Feeding Your Shaders

ARM

Jesse Barker
Principal Software Engineer

Moving to Vulkan: How to make your 3D graphics more explicit

May 26, 2016

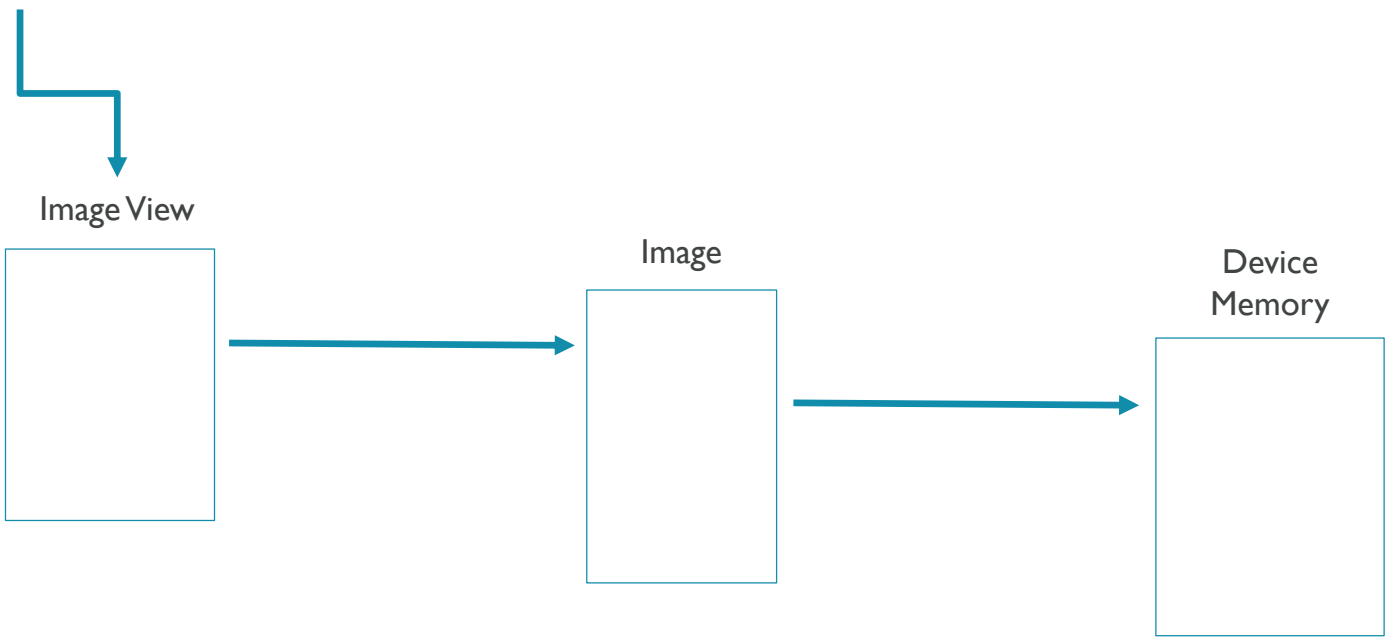
What is a Vulkan Resource?

- Shader Input/Output
- Referenced via *Descriptors*
- Some are specialized in the hardware
  - Vertex Input Attributes
  - Render Targets

- Buffers
- Images
- Samplers
- Input Attachments

What are Vulkan Descriptors?

Handle	Type
myImageView	SAMPLED_IMAGE



What are Descriptor Sets?

```
// uniform blocks:
layout(set = 0, binding = 0) uniform Type0 { ... } ubo0;

// textures:
layout(set = 0, binding = 1) uniform sampler2D tex0;

// SSBO:
layout(set = 0, binding = 2) buffer Type2 { ... } ssbo0;

void main()
    // ...
}
```

binding	type	stages
0	Uniform Buffer	Graphics
1	Image/Sampler	Graphics
2	Storage Buffer	Graphics

What is a Descriptor Pool?

