Vulkanised 2023

The 5th Vulkan Developer Conference
Munich, Germany / February 7–9

An Introduction to Vulkan

Lukas Lipp

TU Wien

Schedule

PART 1:

Setup	Lecture	Coding Session
10 min	**20** min	**90** min
Starts at 09:00	Starts at 09:10	Starts at 09:30

PART 2:

Lecture	Coffee Break	Coding Session
15 min	**25** min	**80** min
Starts at 11:00	Starts at 11:15	Starts at 11:40

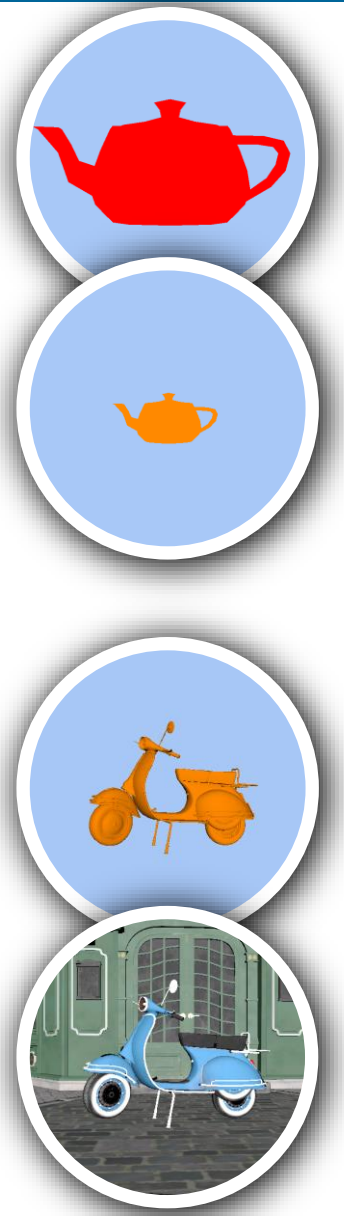
Lunch Break 13:00 – 14:00

PART 3:

Lecture	Coding Session	Coffee Break
15 min	**65** min	**30** min
Starts at 14:00	Starts at 14:15	Starts at 15:20

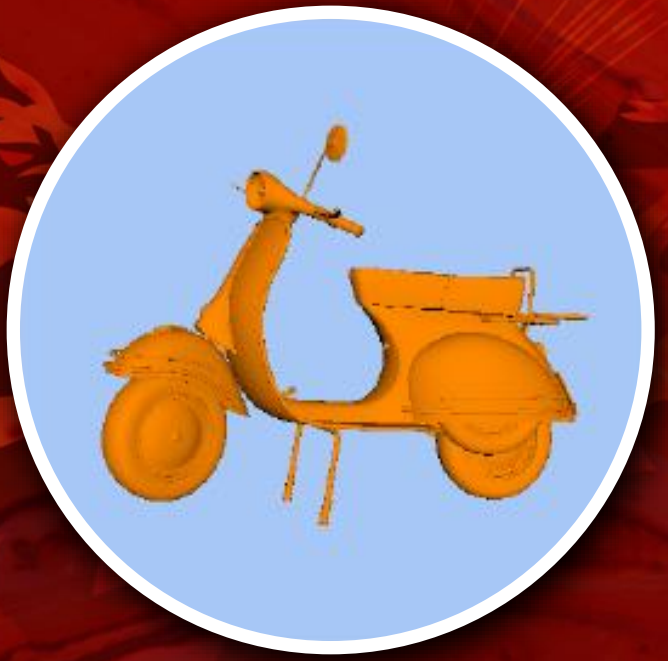
PART 4:

Lecture	Coding Session	Closing
20 min	**70** min	**10** min
Starts at 15:50	Starts at 16:10	Starts at 17:20



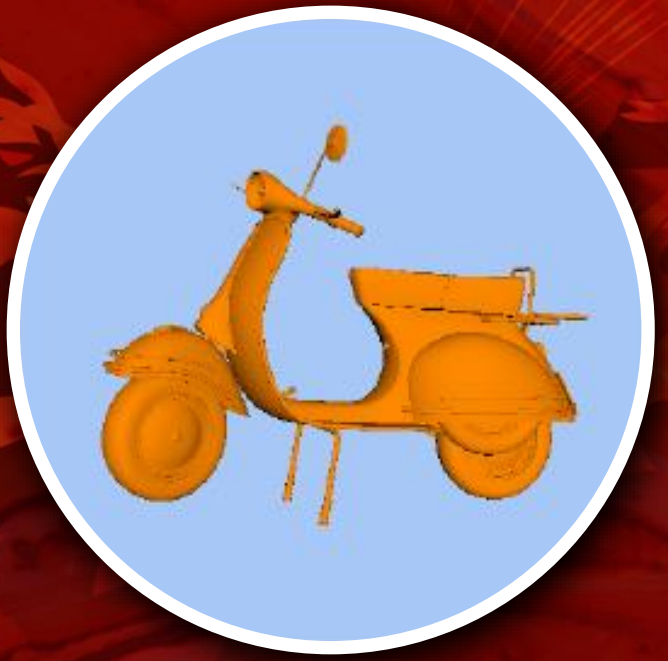
PART 3

- Multiple Vertex Buffers
- Command Buffer Recording
- Multiple Graphics Pipelines
- Depth Test

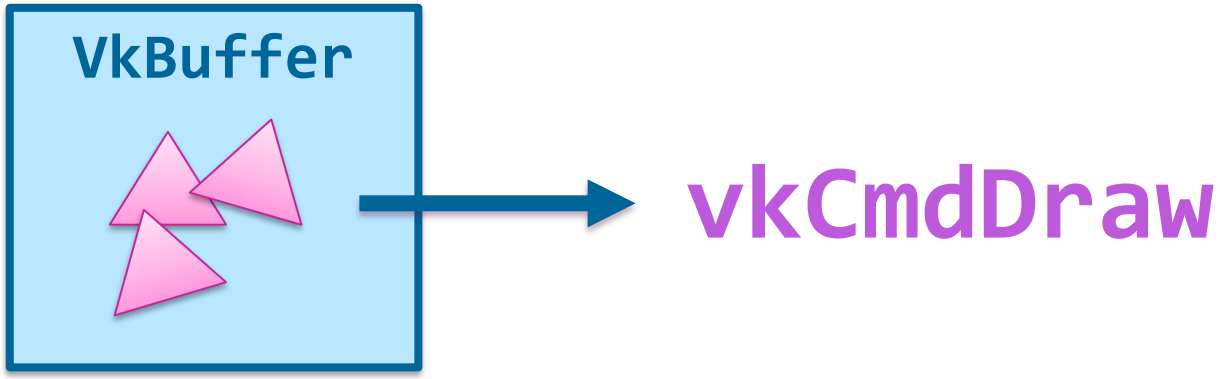


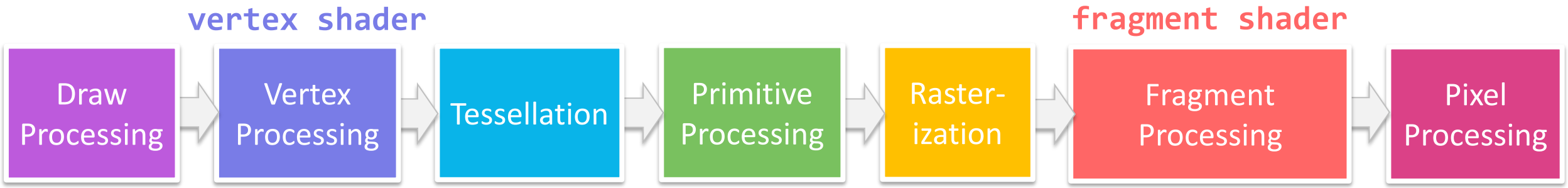
PART 3

- **Multiple Vertex Buffers**
- Command Buffer Recording
- Multiple Graphics Pipelines
- Depth Test

