Pipeline Vulkan: A case of Gate-ing and Wait-ing

Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

From the Command Buffer Notes:
These are the Commands that can be entered into the Command Buffer, I

vkCmdSetStencilCompareMask( commandBuffer, faceMask, compareMask);
vkCmdWriteTimestamp( commandBuffer, pipelineStage, queryPool, query);
vkCmdUpdateBuffer( commandBuffer, dstBuffer, dstOffset, dataSize, pData);
vkCmdSetStencilWriteMask( commandBuffer, faceMask, writeMask);
vkCmdSetStencilReference( commandBuffer, faceMask, reference);
vkCmdSetLineWidth( commandBuffer, lineWidth);
vkCmdSetEvent( commandBuffer, event, stageMask);
vkCmdSetDiscardRectangleEXT( commandBuffer, firstDiscardRectangle, discardRectangleCount, pDiscardRectangles);
vkCmdSetDeviceMaskKHX( commandBuffer, deviceMask);
vkCmdSetViewportWScalingNV( commandBuffer, firstViewport, viewportCount, pViewportWScalings);
vkCmdReserveSpaceForCommandsNVX( commandBuffer, pReserveSpaceInfo);
vkCmdPushDescriptorSetWithTemplateKHR( commandBuffer, descriptorUpdateTemplate, layout, set, pData);
vkCmdPushDescriptorSetKHR( commandBuffer, pipelineBindPoint, layout, set, descriptorWriteCount, pDescriptorWrites);
vkCmdFillBuffer( commandBuffer, dstBuffer, dstOffset, size, data);
vkCmdSetDepthBias( commandBuffer, depthBiasConstantFactor, depthBiasClamp, depthBiasSlopeFactor);
vkCmdSetBlendConstants( commandBuffer, blendConstants[4]);
vkCmdResetQueryPool( commandBuffer, queryPool, firstQuery, queryCount);
vkCmdResetEvent( commandBuffer, event, stageMask);
vkCmdPushConstants( commandBuffer, layout, stageFlags, offset, size, pValues);
vkCmdProcessCommandsNVX( commandBuffer, pProcessCommandsInfo);
vkCmdPipelineBarrier( commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, VkMemoryBarrier* pMemoryBarriers, bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers);

From the Command Buffer Notes:
These are the Commands that can be entered into the Command Buffer, II

vkCmdEndRenderPass( commandBuffer);
vkCmdEndQuery( commandBuffer, query);
vkCmdDrawIndirectCountAMD( commandBuffer, stride);
vkCmdDrawIndexedIndirectCountAMD( commandBuffer, stride);
vkCmdDrawIndexedIndirect( commandBuffer, stride);
vkCmdDrawIndexed( commandBuffer, indexCount, instanceCount, firstIndex, int32_t vertexOffset, firstInstance);
vkCmdDraw( commandBuffer, vertexCount, instanceCount, firstVertex, firstInstance);
vkCmdDebugMarkerInsertEXT( commandBuffer, pMarkerInfo);
vkCmdCopyImageToBuffer( commandBuffer, pRegions);
vkCmdCopyBuffer( commandBuffer, pRegions);
vkCmdClearDepthStencilImage( commandBuffer, pRanges);
vkCmdClearColorImage( commandBuffer, pRanges);
vkCmdClearAttachments( commandBuffer, attachmentCount, const pRects);
vkCmdBindVertexBuffers( commandBuffer, firstBinding, bindingCount, const pOffsets);
vkCmdBindPipeline( commandBuffer, pipeline);
vkCmdBindIndexBuffer( commandBuffer, indexType);
vkCmdBindDescriptorSets( commandBuffer, pDynamicOffsets);
vkCmdBeginRenderPass( commandBuffer, const contents);
vkCmdBeginQuery( commandBuffer, flags);

Potential Memory Race Conditions that Pipeline Barriers can Prevent

1. Write-then-Read (WtR) – the memory write in one operation starts overwriting the memory that another operation’s read needs to use
2. Read-then-Write (RtW) – the memory read in one operation hasn’t yet finished before another operation starts overwriting that memory
3. Write-then-Write (WtW) – two operations start overwriting the same memory and the end result is non-deterministic

Note: there is no problem with Read-then-Read (RRR) as no data has been changed

We don’t any one of these commands to have to wait on a previous command unless you say so. In general, we want all of these commands to be able to run “tail-out.” But, if we do that, surely, there will be nasty race conditions

vkCmdPipelineBarrier() Function Call

A Pipeline Barrier is a way to establish a memory dependency between commands that were submitted before the barrier and commands that are submitted after the barrier.

