An Introduction to Vulkan

Johannes Unterguggenberger
TU Wien, Huawei
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
Graphics Pipelines

Draw Processing → Vertex Processing → Tessellation → Primitive Processing → Rasterization → Fragment Processing → Pixel Processing
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing
Graphics Pipelines

Draw Processing → Vertex Processing → Tessellation → Primitive Processing → Rasterization → Fragment Processing → Pixel Processing
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader

fragment shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

vertex shader

fragment shader
Graphics Pipelines

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

- vertex shader
- fragment shader
Graphics Pipelines

=> ▪ Use a **graphics pipeline** (rasterizer!)
  ▪ Which shaders to use? (e.g., **vertex** and **fragment**)
  ▪ Which geometry as **input** (e.g., triangles)
  ▪ Which **resources** to bind? (through descriptors)

**vkCmdDraw**

- Draw Processing
- Vertex Processing
- Tessellation
- Primitive Processing
- Rasterization
- Fragment Processing
- Pixel Processing

**vertex shader**

**fragment shader**
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
#version 450

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

layout (location = 0) in vec3 in_position;

void main() {
    gl_Position = uniform_buffer.transformationMatrix * vec4(in_position, 1.0);
}
#version 450

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

layout (location = 0) in vec3 in_position;

void main() {
  gl_Position = uniform_buffer.transformationMatrix * vec4(in_position, 1.0);
}
GLSL vertex shader

```glsl
#version 450

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

layout (location = 0) in vec3 in_position;

void main() {
    gl_Position = uniform_buffer.transformationMatrix * vec4(in_position, 1.0);
}
```

vertex shader

fragment shader
#version 450

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

layout (location = 0) in vec3 in_position;

void main() {
    gl_Position = uniform_buffer.transformationMatrix * vec4(in_position, 1.0);
}
```glsl
#version 450

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

layout (location = 0) out vec4 out_color;

void main() {
    out_color = uniform_buffer.color;
}
```
```glsl
#version 450

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

layout (location = 0) out vec4 out_color;

void main() {
    out_color = uniform_buffer.color;
}
```
GLSL fragment shader

```glsl
#version 450

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

layout (location = 0) out vec4 out_color;

void main() {
    out_color = uniform_buffer.color;
}
```

Draw Processing ➔ Vertex Processing ➔ Tessellation ➔ Primitive Processing ➔ Rasterization ➔ Fragment Processing ➔ Pixel Processing

vertex shader

fragment shader
Buffers

=> ▪ Use a **graphics pipeline** (rasterizer!)
  ▪ Which shaders to use? (e.g., *vertex* and *fragment*)
  ▪ Which geometry as **input** (e.g., triangles)
  ▪ Which **resources** to bind? (through descriptors)

Vertex Processing → Tessellation → Primitive Processing → Rasterization → Fragment Processing → Pixel Processing

vertex shader

fragment shader
• Which geometry as **input** (e.g., triangles)

• Which **resources** to bind? (through descriptors)

```
VkBuffer
Buffers

"Vertex Buffer"

"Uniform Buffer"

vertex shader

fragment shader

vkCmdDraw

```

**Buffers**

Which geometry as **input** (e.g., triangles)

Which **resources** to bind? (through descriptors)
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
**Buffers**

- Which **input** geometry as input (e.g., triangles)

- Which **resources** to bind? (through descriptors)

---

**Vertex Buffer**

```
vkCmdDraw
```

---

**Descriptor**

```
VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER
```

---

**Uniform Buffer**

```
vkBuffer
```

---

**Vertex shader**

**Fragment shader**

---

**Draw Processing**

**Vertex Processing**

**Tessellation**

**Primitive Processing**

**Rasterization**

**Fragment Processing**

**Pixel Processing**
Descriptors and Descriptor Sets

- One **descriptor** describes one **resource**

![Descriptor Diagram]

**Descriptor**

```
VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER
```
Descriptors and Descriptor Sets

- One descriptor describes one resource
- Descriptors are organized in descriptor sets
Descriptors and Descriptor Sets

- One descriptor describes one resource
- Descriptors are organized in descriptor sets

**Descriptor Set A**

- **Descriptor**
  - VK_DESCRIPTOR_TYPE_SAMPLER
    - binding = 0

- **Descriptor**
  - VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER
    - binding = 1

- **Descriptor**
  - VK_DESCRIPTOR_TYPE_SAMPLED_IMAGE
    - binding = 2
Descriptors and Descriptor Sets

- One **descriptor** describes one **resource**
- Descriptors are organized in **descriptor sets**
- Descriptor set bound while recording a command buffer

**COMMAND BUFFER**

set = 0

Descriptor Set A

vkCmdDraw
Descriptors and Descriptor Sets

- One **descriptor** describes one **resource**
- Descriptors are organized in **descriptor sets**
- Descriptor set bound while recording a command buffer

**COMMAND BUFFER**

<table>
<thead>
<tr>
<th>set = 0</th>
<th>set = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptor Set A</td>
<td>Descriptor Set B</td>
</tr>
<tr>
<td></td>
<td>Descriptor Set C</td>
</tr>
<tr>
<td>vkCmdDraw</td>
<td>vkCmdDraw</td>
</tr>
</tbody>
</table>
Descriptors and Descriptor Sets

vkCmdDraw

Descriptor Set A
Descriptor Set B
Descriptor Set C
Descriptors and Descriptor Sets

**Queue**

- `vkCmdDraw`
- `vkCmdDraw`

- Descriptor Set A
- Descriptor Set B
- Descriptor Set C
Descriptors and Descriptor Sets

- VkCmdDraw
- VkBuffer
  - color
  - matrix
- VkImage
- VkSampler

Descriptor Set A
Descriptor Set B
Descriptor Set C
"Vertex Buffer"

VkBuffer

vertex shader

draw processing

tessellation

vkCmdDraw

fragment shader

fragment processing

Descriptor

VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER

"Uniform Buffer"

VkBuffer

color

matrix

"Vertex Buffer"

"Uniform Buffer"
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
PART 2

- Graphics Pipelines
- Uniform Buffers
- Descriptors
PART 1:

- **Setup**
  - Duration: 10 min
  - Starts at: 09:00

- **Lecture**
  - Duration: 20 min
  - Starts at: 09:10

- **Coding Session**
  - Duration: 90 min
  - Starts at: 09:30

PART 2:

- **Lecture**
  - Duration: 15 min
  - Starts at: 11:00

- **Coffee Break**
  - Duration: 25 min
  - Starts at: 11:15

- **Coding Session**
  - Duration: 80 min
  - Starts at: 11:40

**Lunch Break**
- Duration: 13:00 – 14:00

PART 3:

- **Lecture**
  - Duration: 15 min
  - Starts at: 14:00

- **Coding Session**
  - Duration: 65 min
  - Starts at: 14:15

- **Coffee Break**
  - Duration: 30 min
  - Starts at: 15:20

PART 4:

- **Lecture**
  - Duration: 20 min
  - Starts at: 15:50

- **Coding Session**
  - Duration: 70 min
  - Starts at: 16:10

- **Closing**
  - Duration: 10 min
  - Starts at: 17:20