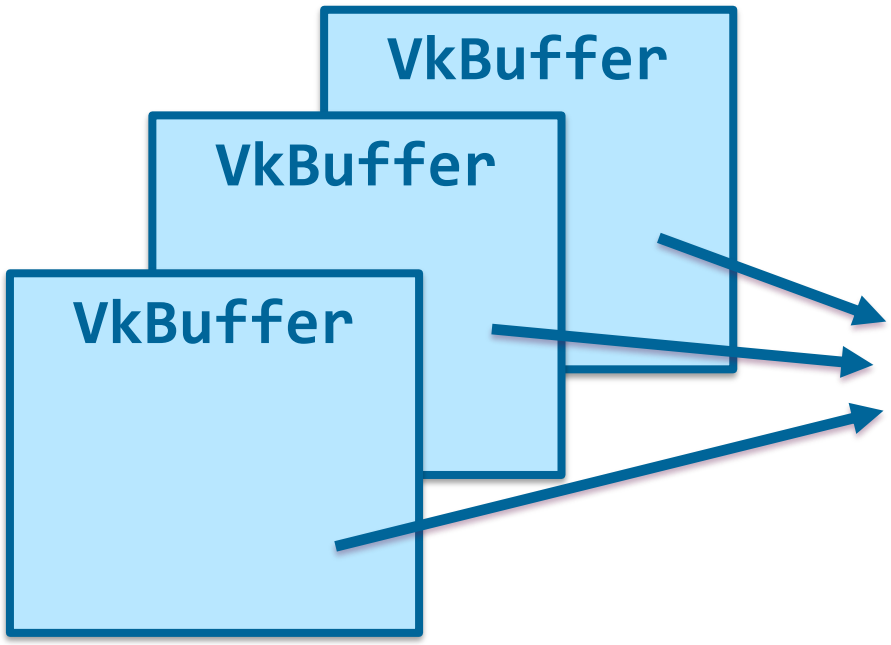


Multiple Vertex Buffers





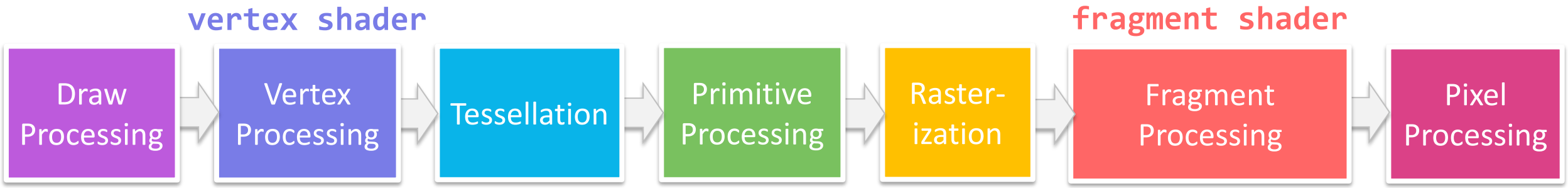
Multiple Vertex Buffers



Multiple Vertex Buffers:

- Positions
- Normals
- Texture Coordinates

`vkCmdDraw`



Multiple Vertex Buffers

```
VkVertexInputBindingDescription binding0 = {};
binding0.binding = 0;
binding0.stride = sizeof(float) * 3;
binding0.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;
```

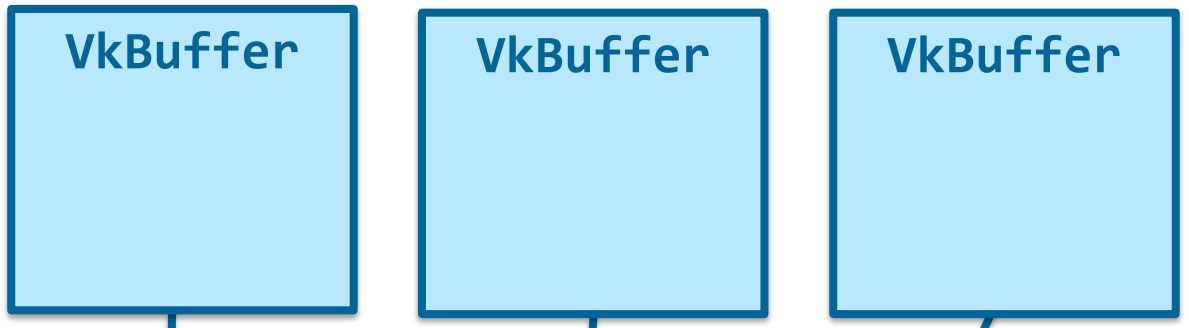
```
VkVertexInputAttributeDescription attribute0 = {};
attribute0.binding = 0;
```


```
VkBuffer vertexBuffers[1] = {
  buffer0
};
VkDeviceSize offsets[1] = {
  0
};
vkCmdBindVertexBuffers(
  commandBuffer,
  0, 1,
  vertexBuffers, offsets);
vkCmdDraw(...);
```

Multiple Vertex Buffers

```
VkVertexInputBindingDescription binding0 = {};
binding0.binding = 0;
binding0.stride = sizeof(float) * 3;
binding0.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;
```

