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

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License
From the Command Buffer Notes:
These are the Commands that can be entered into the Command Buffer, I

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

```c
vkCmdFillBuffer( commandBuffer, dstBuffer, dstOffset, size, data );
vkCmdNextSubpass( commandBuffer, contents );
vkCmdPipelineBarrier( commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, VkMemoryBarrier* pMemoryBarriers,
                      bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers );
vkCmdProcessCommandsNVX( commandBuffer, pProcessCommandsInfo );
vkCmdPushConstants( commandBuffer, layout, stageFlags, offset, size, pValues );
vkCmdPushDescriptorSetKHR( commandBuffer, pipelineBindPoint, layout, set, descriptorWriteCount, pDescriptorWrites );
vkCmdPushDescriptorSetWithTemplateKHR( commandBuffer, descriptorUpdateTemplate, layout, set, pData );
vkCmdResolveImage( commandBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, regionCount, pRegions );
vkCmdSetBlendConstants( commandBuffer, blendConstants[4] );
vkCmdSetDepthBias( commandBuffer, depthBiasConstantFactor, depthBiasClamp, depthBiasSlopeFactor );
vkCmdSetDeviceMaskKHX( commandBuffer, deviceMask );
vkCmdSetDiscardRectangleEXT( commandBuffer, firstDiscardRectangle, discardRectangleCount, pDiscardRectangles );
vkCmdSetEvent( commandBuffer, event, stageMask );
vkCmdSetLineWidth( commandBuffer, lineWidth );
vkCmdSetScissor( commandBuffer, firstScissor, scissorCount, pScissors );
vkCmdSetStencilCompareMask( commandBuffer, faceMask, compareMask );
vkCmdSetStencilReference( commandBuffer, faceMask, reference );
vkCmdSetStencilWriteMask( commandBuffer, faceMask, writeMask );
vkCmdSetViewport( commandBuffer, firstViewport, viewportCount, pViewports );
vkCmdSetViewportWScalingNV( commandBuffer, firstViewport, viewportCount, pViewportWScalings );
vkCmdUpdateBuffer( commandBuffer, dstBuffer, dstOffset, dataSize, pData );
vkCmdWaitEvents( commandBuffer, eventCount, pEvents, srcStageMask, dstStageMask, memoryBarrierCount, pMemoryBarriers,
                 bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierCount, pImageMemoryBarriers );
vkCmdWriteTimestamp( commandBuffer, pipelineStage, queryPool, query );
```
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 (RtR) as no data has been changed
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. This allows the pipeline stage to be used once the previous stage is completely done, ensuring data integrity and correct execution order of commands.

The `vkCmdPipelineBarrier()` function call is used to create a pipeline barrier. It takes the following arguments:

- `commandBuffer`: The command buffer to perform the barrier on.
- `srcStageMask`: A bitmask specifying the source stages that the barrier is effective on.
- `dstStageMask`: A bitmask specifying the destination stages that the barrier is effective on.
- `VK_DEPENDENCY_BY_REGION_BIT`: A constant to indicate the dependency type.
- `memoryBarrierCount`, `pMemoryBarriers`: The number of memory barriers and an array of memory barriers.
- `bufferMemoryBarrierCount`, `pBufferMemoryBarriers`: The number of buffer memory barriers and an array of buffer memory barriers.
- `imageMemoryBarrierCount`, `pImageMemoryBarriers`: The number of image memory barriers and an array of image memory barriers.

This function call ensures that the specified stages of the pipeline are synchronized, allowing for correct data processing and rendering.
The Scenario
1. The cross-streets are named after pipeline stages

2. All traffic lights start out green

3. There are special sensors at all intersections that will know when any car in the src group is in that intersection

4. There are connections from those sensors to the traffic lights so that when any car in the src group is in the intersection, the proper dst traffic light will be turned red

5. When the last car in the src group completely makes it through its intersection, the proper dst traffic light is turned back to green

6. The Vulkan command pipeline ordering is this: (1) the src cars get released, (2) the pipeline barrier is invoked (which turns some light red), (3) the dst cars stop at the red light, (4) the src intersection clears, (5) all lights are now green, (6) the dst cars continue.
### Pipeline Stage Masks – Where in the Pipeline is this Memory Data being Generated or Consumed?

