Queues and Command Buffers

Application

Vulkan: Overall Block Diagram

Vulkan: a More Typical (and Simplified) Block Diagram

Vulkan Queues and Command Buffers

- Graphics commands are recorded in command buffers, e.g., `vkCmdDoSomething( cmdBuffer, ... );`
- You can have as many simultaneous Command Buffers as you want
- Each command buffer can be filled from a different thread
- Command Buffers record our commands, but no work takes place until a Command Buffer is submitted to a Queue
- We don't create Queues – the Logical Device has them already
- Each Queue belongs to a Queue Family
- We don't create Queue Families – the Physical Device already has them

// These are not the Queue Families you're looking for.

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

```cpp
int FindQueueFamilyThatDoesGraphics( ) {
    uint32_t count = -1;
    vkGetPhysicalDeviceQueueFamilyProperties( IN PhysicalDevice, &count, OUT (VkQueueFamilyProperties *)nullptr ) ;
    VkQueueFamilyProperties *vqfp = new VkQueueFamilyProperties[ count ];
    vkGetPhysicalDeviceQueueFamilyProperties( PhysicalDevice, &count, OUT vqfp );
    for( unsigned int i = 0; i < count; i++ ) {
        if( ( vqfp[i].queueFlags & VK_QUEUE_GRAPHICS_BIT ) != 0 )
            return i;
    }
    return -1;
}
```
Creating a Logical Device Queue Needs to Know Queue Family Information

```
VkDeviceQueueCreateInfo vdqci[1];
{
    .queuePriorities = (float *) queuePriorities;
    .queueFamilyIndex = FindQueueFamilyThatDoesGraphics();
    .flags = 0;
    .pNext = nullptr;
    .sType = VK_STRUCTURE_TYPE_QUEUE_CREATE_INFO;
}
```

Creating the Command Pool as part of the Logical Device

```
VkCommandPoolCreateInfo vcpci;
VkResult result;
{
    .flags = VK_COMMAND_POOL_CREATE_RESET_COMMAND_BUFFER_BIT | VK_COMMAND_POOL_CREATE_TRANSIENT_BIT;
    .pNext = nullptr;
    .sType = VK_STRUCTURE_TYPE_COMMAND_POOL_CREATE_INFO;
}
```

Beginning a Command Buffer

```
vkBeginCommandBuffer(vcbai, CommandBuffers[nextImageIndex], IN &LogicalDevice);
```

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

```
vkCmdExecuteCommands(commandBuffer, commandBufferCount, const pCommandBuffers);
vkCmdEndQuery(commandBuffer, query);
vkCmdDrawIndirectCountAMD(commandBuffer, stride);
vkCmdDrawIndexedIndirectCountAMD(commandBuffer, stride);
vkCmdDrawIndexed(commandBuffer, indexCount, instanceCount, firstIndex, int32_t vertexOffset, firstInstance);
vkCmdDraw(commandBuffer, vertexCount, instanceCount, firstVertex, firstInstance);
vkCmdDispatchIndirect(commandBuffer, offset);
vkCmdDispatch(commandBuffer, groupCountX, groupCountY, groupCountZ);
vkCmdDebugMarkerInsertEXT(commandBuffer, pMarkerInfo);
vkCmdDebugMarkerEndEXT(commandBuffer);
vkCmdDebugMarkerBeginEXT(commandBuffer, pMarkerInfo);
vkCmdCopyQueryPoolResults(commandBuffer, flags);
vkCmdCopyImage(commandBuffer, pRegions);
vkCmdCopyBuffer(commandBuffer, pRegions);
vkCmdClearDepthStencilImage(commandBuffer, pRanges);
vkCmdClearColorImage(commandBuffer, pRanges);
vkCmdClearAttachments(commandBuffer, attachmentCount, const pRects);
vkCmdBlitImage(commandBuffer, filter);
vkCmdBindVertexBuffers(commandBuffer, firstBinding, bindingCount, const pOffsets);
vkCmdBindPipeline(commandBuffer, pipeline);
vkCmdBindIndexBuffer(commandBuffer, indexType);
vkCmdBeginRenderPass(commandBuffer, const contents);
vkCmdBeginQuery(commandBuffer, flags);
```
These are the Commands that could be entered into the Command Buffer, II