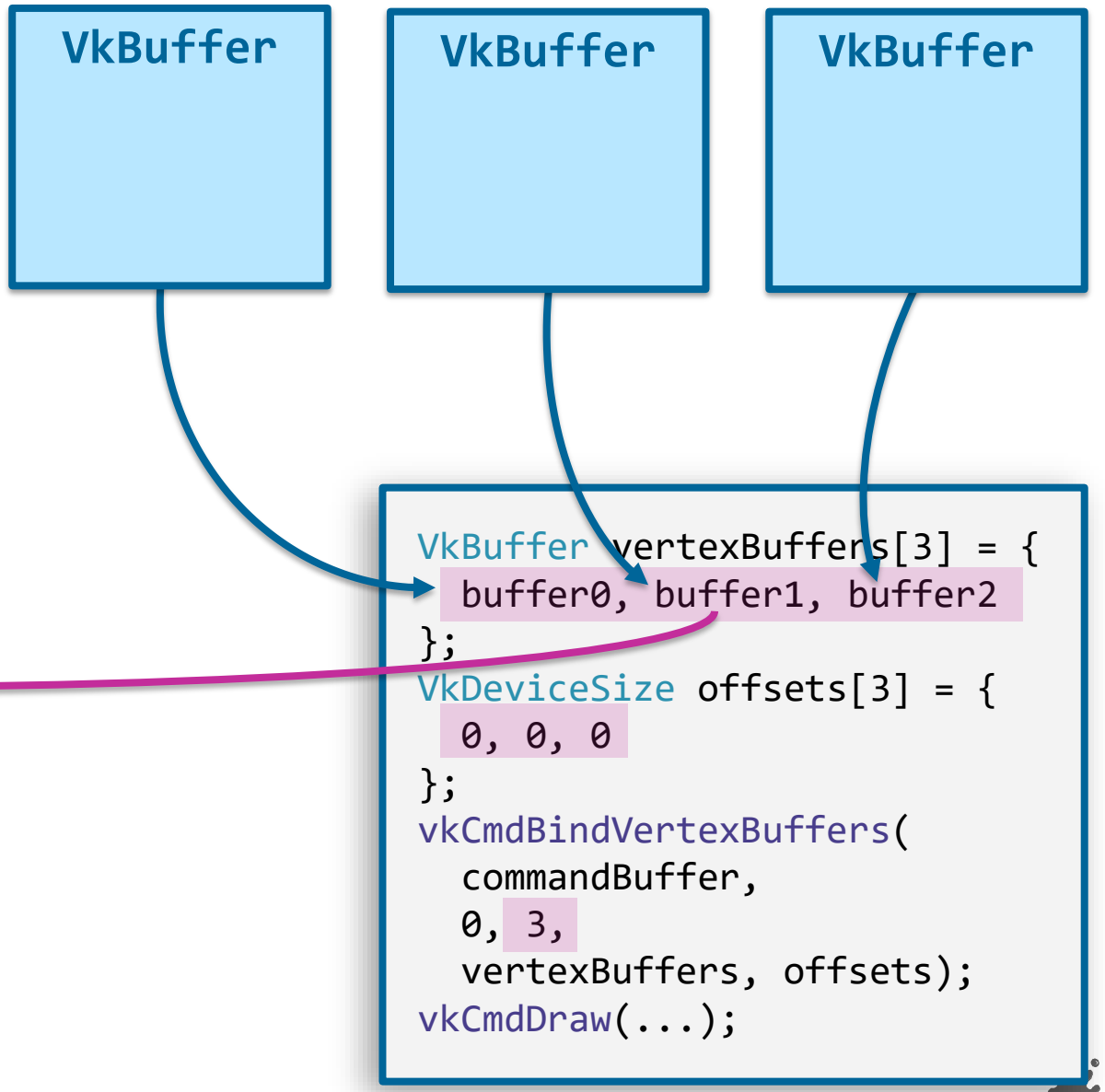
```
VkVertexInputAttributeDescription attribute0 = {};
attribute0.binding = 0;
```




```
VkBuffer vertexBuffers[3] = {
  buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3] = {
  0, 0, 0
};
vkCmdBindVertexBuffers(
  commandBuffer,
  0, 3,
  vertexBuffers, offsets);
vkCmdDraw(...);
```

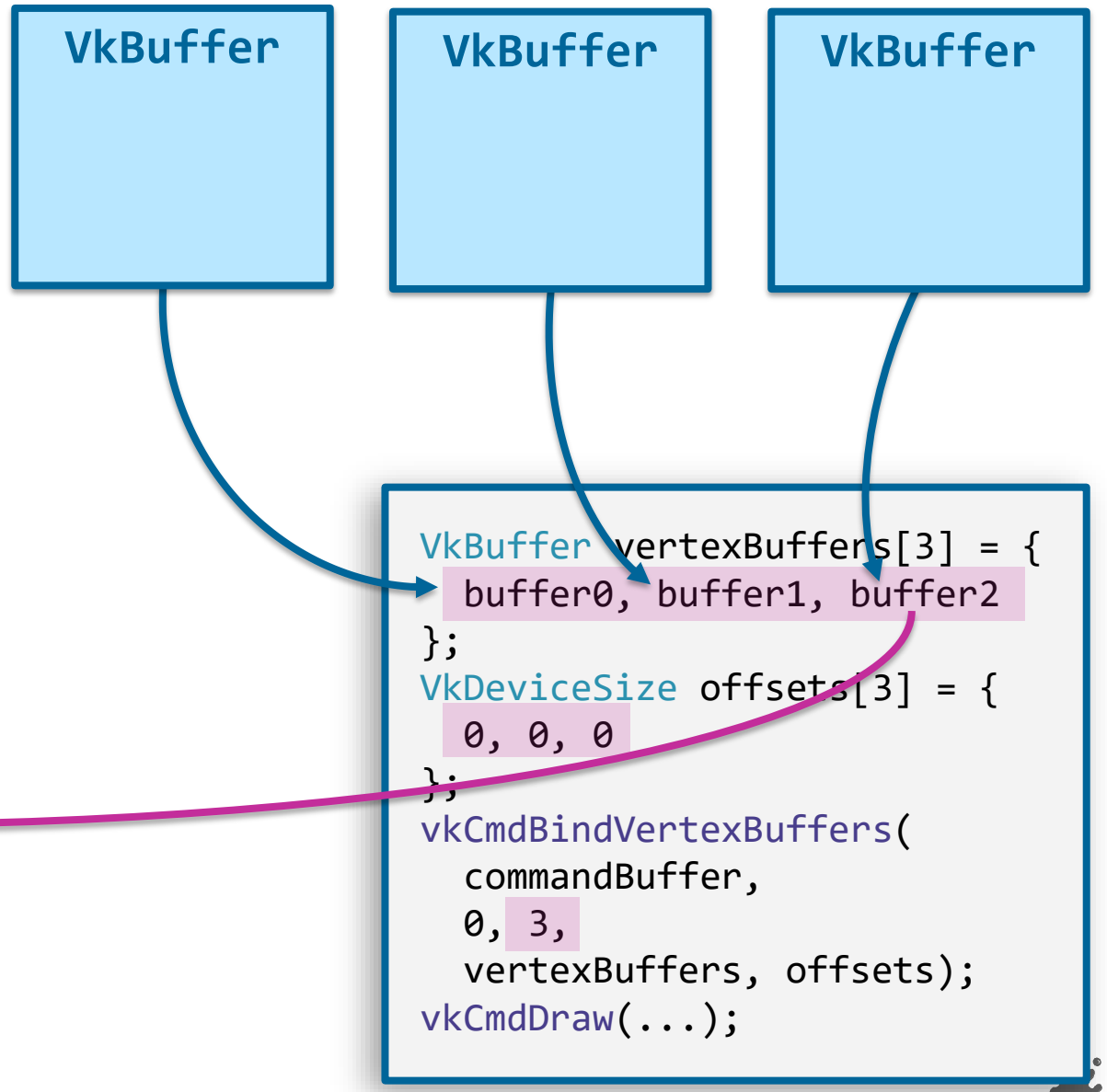
Multiple Vertex Buffers

```
VkVertexInputBindingDescription binding0 = {};
binding0.binding = 0;
binding0.stride = sizeof(float) * 3;
binding0.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;

VkVertexInputBindingDescription binding1 = {};
binding1.binding = 1;
binding1.stride = sizeof(float) * 3;
binding1.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;

VkVertexInputBindingDescription binding2 = {};
binding2.binding = 2;
binding2.stride = sizeof(float) * 2;
binding2.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;

VkVertexInputAttributeDescription attribute0 = {};
attribute0.binding = 0;

VkVertexInputAttributeDescription attribute1 = {};
attribute1.binding = 1;

VkVertexInputAttributeDescription attribute2 = {};
attribute2.binding = 2;
```