<table>
<thead>
<tr>
<th>VK PIPELINE STAGE TOP OF PIPE BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK PIPELINE STAGE DRAW INDIRECT BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE VERTEX INPUT BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE VERTEX SHADER BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE TESSELLATION CONTROL SHADER BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE TESSELLATION EVALUATION SHADER BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE GEOMETRY SHADER BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE FRAGMENT SHADER BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE EARLY_FRAGMENT_TESTS_BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE LATE_FRAGMENT_TESTS_BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE COLOR_ATTACHMENT_OUTPUT_BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE COMPUTE SHADER BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE TRANSFER_BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE BOTTOM OF PIPE BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE HOST BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE ALL_GRAPHICS_BIT</td>
</tr>
<tr>
<td>VK PIPELINE STAGE ALL_COMMANDS_BIT</td>
</tr>
</tbody>
</table>
Access Masks –
What are you Interested in Generating or Consuming this Memory for?

VK_ACCESS_INDIRECT_COMMAND_READ_BIT
VK_ACCESS_INDEX_READ_BIT
VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT
VK_ACCESS_UNIFORM_READ_BIT
VK_ACCESS_INPUT_ATTACHMENT_READ_BIT
VK_ACCESS_SHADER_READ_BIT
VK_ACCESS_SHADER_WRITE_BIT
VK_ACCESS_COLOR_ATTACHMENT_READ_BIT
VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT
VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT
VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_WRITE_BIT
VK_ACCESS_TRANSFER_READ_BIT
VK_ACCESS_TRANSFER_WRITE_BIT
VK_ACCESS_HOST_READ_BIT
VK_ACCESS_HOST_WRITE_BIT
VK_ACCESS_MEMORY_READ_BIT
VK_ACCESS_MEMORY_WRITE_BIT
## Pipeline Stages and what Access Operations are Allowed

<table>
<thead>
<tr>
<th>Pipeline Stage</th>
<th>VK_ACCESS_INDIRECT_COMMAND_READ_BIT</th>
<th>VK_ACCESS_INDEX_READ_BIT</th>
<th>VK_ACCESS_VERTEX_ATTRIB_READ_BIT</th>
<th>VK_ACCESS_UNIFORM_READ_BIT</th>
<th>VK_ACCESS_INPUT_ATTACHMENT_READ_BIT</th>
<th>VK_ACCESS_SHADER_READ_BIT</th>
<th>VK_ACCESS_SHADER_WRITE_BIT</th>
<th>VK_ACCESS_COLOR_ATTACHMENT_READ_BIT</th>
<th>VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT</th>
<th>VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT</th>
<th>VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_WRITE_BIT</th>
<th>VK_ACCESS_TRANSFER_READ_BIT</th>
<th>VK_ACCESS_TRANSFER_WRITE_BIT</th>
<th>VK_ACCESS_HOST_READ_BIT</th>
<th>VK_ACCESS_HOST_WRITE_BIT</th>
<th>VK_ACCESS_MEMORY_READ_BIT</th>
<th>VK_ACCESS_MEMORY_WRITE_BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_VERTEX_INPUT_BIT</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_VERTEX_SHADER_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_COMPUTE_SHADER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TRANSFER_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_HOST_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Pipeline Stages
- VK_PIPELINE_STAGE_TOP_OF_PIPE
- VK_PIPELINE_STAGE_DRAW_INDIRECT
- VK_PIPELINE_STAGE_VERTEX_INPUT
- VK_PIPELINE_STAGE_VERTEX_SHADER
- VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER
- VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER
- VK_PIPELINE_STAGE_GEOMETRY_SHADER
- VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS
- VK_PIPELINE_STAGE_FRAGMENT_SHADER
- VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS
- VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT
- VK_PIPELINE_STAGE_BOTTOM_OF_PIPE
- VK_PIPELINE_STAGE_COMPUTE_SHADER
- VK_PIPELINE_STAGE_TRANSFER
- VK_PIPELINE_STAGE_HOST
Access Operations and what Pipeline Stages they can be used In