vkCmdWaitEvents( commandBuffer, eventCount, pEvents, srcStageMask, dstStageMask, memoryBarrierCount, pMemoryBarriers, ... )
vkCmdUpdateBuffer( commandBuffer, dstBuffer, dstOffset, dataSize, pData );
vkCmdSetViewport( commandBuffer, firstViewport, viewportCount, pViewports );
vkCmdSetStencilReference( commandBuffer, faceMask, reference );
vkCmdSetScissor( commandBuffer, firstScissor, scissorCount, pScissors );
vkCmdSetLineWidth( commandBuffer, lineWidth );
vkCmdSetEvent( commandBuffer, event, stageMask );
vkCmdSetDeviceMaskKHX( commandBuffer, deviceMask );
vkCmdSetViewportWScalingNV( commandBuffer, firstViewport, viewportCount, pViewportWScalings );
vkCmdSetDepthBounds( commandBuffer, minDepthBounds, maxDepthBounds );
vkCmdReserveSpaceForCommandsNVX( commandBuffer, pReserveSpaceInfo );
vkCmdPushDescriptorSetWithTemplateKHR( commandBuffer, descriptorUpdateTemplate, layout, set, pData );
vkCmdPushDescriptorSetKHR( commandBuffer, pipelineBindPoint, layout, set, descriptorWriteCount, pDescriptorWrites );
vkCmdPushConstants( commandBuffer, layout, stageFlags, offset, size, pValues );

bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers );
vkCmdFillBuffer( commandBuffer, dstBuffer, dstOffset, size, data );
vkCmdResolveImage( commandBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, regionCount, pRegions );
vkCmdResetQueryPool( commandBuffer, queryPool, firstQuery, queryCount );
vkCmdResetEvent( commandBuffer, event, stageMask );
vkCmdPipelineBarrier( commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, VkMemoryBarrier* pMemoryBarriers, ... )

Submitting a Command Buffer to a Queue for Execution

The Entire Submission / Wait / Display Process

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

vkCmdWaitEvents( commandBuffer, eventCount, pEvents, srcStageMask, dstStageMask, memoryBarrierCount, pMemoryBarriers, ... )
vkCmdUpdateBuffer( commandBuffer, dstBuffer, dstOffset, dataSize, pData );
vkCmdSetViewport( commandBuffer, firstViewport, viewportCount, pViewports );
vkCmdSetStencilReference( commandBuffer, faceMask, reference );
vkCmdSetScissor( commandBuffer, firstScissor, scissorCount, pScissors );
vkCmdSetLineWidth( commandBuffer, lineWidth );
vkCmdSetEvent( commandBuffer, event, stageMask );
vkCmdSetDeviceMaskKHX( commandBuffer, deviceMask );
vkCmdSetViewportWScalingNV( commandBuffer, firstViewport, viewportCount, pViewportWScalings );
vkCmdSetDepthBounds( commandBuffer, minDepthBounds, maxDepthBounds );
vkCmdReserveSpaceForCommandsNVX( commandBuffer, pReserveSpaceInfo );
vkCmdPushDescriptorSetWithTemplateKHR( commandBuffer, descriptorUpdateTemplate, layout, set, pData );
vkCmdPushDescriptorSetKHR( commandBuffer, pipelineBindPoint, layout, set, descriptorWriteCount, pDescriptorWrites );
vkCmdPushConstants( commandBuffer, layout, stageFlags, offset, size, pValues );

bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers );
vkCmdFillBuffer( commandBuffer, dstBuffer, dstOffset, size, data );
vkCmdResolveImage( commandBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, regionCount, pRegions );
vkCmdResetQueryPool( commandBuffer, queryPool, firstQuery, queryCount );
vkCmdResetEvent( commandBuffer, event, stageMask );
vkCmdPipelineBarrier( commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, VkMemoryBarrier* pMemoryBarriers, ... )