Multiple Vertex Buffers

```
VkVertexInputBindingDescription binding0 = {};
binding0.binding = 0;
binding0.stride = sizeof(float) * 3;
binding0.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;

VkVertexInputBindingDescription binding1 = {};
binding1.binding = 1;
binding1.stride = sizeof(float) * 3;
binding1.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;

VkVertexInputBindingDescription binding2 = {};
binding2.binding = 2;
binding2.stride = sizeof(float) * 2;
binding2.inputRate = VK_VERTEX_INPUT_RATE_VERTEX;

VkVertexInputAttributeDescription attribute0 = {};
attribute0.binding = 0;

VkVertexInputAttributeDescription attribute1 = {};
attribute1.binding = 1;

VkVertexInputAttributeDescription attribute2 = {};
attribute2.binding = 2;
```



Multiple Vertex Buffers

```
VkVertexInputAttributeDescription attribute0 = {};
attribute0.location = 0;
attribute0.binding = 0;
attribute0.format = VK_FORMAT_R32G32B32_SFLOAT;
attribute0.offset = 0;

VkVertexInputAttributeDescription attribute1 = {};
attribute1.location = 2;
attribute1.binding = 1;
attribute1.format = VK_FORMAT_R32G32B32_SFLOAT;
attribute1.offset = 0;

VkVertexInputAttributeDescription attribute2 = {};
attribute2.location = 1;
attribute2.binding = 2;
attribute2.format = VK_FORMAT_R32G32_SFLOAT;
attribute2.offset = 0;
```

GLSL vertex shader

```
#version 450

layout (binding = 0) uniform UniformBuffer {
    vec4 color;
    mat4 transformationMatrix;
} uniform_buffer;

layout (location = 0) in vec3 in_position;
layout (location = 1) in vec2 in_tex_coord;
layout (location = 2) in vec3 in_normal;

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

Multiple Vertex Buffers

```
VkVertexInputAttributeDescription attribute0 = {};
attribute0.location = 0;
attribute0.binding = 0;
attribute0.format = VK_FORMAT_R32G32B32_SFLOAT;
attribute0.offset = 0;

VkVertexInputAttributeDescription attribute1 = {};
attribute1.location = 2;
attribute1.binding = 1;
attribute1.format = VK_FORMAT_R32G32B32_SFLOAT;
attribute1.offset = 0;

VkVertexInputAttributeDescription attribute2 = {};
attribute2.location = 1;
attribute2.binding = 2;
attribute2.format = VK_FORMAT_R32G32_SFLOAT;
attribute2.offset = 0;
```

GLSL vertex shader

```
#version 450

layout (binding = 0) uniform UniformBuffer {
    vec4 color;
    mat4 transformationMatrix;
} uniform_buffer;

layout (location = 0) in vec3 in_position;
layout (location = 1) in vec2 in_tex_coord;
layout (location = 2) in vec3 in_normal;

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

Multiple Vertex Buffers

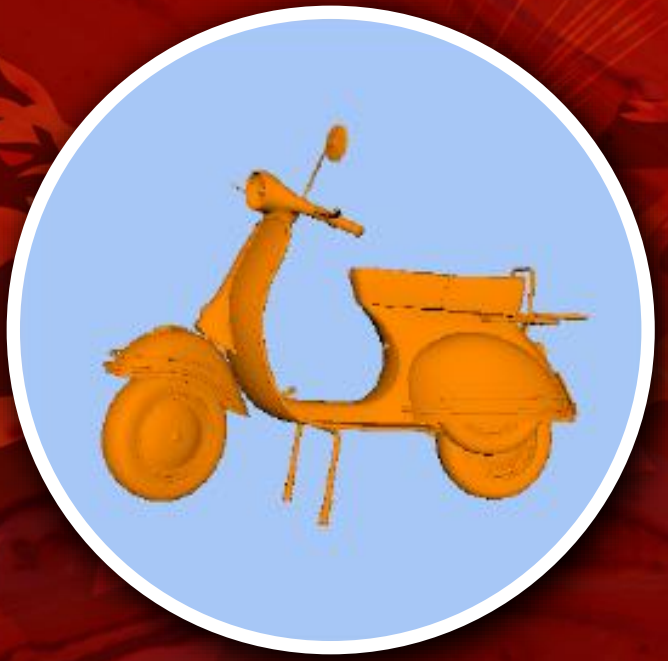
```
VkVertexInputAttributeDescription attribute0 = {};
attribute0.location = 0;
attribute0.binding = 0;
attribute0.format = VK_FORMAT_R32G32B32_SFLOAT;
attribute0.offset = 0;

VkVertexInputAttributeDescription attribute1 = {};
attribute1.location = 2;
attribute1.binding = 1;
attribute1.format = VK_FORMAT_R32G32B32_SFLOAT;
attribute1.offset = 0;

VkVertexInputAttributeDescription attribute2 = {};
attribute2.location = 1;
attribute2.binding = 2;
attribute2.format = VK_FORMAT_R32G32_SFLOAT;
attribute2.offset = 0;
```