- Parent object of a Descriptor Set
- Allows Descriptor Set management to be threaded
- Manages memory for hardware descriptors

```
typedef struct VkDescriptorPoolSize {
    VkDescriptorType    type;
    uint32_t            descriptorCount;
} VkDescriptorPoolSize;

typedef struct VkDescriptorPoolCreateInfo {
    VkStructureType                 sType;
    const void*                     pNext;
    VkDescriptorPoolCreateFlags     flags;
    uint32_t                        maxSets;
    uint32_t                        poolSizeCount;
    const VkDescriptorPoolSize*     pPoolSizes;
} VkDescriptorPoolCreateInfo;
```

Allocating Descriptor Sets

- Define desired layouts of descriptors
- Ask the Descriptor Pool to allocate a Descriptor Set per layout

What is a Pipeline Layout?

```
// uniform blocks:
layout(set = 0, binding = 0) uniform Type0
{ ... } ubo0;
layout(set = 0, binding = 0) uniform Type1
{ ... } ubo1;

// textures:
layout(set = 0, binding = 1) uniform
sampler2D tex0;
layout(set = 1, binding = 0) uniform
sampler2D tex1;

// SSBO:
layout(set = 1, binding = 1) buffer Type2 {
... } ssbo0;

void main() {
    // ...
}
```

Descriptor Set 0

binding	type	stages
0	Uniform Buffer	Graphics
0	Uniform Buffer	Graphics
1	Image/Sampler	Graphics

Descriptor Set 1

binding	type	stages
0	Image/Sampler	Graphics
1	Storage Buffer	Graphics

How do Descriptors get into Descriptor Sets?

```
VKAPI_ATTR void VKAPI_CALL vkUpdateDescriptorSets(
    VkDevice                    device,
    uint32_t
descriptorWriteCount,
    const VkWriteDescriptorSet* pDescriptorWrites,
    uint32_t                    descriptorCopyCount,
    const VkCopyDescriptorSet* pDescriptorCopies);
```

```
typedef struct VkWriteDescriptorSet {
    VkStructureType                  sType;
    const void*                      pNext;
    VkDescriptorSet                  dstSet;
    uint32_t                         dstBinding;
    uint32_t                         dstArrayElement;
    uint32_t                         descriptorCount;
    VkDescriptorType                 descriptorType;
    const VkDescriptorImageInfo*  pImageInfo;
    const VkDescriptorBufferInfo* pBufferInfo;
    const VkBufferView*              pTexelBufferView;
} VkWriteDescriptorSet;

typedef struct VkCopyDescriptorSet {
    VkStructureType    sType;
    const void*        pNext;
    VkDescriptorSet    srcSet;
    uint32_t           srcBinding;
    uint32_t           srcArrayElement;
    VkDescriptorSet    dstSet;
    uint32_t           dstBinding;
    uint32_t           dstArrayElement;
    uint32_t           descriptorCount;
} VkCopyDescriptorSet;
```

Finally, I'm ready to use my Descriptor Sets

```
VKAPI_ATTR void VKAPI_CALL vkCmdBindDescriptorSets(
    VkCommandBuffer         commandBuffer,
    VkPipelineBindPoint     pipelineBindPoint,
    VkPipelineLayout        layout,
    uint32_t                firstSet,
    uint32_t                descriptorSetCount,
    const VkDescriptorSet*  pDescriptorSets,
    uint32_t                dynamicOffsetCount,
    const uint32_t*         pDynamicOffsets);
```

- Bound sets must match pipeline layout
- Graphics or compute?
- Simple layout is best

What about Vertex Input?

Vertex Input Description

If your shader declares:

```
in vec3 position;
in uvec2 texcoord;
```

Your C code declares:

```
struct Position
{
    float x, y, z;
};

struct Texcoord
{
    uint8_t u, v;
};
```

```
const VkVertexInputBindingDescription binding[] =
{
    {
        0,                                          // binding
        sizeof(float) * 3,                          // stride
        VK_VERTEX_INPUT_RATE_VERTEX                 // inputRate
    },
    {
        1,                                          // binding
        sizeof(uint8_t) * 2,                        // stride
        VK_VERTEX_INPUT_RATE_VERTEX                 // inputRate
    },
};

const VkVertexInputAttributeDescription attributes[] =
{
    {
        0,                                          // location
        binding[0].binding,                         // binding
        VK_FORMAT_R32G32B32_SFLOAT,                 // format
        0                                           // offset
    },
    {
        1,                                          // location
        binding[1].binding,                         // binding
        VK_FORMAT_R8G8_UNORM,                       // format
        0                                           // offset
    }
};
```

Questions?

ARM