptor_indexing**
  - Applications can create much larger sets
  - Shaders can index descriptors using dynamic non-uniform
  - Limits queried using `VkPhysicalDeviceDescriptorIndexingPropertiesEXT`
  - Features queried using `VkPhysicalDeviceDescriptorIndexingFeaturesEXT`
  - Features toggled using `VkPhysicalDeviceDescriptorIndexingFeaturesEXT`
- **Requirements**
  - SDK 1.1.73.0
  - Driver that supports `VK_EXT_descriptor_indexing`
- **Shader Compiler Support**
  - HLSL - DXC
  - GLSL - glslang/shaderc

Descriptor Indexing: Limits & Organization

- **Much Larger Sets**
  - Driver still imposes limits
  - Minimum limit per set is 500K
- **How Much Larger?**
  - Example: IHV 1 (per set):
    - SampledImages: 90K -> 1M
    - Samplers: 4K -> 1M
    - StorageBuffers: 90K -> 1M
  - Example: IHV 2 (per set):
    - 4B -> 4B (was already really high)

Descriptor Indexing: Limits & Organization

- **Arrayed Descriptors Can Be Unbounded**
  - Unbounded is called ***variable size*** in Vulkan parlance
  - Use `VK_DESCRIPTOR_BINDING_VARIABLE_DESCRIPTOR_COUNT_BIT_EXT` to indicate a descriptor binding has a variable size
    - Variable count is restricted to the last binding in the set

Descriptor Indexing: Updating

- **Changes to descriptor updating with VK_EXT_descriptor_indexing**
  - Descriptors can be updated after binding in command buffers
    - Command buffer execution will use most recent updates
  - Descriptors not in use by pending command buffers can be updated
    - Enables writing new descriptors for frame N+1 while frame N is executing
- **Some assembly required**
  - Update after binding takes some setup
    - Descriptor bindings, sets, and pools must all be created with the required update after bind flags
  - Update unused while pending only requires a descriptor binding flag

Descriptors Indexing: Updating

```
// OK! cmd is in initial state
vkUpdateDescriptorSets([set0, set1, set2]);

// Record cmd
vkBeginCommandBuffer(cmd);
vkCmdBindDescriptorSets(cmd, ...,[set0, set1, set2], ...);
vkEndCommandBuffer(cmd);

// ~~NO! BAD!~~ OK! Latest updates will be used
vkUpdateDescriptorSets([set0, set1, set2]);

// Submit cmd
vkQueueSubmit(...)

// ~~NO! BAD!~~ OK! For descriptors not in use
vkUpdateDescriptorSets([set0, set1, set2]);

// ...synchronization to tell us work is done for cmd...

// OK! cmd is in executable state
vkUpdateDescriptorSets([set0, set1, set2]);
```

Descriptor Indexing: Shader Bindings

- It's OK to have invalid descriptors in a binding if they're *not* in use
- Example: `Texture2D MyTextures[65536] : register(t0);`
  - Same shader is used by 8,192 objects
    - Each object references 8 textures
    - Index using [ObjectID*8, ObjectID*8 + 8)
  - Objects are deferred until they come into scene
    - PSOs are created and descriptors are updated when objects are "visible"
  - Descriptors will be partially populated until all objects are seen...
    - ...and no one will die...because it's all OK
- Some assembly required (just tiny bit)
  - Binding must be created with VK_DESCRIPTOR_BINDING_PARTIALLY_BOUND_BIT_EXT

More On Unbounded

- **Quick recap from earlier**
  - *Variable size* in Vulkan parlance
  - Requires `VK_DESCRIPTOR_BINDING_VARIABLE_DESCRIPTOR_COUNT_BIT_EXT`
    - `VkDescriptorSetLayoutBinding::descriptorCount` is an upper bound on the size of the binding
    - Variable count is restricted to last binding in the set
- **Slightly different than how it works in DirectX**
  - DirectX namespaces descriptor types (b#, s#, t#, u#)
    - Any of these can be unbounded on the last binding
  - Vulkan only permits the last binding in the set to be unbounded
    - No restriction on the descriptor type

Non-Uniform Indexing

- **Quick recap from earlier**
  - Array indexing works the same in HLSL for both VK and DX
    - Lower level mechanics notwithstanding 
  - Slight differences in unbounded behavior between VK and DX
    - Hopefully larger descriptor sets helps with this
- **Non- Uniform Indexing must be enabled at device creation time**
  - `shader*ArrayNonUniformIndexing` in `VkPhysicalDeviceDescriptorIndexingFeaturesEXT`
- **NonUniformResourceIndex in HLSL**
  - Works the same way it does in DirectX
  - Currently only supported in DXC

Non-Uniform Indexing

```
RWTexture2D<float>  OutBuffer : register(u0);
Texture2D           MyTextures[] : register(t1);

[numthreads(1, 1, 1)]
void main(uint3 tid : SV_DispatchThreadid) {
  uint index = (tid.y * 256 + tid.y) % 1024;
  Texture2D tex = MyTextures[NonUniformResourceIndex(index)];
  float4 val = tex[tid.xy];
  OutBuffer[tid.xy] = val;
}
```

Summary

- **VK_EXT_descriptor_indexing**
  - Much larger descriptor sets
  - Flexible descriptor updates
  - Non-uniform indexing (NonUniformResourceIndex)