GLSL vertex shader

```
#version 450

layout (binding = 0) uniform UniformBuffer {
    vec4 color;
    mat4 transformationMatrix;
} uniform_buffer;

layout (location = 0) in vec3 in_position;
layout (location = 1) in vec2 in_tex_coord;
layout (location = 2) in vec3 in_normal;

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

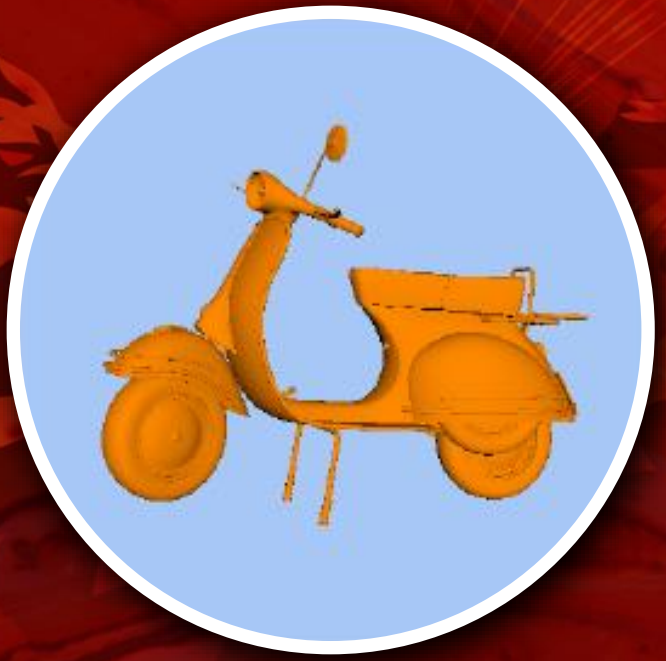
PART 3

- **Multiple Vertex Buffers**
- Command Buffer Recording
- Multiple Graphics Pipelines
- Depth Test

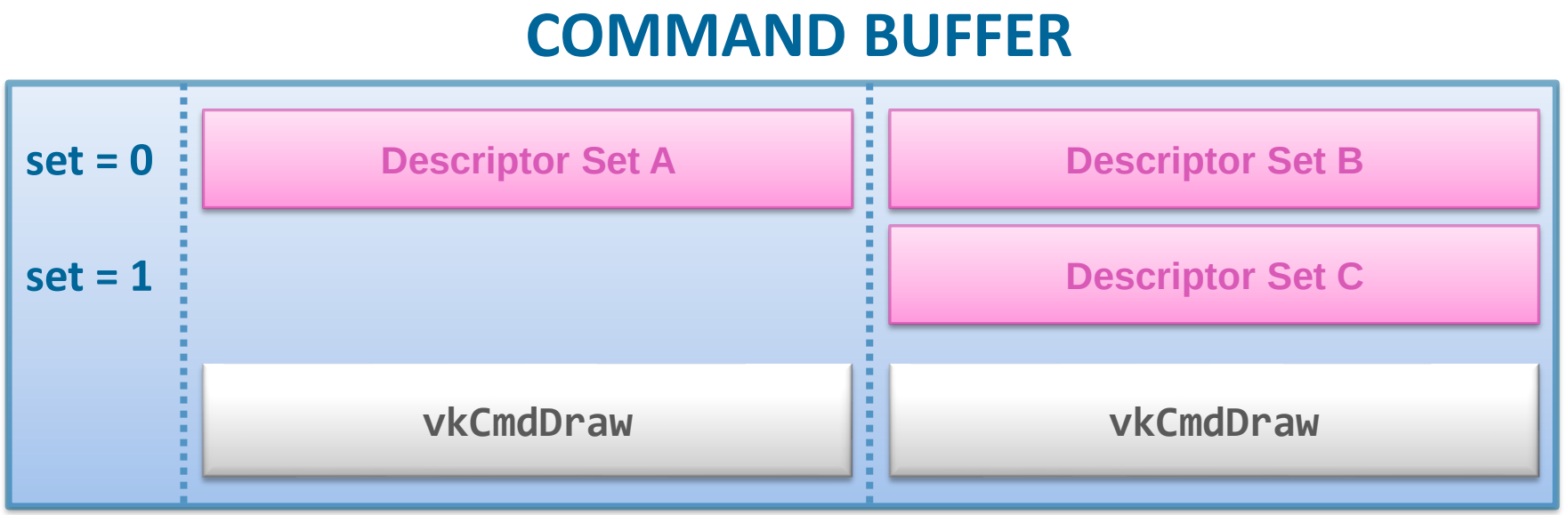


PART 3

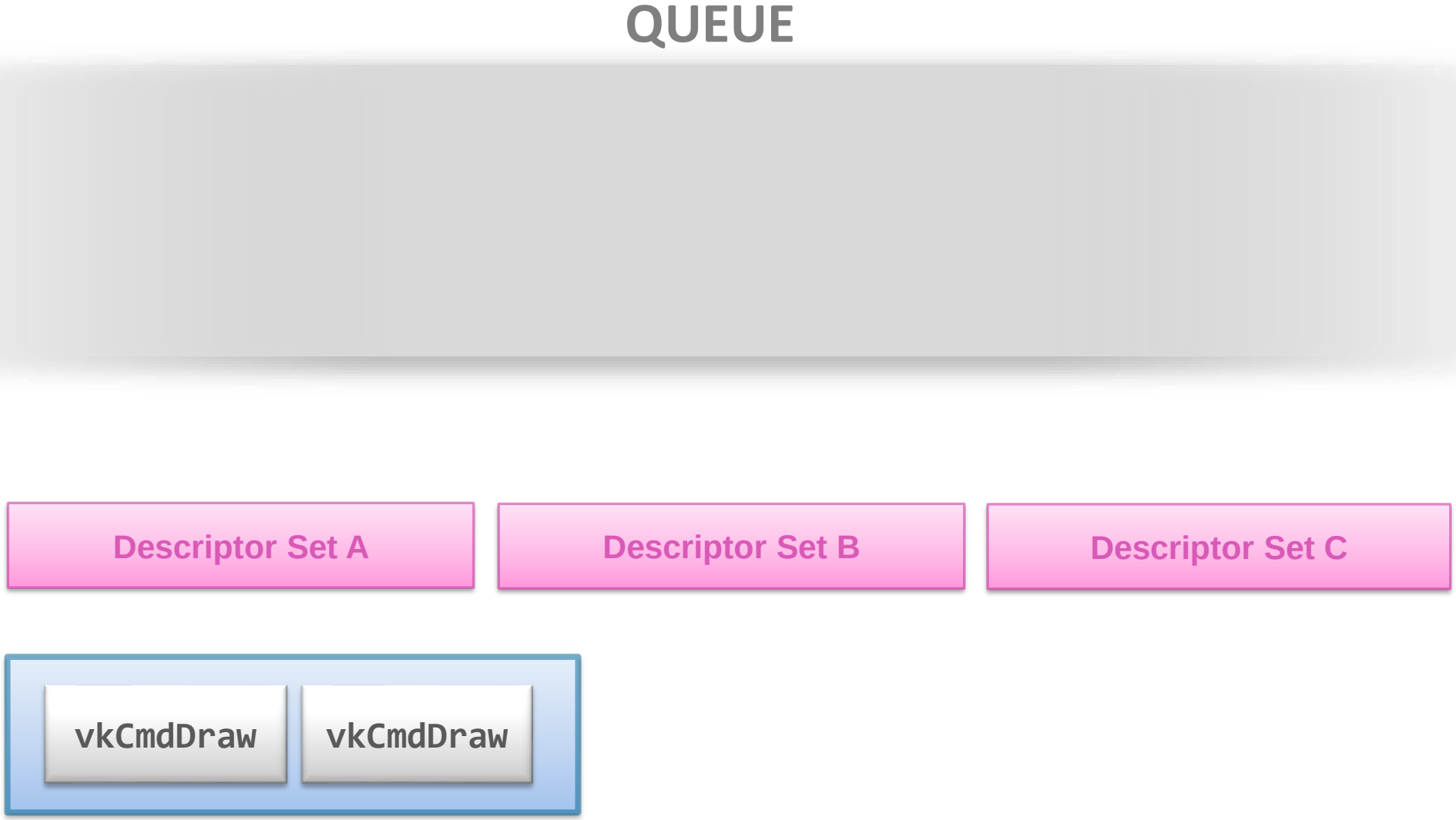
