Работа с командами в Vulkan API

Быковских Дмитрий Александрович

12.10.2024

Работа с командами в Vulkan API
Быковских Дмитрий Александрович
12.10.2024

Содержание

Содержание
- Концептуальная модель Vulkan API
- Объекты Vulkan API
- Очередь (Queue)
- Буфер команд (Command Buffer) и пул команд (Command Pool)
- Команды (Commands)
- Синхронизация (Synchronization)

Концептуальная модель Vulkan API

Концептуальная модель Vulkan API представляет собой низкоуровневую архитектуру для работы с графическими и вычислительными задачами на GPU, предоставляя разработчикам прямой контроль над аппаратными ресурсами.

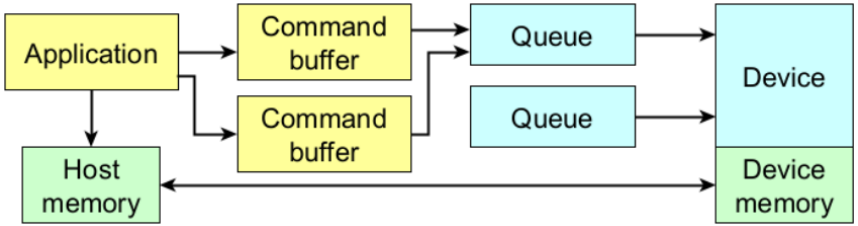
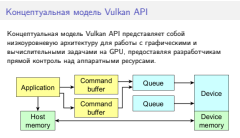


Рис. 1: Схема концептуальной модели Vulkan API



Особенности:

- Высокий уровень параллелизма (Формирование буфера команд)
- Асинхронность (Выполнение команд через очереди Queue)
- Явное (низкоуровневое) управление памятью (Host memory и Device memory)

Объекты Vulkan API

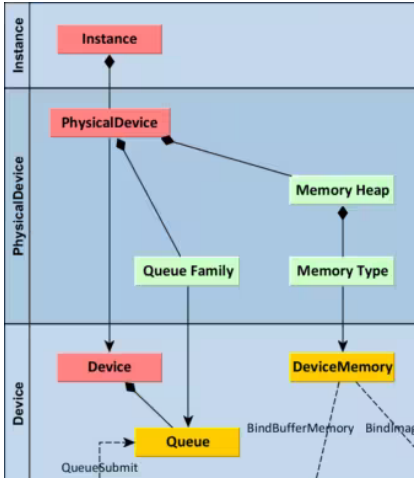
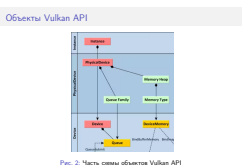


Рис. 2: Часть схемы объектов Vulkan API



Функции создания объектов

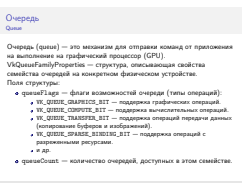
- vkCreateInstance — точка взаимодействия с драйвером (м.б. несколько)
- vkEnumeratePhysicalDevices — выбор физических устройств
- vkGetPhysicalDeviceQueueFamilyProperties — выбор поддерживаемых типов очередей
- vkCreateDevice — логическое устройство, с поддерживаемыми типами очередей, памятью и т.д.
- vkGetDeviceQueue — дескриптор очереди для отправки команд на физической устройство через драйвер

Очередь

Queue

Очередь (queue) — это механизм для отправки команд от приложения на выполнение на графический процессор (GPU).
VkQueueFamilyProperties — структура, описывающая свойства семейства очередей на конкретном физическом устройстве.
Поля структуры:

- `queueFlags` — флаги возможностей очереди (типы операций):
  - `VK_QUEUE_GRAPHICS_BIT` — поддержка графических операций.
  - `VK_QUEUE_COMPUTE_BIT` — поддержка вычислительных операций.
  - `VK_QUEUE_TRANSFER_BIT` — поддержка операций передачи данных (копирование буферов и изображений).
  - `VK_QUEUE_SPARSE_BINDING_BIT` — поддержка операций с разреженными ресурсами.
  - и др.
- `queueCount` — количество очередей, доступных в этом семействе.



Поля структуры (продолжение)

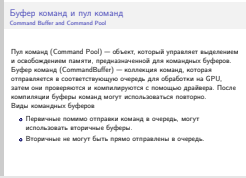
- `minImageTransferGranularity` — минимальная единица передачи изображений (размеры элемента по X, Y и Z). Полезно для операций передачи изображения и использования разреженных текстур (sparse textures), где может понадобиться копировать данные с определённой минимальной единицей.
- `timestampValidBits` — количество валидных битов в метках времени (timestamps). Если поддерживается, это число указывает, сколько бито будет использоваться для меток времени, полученных с помощью команд, вроде `vkCmdWriteTimestamp`.
- и др.

Буфер команд и пул команд

Command Buffer and Command Pool

Пул команд (Command Pool) — объект, который управляет выделением и освобождением памяти, предназначенной для командных буферов.
Буфер команд (CommandBuffer) — коллекция команд, которая отправляется в соответствующую очередь для обработки на GPU, затем они проверяются и компилируются с помощью драйвера. После компиляции буферы команд могут использоваться повторно.
Виды командных буферов

- Первичные помимо отправки команд в очередь, могут использовать вторичные буферы.
- Вторичные не могут быть прямо отправлены в очередь.



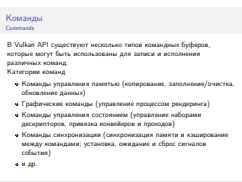
Примечания.
1. Пул команд зависит от семейства очередей, позволяет многократно использовать эти командные буферы, что минимизирует накладные расходы на запись команд.
2. Каждый буфер команд управляет своим состояниями независимо. Существует три состояния буфера команд:

- Начальное состояние (до записи или при сбросе)
- Состояние записи (между вызовами vkBeginCommandBuffer и vkEndCommandBuffer)
- Состояние выполнения (отправка на выполнение)

Команды
Commands

В Vulkan API существуют несколько типов командных буферов, которые могут быть использованы для записи и исполнения различных команд.
Категории команд

- Команды управления памятью (копирование, заполнение/очистка, обновление данных)
- Графические команды (управление процессом рендеринга)
- Команды управления состоянием (управление наборами дескрипторов, привязка конвейеров и проходов)
- Команды синхронизации (синхронизация памяти и кэширование между командами; установка, ожидание и сброс сигналов события)
- и др.

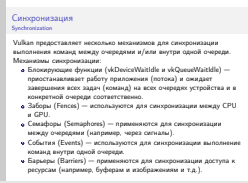


Синхронизация

Synchronization

Vulkan предоставляет несколько механизмов для синхронизации выполнения команд между очередями и/или внутри одной очереди.
Механизмы синхронизации:

- Блокирующие функции (vkDeviceWaitIdle и vkQueueWaitIdle) — приостанавливает работу приложения (потока) и ожидает завершения всех задач (команд) на всех очередях устройства и в конкретной очереди соответственно.
- Заборы (Fences) — используются для синхронизации между CPU и GPU.
- Семафоры (Semaphores) — применяются для синхронизации между очередями (например, через сигналы).
- События (Events) — используются для синхронизации выполнение команд внутри одной очереди.
- Барьеры (Barriers) — применяются для синхронизации доступа к ресурсам (например, буферам и изображениям и т.д.).



Выполнение команд в происходит через очереди.
Вообще говоря команды выполняются асинхронно.
Команды могут быть различной сложности, а следовательно длиться разное количество времени.