Queues and Command Buffers

Application

Vulkan: Overall Block Diagram

Vulkan: Overall Block Diagram

Vulkan Queues and Command Buffers

Vulkan Queues and Command Buffers

Querying what Queue Families are Available

Similarly, we Can Write a Function that Finds the Proper Queue Family

Vulkan: a More Typical (and Simplified) Block Diagram
Creating a Logical Device Queue Needs to Know Queue Family Information

```c
VkDeviceQueueCreateInfo vdqci[1];
{
    uint32_t  queueIndex = 0;
    uint32_t  queueFamilyIndex = FindQueueFamilyThatDoesGraphics();
    VkQueue Queue;
    VkDeviceCreateInfo vdci;
    float   queuePriorities[ ] =
            vdqci.queuePriorities = (float *) queuePriorities;
            vdqci.queueCount = 1;
            vdqci.queueFamilyIndex = FindQueueFamilyThatDoesGraphics();
            vdqci.flags = 0;
            vdqci.pNext = nullptr;
            vdqci.sType = VK_STRUCTURE_TYPE_QUEUE_CREATE_INFO;

    result = vkCreateDevice(PhysicalDevice, IN &vdci, PALLOCATOR, OUT &LogicalDevice);
}
```

Creating the Command Buffers

```c
void CreateCommandBuffers() {
    VkCommandBufferAllocateInfo vcbbi;
    vcbbi.commandBufferCount = 1;
    vcbbi.level = VK_COMMAND_BUFFER_LEVEL_PRIMARY;
    vcbbi.commandPool = CommandPool;
    vcbbi.pNext = nullptr;
    vcbbi.sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_ALLOCATE_INFO;
    result = vkAllocateCommandBuffers(LogicalDevice, IN &vcbbi, PALLOCATOR, OUT &CommandBuffers[0]);
}
```

Beginning a Command Buffer

```c
void BeginCommandBuffer() {
    VkCommandBufferBeginInfo vcbbi;
    vcbbi.pInheritanceInfo = (VkCommandBufferInheritanceInfo *)nullptr;
    vcbbi.flags = VK_COMMAND_BUFFER_USAGE_ONE_TIME_SUBMIT_BIT;
    vcbbi.pNext = nullptr;
    vcbbi.sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_BEGIN_INFO;
    result = vkBeginCommandBuffer(CommandBuffers[0], IN &vcbbi);  // 1. Create a single command buffer for the double-buffered rendering
}
```

These are the Commands that could be entered into the Command Buffer, I
These are the Commands that could be entered into the Command Buffer, II

- `vkCmdWriteTimestamp(commandBuffer, pipelineStage, queryPool, query);`
- `vkCmdWaitEvents(commandBuffer, eventCount, pEvents, srcStageMask, dstStageMask, memoryBarrierCount, pMemoryBarriers);`
- `vkCmdUpdateBuffer(commandBuffer, dstBuffer, dstOffset, dataSize, pData);`
- `vkCmdSetViewport(commandBuffer, firstViewport, viewportCount, pViewports);`
- `vkCmdSetScissor(commandBuffer, firstScissor, scissorCount, pScissors);`
- `vkCmdSetLineWidth(commandBuffer, lineWidth);`
- `vkCmdSetEvent(commandBuffer, event, stageMask);`
- `vkCmdSetDiscardRectangleEXT(commandBuffer, firstDiscardRectangle, discardRectangleCount, pDiscardRectangles);`
- `vkCmdSetDeviceMaskKHX(commandBuffer, deviceMask);`
- `vkCmdSetViewportWScalingNV(commandBuffer, firstViewport, viewportCount, pViewportWScalings);`
- `vkCmdProcessCommandsNVX(commandBuffer, pProcessCommandsInfo);`
- `vkCmdFillBuffer(commandBuffer, dstBuffer, dstOffset, size, data);`
- `vkCmdSetDepthBias(commandBuffer, depthBiasConstantFactor, depthBiasClamp, depthBiasSlopeFactor);`
- `vkCmdSetBlendConstants(commandBuffer, blendConstants[4]);`
- `vkCmdResolveImage(commandBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, regionCount, pRegions);`
- `vkCmdResetEvent(commandBuffer, event, stageMask);`
- `vkCmdPipelineBarrier(commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, VkMemoryBarrier* pMemoryBarriers, bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers);`

The Entire Submission / Wait / Display Process

1. **Create fence**
2. **Fill in the queue**
3. **Submit the queue**
4. **Wait for the fence**

**Submitting a Command Buffer to a Queue for Execution**

```cpp
VKCmdBeginInfo vcbi;  // void:

vkBeginCommandBuffer(commandBuffer, 0, 1, IN &vcbi);
```