- Multiple Vertex Buffers
- **Command Buffer Recording**
- Multiple Graphics Pipelines
- Depth Test



Recap: Command Buffer and Descriptors

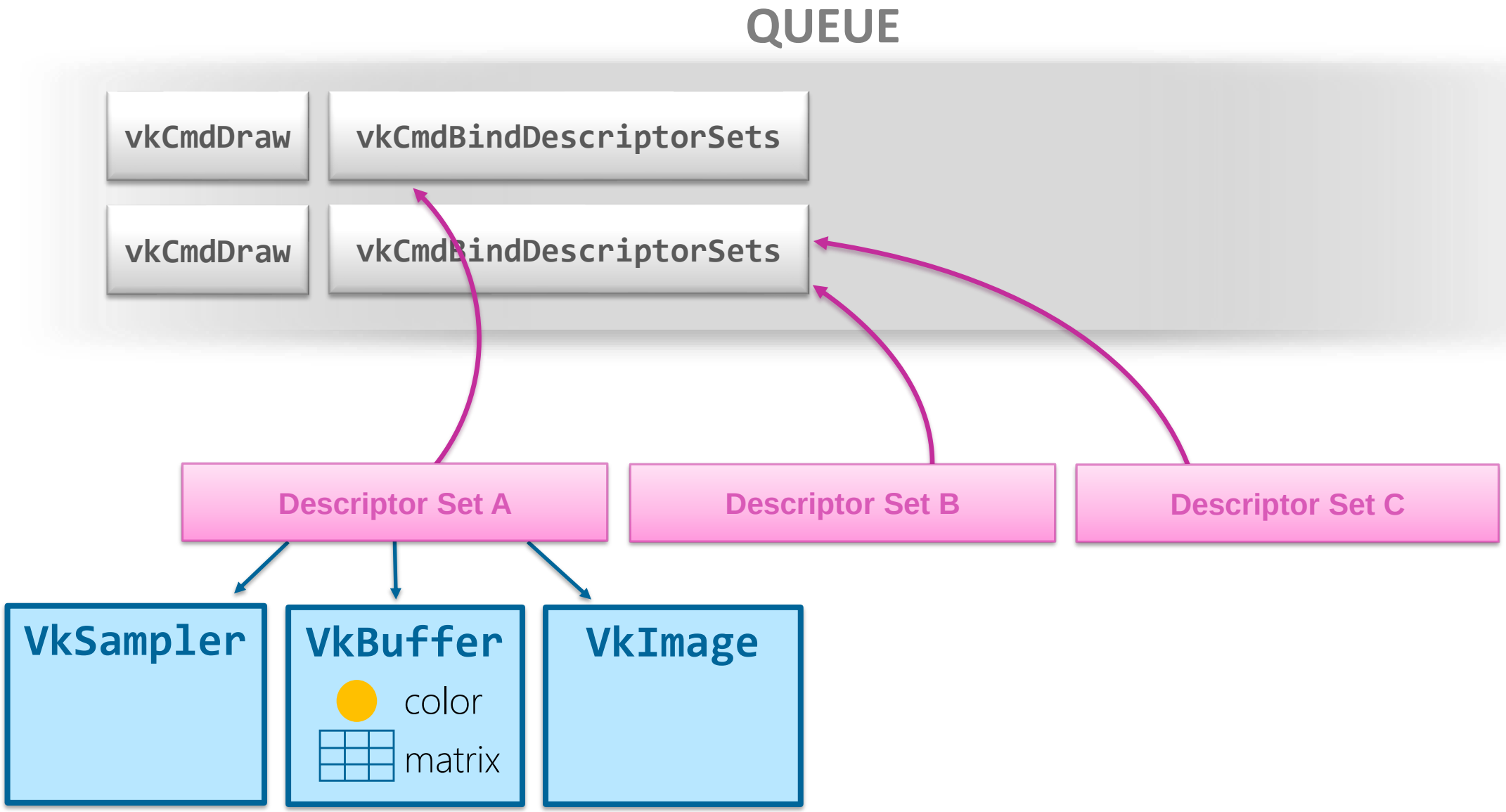


Recap: Command Buffer and Descriptors



Recap: Command Buffer and Descriptors

Recap: Command Buffer and Descriptors



Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```

Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```

Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

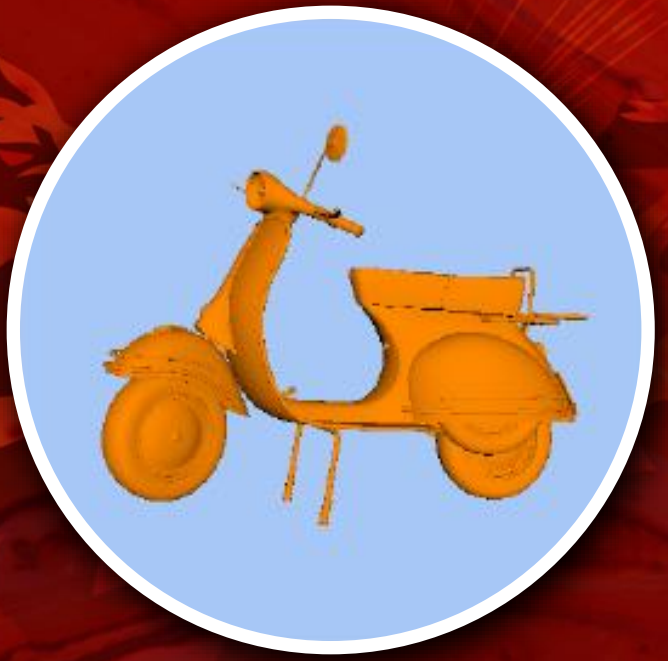
```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```