vkCmdPipelineBarrier(commandBuffer, srcStageMask, dstStageMask, memoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers, srcStageMask, dstStageMask);

The Scenario

1. The cross-streets are named after pipeline stages
2. All traffic lights start out green (“we want all of these commands to be able to run “tail-out”)
3. There are special sensors at all intersections that will know when the first car in the src group enters that intersection
4. When the first car in the src group enters the intersection, the src traffic light can be turned back to green.
5. The last car in the src group completes its trip through the intersection, the dst traffic light can be turned back to green.
6. Vulkan command pipeline ordering is this: (1) the src cars get released; (2) the pipeline barrier is invoked (which turns some reds red); (3) the dst cars get released (which end up being stopped by a red light somewhere)
**Pipeline Stage Masks**  
Where in the Pipeline is this Memory Data being Generated or Consumed?

- VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT
- VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT
- VK_PIPELINE_STAGE_VERTEX_INPUT_BIT
- VK_PIPELINE_STAGE_VERTEX_SHADER_BIT  
- VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT
- VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT
- VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT
- VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT
- VK_PIPELINE_STAGE-placeholder_SHADER_BIT
- VK_PIPELINE_STAGE-placeholder_SHADER_BIT  
- VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT
- VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT
- VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
- VK_PIPELINE_STAGE_ALL_GRAPHICS_BIT  
- VK_PIPELINE_STAGE_ALL_COMMANDS_BIT
- VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT
- VK_PIPELINE_STAGE_TRANSFER_BIT
- VK_PIPELINE_STAGE_HOST_BIT
- VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT

**Access Operations and what Pipeline Stages they can be used in**

- VK_ACCESS_MEMORY_READ_BIT
- VK_ACCESS_MEMORY_WRITE_BIT
- VK_ACCESS_HOST_READ_BIT
- VK_ACCESS_HOST_WRITE_BIT
- VK_ACCESS_TRANSFER_READ_BIT
- VK_ACCESS_TRANSFER_WRITE_BIT
- VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT
- VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_WRITE_BIT
- VK_ACCESS_COLOR_ATTACHMENT_READ_BIT
- VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT
- VK_ACCESS_SHADER_READ_BIT
- VK_ACCESS_SHADER_WRITE_BIT
- VK_ACCESS_INPUT_ATTACHMENT_READ_BIT
- VK_ACCESS_UNIFORM_READ_BIT
- VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT

**Access Masks**  
What are you Interested in Generating or Consuming this Memory for?

- VK_ACCESS_INDIRECT_COMMAND_READ_BIT
- VK_ACCESS_COMMAND_READ_BIT
- VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT
- VK_ACCESS_UNIFORM_READ_BIT
- VK_ACCESS_INDEX_READ_BIT
- VK_ACCESS_MEMORY_READ_BIT
- VK_ACCESS_MEMORY_WRITE_BIT
- VK_ACCESS_HOST_READ_BIT
- VK_ACCESS_HOST_WRITE_BIT
- VK_ACCESS_TRANSFER_READ_BIT
- VK_ACCESS_TRANSFER_WRITE_BIT
- VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT
- VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_WRITE_BIT
- VK_ACCESS_COLOR_ATTACHMENT_READ_BIT
- VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT
- VK_ACCESS_SHADER_READ_BIT
- VK_ACCESS_SHADER_WRITE_BIT
- VK_ACCESS_INPUT_ATTACHMENT_READ_BIT
- VK_ACCESS_UNIFORM_READ_BIT
- VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT

**Pipeline Stages**

**Example:** Be sure we are done writing an output image before using it for something else.
The Scenario

src cars are generating the image
dst cars are doing something with that image

Example: Don’t read a buffer back to the host until a shader is done writing it

Example: Don’t read a buffer back to the host until a shader is done writing it

Stages

Access types

VkImageLayout – How an Image gets Laid Out in Memory
depends on how it will be Used

Here, the use of vkCmdPipelineBarrier() is to simply change the layout of an image

src
dst

VK_IMAGE_LAYOUT_UNDEFINED
VK_IMAGE_LAYOUT_GENERAL
VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL
VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL
VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL
VK_IMAGE_LAYOUT_TRANSFER_SRC_OPTIMAL
VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL
VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL
VK_IMAGE_LAYOUT_PRESENT_SRC_KHR
VK_IMAGE_LAYOUT_SHARED_PRESENT_KHR

Used as a color attachment
Read into a shader as a texture
Copy from
Copy to
Show image to viewer

vkCmdImageMemoryBarrier()

vkCmdPipelineBarrier()