<table>
<thead>
<tr>
<th>Access Operations</th>
<th>Stage Bit 1</th>
<th>Stage Bit 2</th>
<th>Stage Bit 3</th>
<th>Stage Bit 4</th>
<th>Stage Bit 5</th>
<th>Stage Bit 6</th>
<th>Stage Bit 7</th>
<th>Stage Bit 8</th>
<th>Stage Bit 9</th>
<th>Stage Bit 10</th>
<th>Stage Bit 11</th>
<th>Stage Bit 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_ACCESS_INDIRECT_COMMAND_READ_BIT</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_INDEX_READ_BIT</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_UNIFORM_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td>● ● ● ● ●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_INPUT_ATTACHMENT_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_SHADER_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_SHADER_WRITE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td>● ● ● ● ●</td>
<td>● ● ● ● ●</td>
<td>● ● ● ● ●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_COLOR_ATTACHMENT_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_TRANSFER_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_TRANSFER_WRITE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_HOST_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_HOST_WRITE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
<td></td>
</tr>
<tr>
<td>VK_ACCESS_MEMORY_READ_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>VK_ACCESS_MEMORY_WRITE_BIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>● ● ● ● ●</td>
</tr>
</tbody>
</table>
Example: Be sure we are done writing an output image before using it for something else.

### Stages

- 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_EARLY_FRAGMENT_TESTS_BIT
- VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT
- VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
- VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT
- VK_PIPELINE_STAGE_TRANSFER_BIT
- VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT
- VK_PIPELINE_STAGE_HOST_BIT
- VK_PIPELINE_STAGE_ALL_GRAPHICS_BIT
- VK_PIPELINE_STAGE_ALL_COMMANDS_BIT

### Access types

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

### Src

- src

### Dst

- dst

Dst (no access setting needed)
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

```
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_EARLY_FRAGMENT_TESTS_BIT
VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT
VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT
VK_PIPELINE_STAGE_TRANSFER_BIT
VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT
VK_PIPELINE_STAGE_HOST_BIT
VK_PIPELINE_STAGE_ALL_GRAPHICS_BIT
VK_PIPELINE_STAGE_ALL_COMMANDS_BIT
```

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

Stages

Access types
The Scenario

src cars

TOP_OF_PIPE Street

VERTEX_INPUT Street

VERTEX_SHADER Street

BOTTOM_OF_PIPE Street

dst cars

COLOR_ATTACHMENT_OUTPUT Street

TRANSFER_BIT Street

FRAGMENT_SHADER Street

mjb – January 8, 2020

Oregon State University
Computer Graphics
VkImageLayout – How an Image gets Laid Out in Memory depends on how it will be Used

```c
VkImageMemoryBarrier vimb
    vimb.sType = VK_STRUCTURE_TYPE_IMAGE_MEMORY_BARRIER;
    vimb.pNext = nullptr;
    vimb.srcAccessMask = ???;
    vimb.dstAccessMask = ???;
    vimb.oldLayout = ???;
    vimb.newLayout = ???;
    vimb.srcQueueFamilyIndex = VK_QUEUE_FAMILY_IGNORED;
    vimb.dstQueueFamilyIndex = VK_QUEUE_FAMILY_IGNORED;
    vimb.image = ???;
    vimb.subresourceRange = visr;
```

- VK_IMAGE_LAYOUT_UNDEFINED
- VK_IMAGE_LAYOUT_GENERAL
- VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL
- VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL
- VK_IMAGE_LAYOUT_DEPTH_STENCIL_READ_ONLY_OPTIMAL
- VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL
- VK_IMAGE_LAYOUT_TRANSFER_SRC_OPTIMAL
- VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL
- VK_IMAGE_LAYOUT_PREINITIALIZED
- VK_IMAGE_LAYOUT_PRESENT_SRC_KHR
- VK_IMAGE_LAYOUT_SHARED_PRESENT_KHR

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