Command Buffer Recording

```
VkCommandBuffer command_buffer = // ...
VkPipeline pipeline = // ...
VkPipelineLayout pipeline_layout = // ...
VkDescriptorSet descriptor_set = // ...

vkCmdBindDescriptorSets(command_buffer,
    VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline_layout,
    0, 1, &descriptor_set,
    0, nullptr);

vkCmdBindPipeline(command_buffer, VK_PIPELINE_BIND_POINT_GRAPHICS, pipeline);

VkBuffer vertex_buffers[3] {
    buffer0, buffer1, buffer2
};
VkDeviceSize offsets[3]{ 0, 0, 0 };
vkCmdBindVertexBuffers(command_buffer, 0, 3, vertex_buffers, offsets);

vkCmdDraw(command_buffer, 3, 1, 0, 0);
```

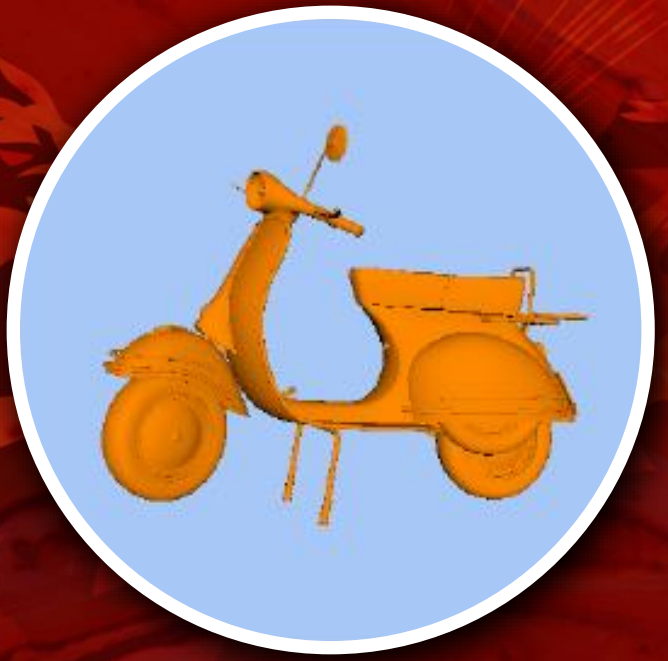
PART 3

- Multiple Vertex Buffers
- **Command Buffer Recording**
- Multiple Graphics Pipelines
- Depth Test



PART 3

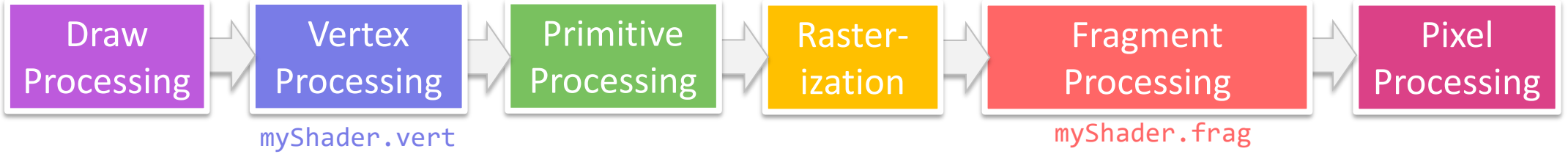
- Multiple Vertex Buffers
- Command Buffer Recording
- **Multiple Graphics Pipelines**
- Depth Test



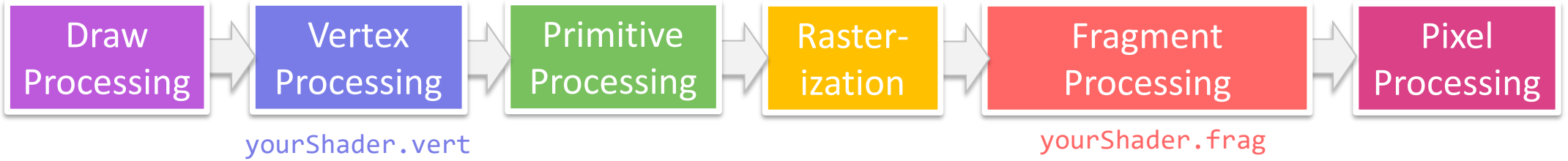
Graphics Pipelines

One specific graphics pipeline: `VkPipeline pipe1;`

A different, but still specific graphics pipeline: `VkPipeline pipe2;`



Yet another specific graphics pipeline: `VkPipeline pipe3;`

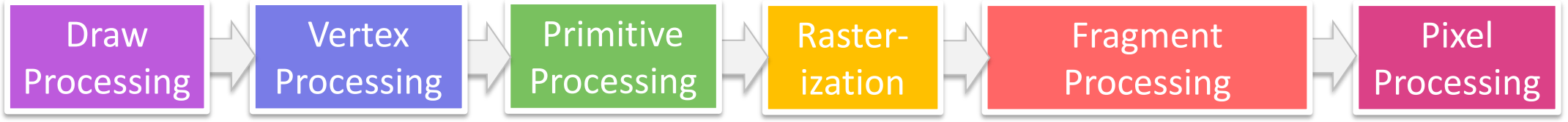


Graphics Pipelines

A different, but still specific graphics pipeline: `VkPipeline pipe2;`

Settings:
- Filled triangles (`VK_POLYGON_MODE_FILL`)
- Culling: Backfaces (`VK_CULL_MODE_BACK_BIT`)

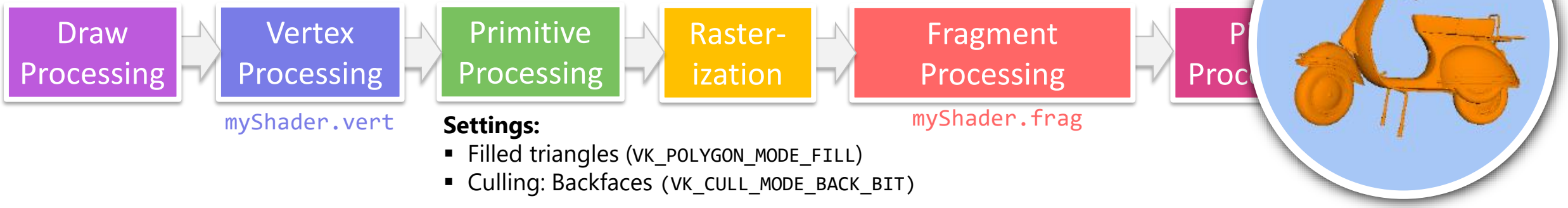
Yet another specific graphics pipeline: `VkPipeline pipe3;`



Settings:
- Lines/"Wireframe" (`VK_POLYGON_MODE_LINE`)
- Culling: Nothing (`VK_CULL_MODE_NONE`)

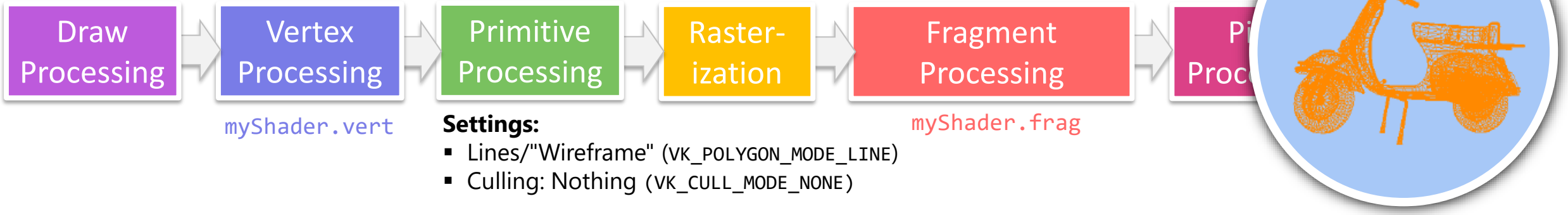
Graphics Pipelines

A different, but still specific graphics pipeline: `VkPipeline pipe2;`



Settings:
- Filled triangles (`VK_POLYGON_MODE_FILL`)
- Culling: Backfaces (`VK_CULL_MODE_BACK_BIT`)

