Vulkan

Paolo Bettelini

Contents

1 Vulkan **2**

2 Queue **2**

3 Buffers **2**

4 Command buffers **2**
4.1 Primary command buffers 2
4.2 Secondary command buffers 2

5 Compute pipelines **2**

6 Descriptors **2**

7 Dispatch **3**

8 Images **3**

9 Graphics pipelines **3**
9.1 Vertex shaders 3
9.2 Fragment shaders 3
9.3 Vertex buffer 3
9.4 Render passes 3

1 Vulkan

Vulkan is a multi-platform low-level graphical interface (API) for GPU rendering.

The GPU is able to perform computation on a lot of data simultaneously (SIMD, Single Instruction stream, Multiple Data stream).

2 Queue

The GPU is able to run multiple operations in parallel. The equivalent of a CPU thread is a **queue**. Queues are grouped by **queue families**.

Whenever we want the device to perform an operation, we have to submit this operation to a specific queue under a family. Some queues support only graphical operations, some others support only compute operations, and some others support both.

3 Buffers

When we need the GPU to read or write data in memory, we need to use a **buffer** (or an image, later section).

4 Command buffers

To execute an operation we need to create a command buffer containg a list of commands to execute.

To submit a command buffer we need to synchronize with the GPU. We can also tell the GPU to send back a signal, call **fence** when the operation is done.

4.1 Primary command buffers

They can contain any command. They are the only type of command buffer that can be submitted to a queue.

4.2 Secondary command buffers

They allow you to store functionality that you can reuse multiple times in primary command buffers.

5 Compute pipelines

In order to ask the GPU to perform an operation on some data, we need to write a program for it. A program that runs on the GPU is called a shader.

Shaders are written in a shading language (hlsl, glsl, wgpu, rust-gpu, ...) which is then compiled into an intermediate bytecode called **SPIR-V**.

6 Descriptors

When we create a compute pipeline for a shader we must bound it to a **descriptor**. A descriptor can contain a buffer to access, buffer viewes, images, samples images etc.

Descriptors are grouped by **descriptor sets**. The shader will declare a specific descriptor from a specific set (both indexed from 0).

7 Dispatch

To execute a compute pipeline we need to create a command buffer to do so. This is called **Dispatch**.

8 Images

Another way to make the GPU interact with memory is with an **image**. Images are often 2-dimensional array of pixels (there also also 3-dimensional images). The pixels of the image are often referred to as **texels**.

Each pixel can have up to four components.

You can't directly change the content of an image as you would for a buffer. For example, we could create a command buffer to tell the GPU to fill an image with white pixels.

9 Graphics pipelines

A graphics pipeline is the same as a compute pipeline but for graphical operations. Graphics pipelines are more restrictive than compute operations, but they're also much faster.

The purpose of this pipeline is to draw a shape on an image.

9.1 Vertex shaders

The first step when executing a graphics operation is the **vertex shader**. This shader is executed for each vertex of the shape.

9.2 Fragment shaders

The GPU will compute which pixels are in the given shape, and will execute for each of them the **fragment shader**.

9.3 Vertex buffer

Vertex buffer is the name given the the buffer containing the certices of the shape. The shape is made out of triangles (3 vertices each).

9.4 Render passes

Before executing a graphics operation we must tell the GPU to enter a "rendering mode", by entering what is called a **render pass**.

A render pass is made of **attachments** and **passes**.