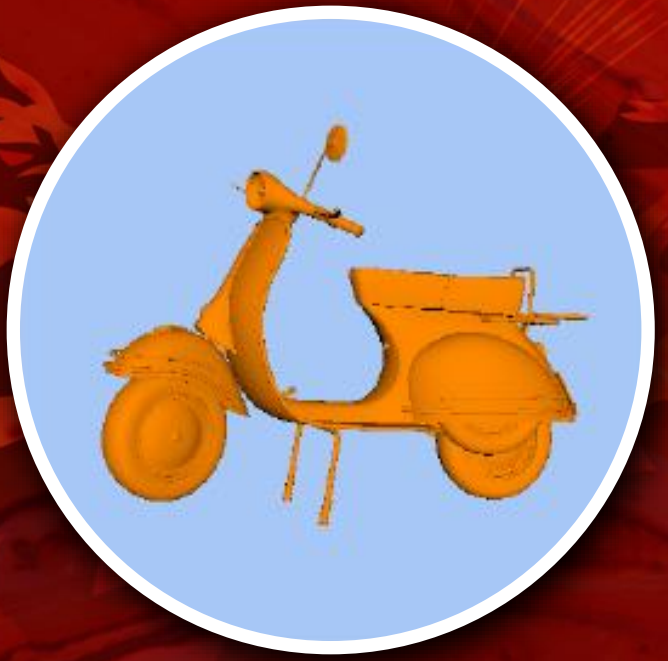
Yet another specific graphics pipeline: `VkPipeline pipe3;`



Settings:
- Lines/"Wireframe" (`VK_POLYGON_MODE_LINE`)
- Culling: Nothing (`VK_CULL_MODE_NONE`)

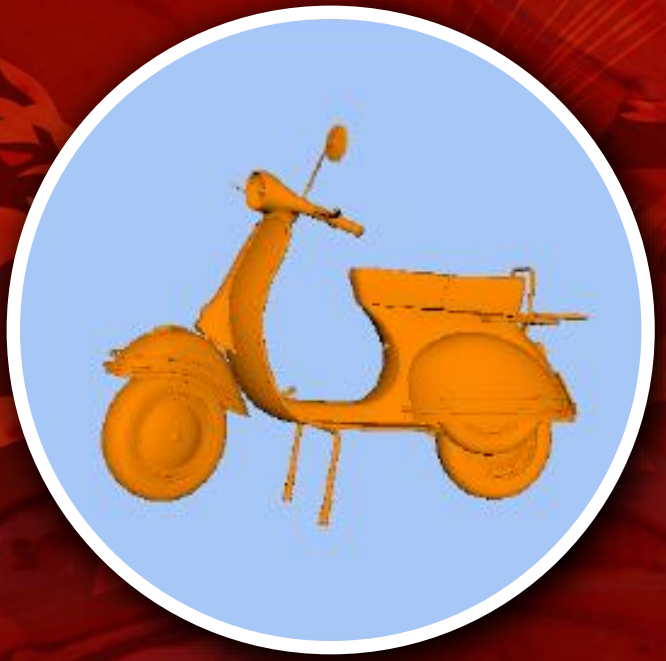
PART 3

- Multiple Vertex Buffers
- Command Buffer Recording
- **Multiple Graphics Pipelines**
- Depth Test

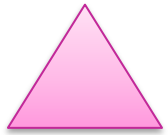


PART 3

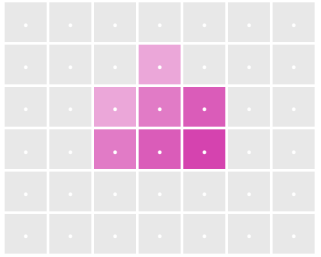
- Multiple Vertex Buffers
- Command Buffer Recording
- Multiple Graphics Pipelines
- **Depth Test**



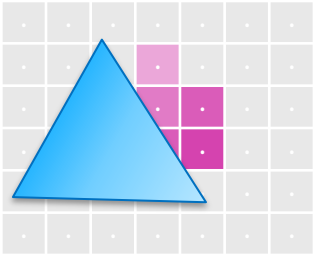
Without Depth Test



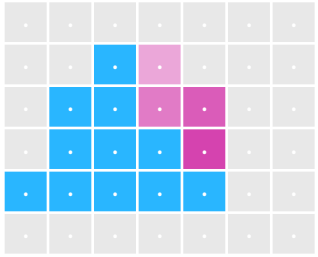
Without Depth Test



Without Depth Test

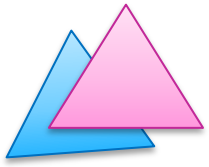


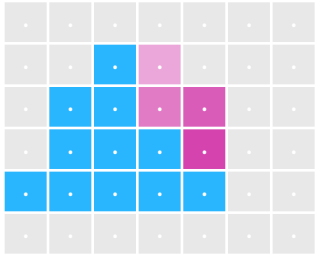
Without Depth Test



Without Depth Test

No depth buffer used

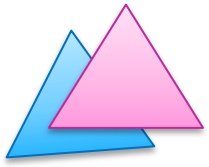




With Depth Test

Solution:

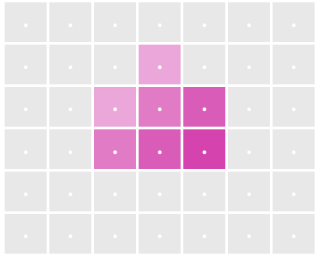
- Use depth buffer image (`VkImage`)
- Image usage: `VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT`
- Add to framebuffer
  (i.e., color attachment *and* depth attachment)



With Depth Test

Solution:

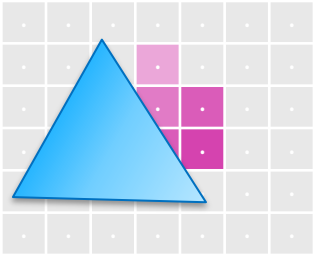
- Use depth buffer image (`VkImage`)
- Image usage: `VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT`
- Add to framebuffer
  (i.e., color attachment *and* depth attachment)



With Depth Test

Solution:

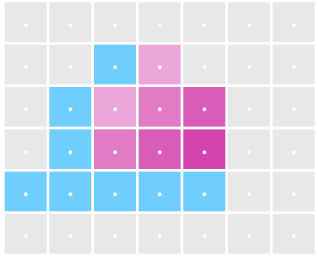
- Use depth buffer image (`VkImage`)
- Image usage: `VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT`
- Add to framebuffer
  (i.e., color attachment *and* depth attachment)



With Depth Test

Solution:

- Use depth buffer image (`VkImage`)
- Image usage: `VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT`
- Add to framebuffer
  (i.e., color attachment *and* depth attachment)



PART 3

- Multiple Vertex Buffers
- Command Buffer Recording
- Multiple Graphics Pipelines
- Depth Test

Schedule

PART 1:

Setup	Lecture	Coding Session
10 min	**20** min	**90** min
Starts at 09:00	Starts at 09:10	Starts at 09:30

PART 2:

Lecture	Coffee Break	Coding Session
15 min	**25** min	**80** min
Starts at 11:00	Starts at 11:15	Starts at 11:40

Lunch Break 13:00 – 14:00

PART 3:

Lecture	Coding Session	Coffee Break
15 min	**65** min	**30** min
Starts at 14:00	Starts at 14:15	Starts at 15:20

PART 4:

Lecture	Coding Session	Closing
20 min	**70** min	**10** min
Starts at 15:50	Starts at 16:10	Starts at 17:20