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

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 );
vkCmdSetDepthBounds( commandBuffer, minDepthBounds, maxDepthBounds );
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 );
vkCmdSetViewport( commandBuffer, firstViewport, viewportCount, pViewports );
vkCmdSetViewportWScalingNV( commandBuffer, firstViewport, viewportCount, pViewportWScalings );
vkCmdWriteTimestamp( commandBuffer, pipelineStage, queryPool, query );

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 “flat-out”.

But, if we do that, surely there will be nasty race conditions!
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.

vkCmdPipelineBarrier( ) Function Call

- `srcStageMask`: Guarantee that this pipeline stage has completely generated one set of data before ...
- `dstStageMask`: ... allowing this pipeline stage to consume it
- `VK_DEPENDENCY_BY_REGION_BIT`,

memoryBarrierCount, pMemoryBarriers,

bufferMemoryBarrierCount, pBufferMemoryBarriers,

imageMemoryBarrierCount, pImageMemoryBarriers

Defines what data we will be blocking/un-blocking on
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 flat-out")
3. There are special sensors at all intersections that will know when the **first car in the src group** enters that intersection
4. There are connections from those sensors to the traffic lights so that when the **first car in the src group** enters its intersection, the **dst** traffic light will be turned red
5. When the **last car in the src group** completely makes it through its intersection, the **dst** traffic light can be 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 lights 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_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
Pipeline 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

Vertex Shader
Primitive Assembly
Tessellation Control Shader
Tessellation Primitive Generator
Tessellation Evaluation Shader
Primitive Assembly
Geometry Shader
Primitive Assembly
Rasterizer
Fragment Shader
Access Masks –
What are you Interested in Generating or Consuming this Memory for?

<table>
<thead>
<tr>
<th>VK_ACCESS_INDIRECT_COMMAND_READ_BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_ACCESS_INDEX_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_UNIFORM_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_INPUT_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_SHADER_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_SHADER_WRITE_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_COLOR_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_WRITE_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_TRANSFER_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_TRANSFER_WRITE_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_HOST_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_HOST_WRITE_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_MEMORY_READ_BIT</td>
</tr>
<tr>
<td>VK_ACCESS_MEMORY_WRITE_BIT</td>
</tr>
</tbody>
</table>
# Pipeline Stages and what Access Operations can Happen There

<table>
<thead>
<tr>
<th>Pipeline Stage</th>
<th>VK Access Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT</td>
<td>VK_ACCESS_INDIRECT_COMMAND_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT</td>
<td>VK_ACCESS_INDEX_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_VERTEX_INPUT_BIT</td>
<td>VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_VERTEX_SHADER_BIT</td>
<td>VK_ACCESS_UNIFORM_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT</td>
<td>VK_ACCESS_INPUT_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT</td>
<td>VK_ACCESS_SHADER_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT</td>
<td>VK_ACCESS_SHADER_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT</td>
<td>VK_ACCESS_COLOR_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT</td>
<td>VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT</td>
<td>VK_ACCESS_TRANSFER_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_COMPUTE_SHADER</td>
<td>VK_ACCESS_TRANSFER_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TRANSFER_BIT</td>
<td>VK_ACCESS_HOST_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_HOST_BIT</td>
<td>VK_ACCESS_HOST_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_MEMORY_READ_BIT</td>
<td>VK_ACCESS_MEMORY_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_MEMORY_WRITE_BIT</td>
<td>VK_ACCESS_MEMORY_WRITE_BIT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pipeline Stage</th>
<th>Access Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT</td>
<td>VK_ACCESS_INDEX_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_VERTEX_INPUT_BIT</td>
<td>VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_VERTEX_SHADER_BIT</td>
<td>VK_ACCESS_UNIFORM_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT</td>
<td>VK_ACCESS_INPUT_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT</td>
<td>VK_ACCESS_SHADER_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT</td>
<td>VK_ACCESS_SHADER_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT</td>
<td>VK_ACCESS_COLOR_ATTACHMENT_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT</td>
<td>VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT</td>
<td>VK_ACCESS_TRANSFER_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_COMPUTE_SHADER</td>
<td>VK_ACCESS_TRANSFER_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_HOST_BIT</td>
<td>VK_ACCESS_HOST_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_MEMORY_READ_BIT</td>
<td>VK_ACCESS_HOST_WRITE_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_MEMORY_WRITE_BIT</td>
<td>VK_ACCESS_MEMORY_READ_BIT</td>
</tr>
<tr>
<td>VK_PIPELINE_STAGE_MEMORY_WRITE_BIT</td>
<td>VK_ACCESS_MEMORY_WRITE_BIT</td>
</tr>
</tbody>
</table>

1. VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT
2. VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT
3. VK_PIPELINE_STAGE_VERTEX_INPUT_BIT
4. VK_PIPELINE_STAGE_VERTEX_SHADER_BIT
5. VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT
6. VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT
7. VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT
8. VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT
9. VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT
10. VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT
11. VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
12. VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT
13. VK_PIPELINE_STAGE_COMPUTE_SHADER
14. VK_PIPELINE_STAGE_TRANSFER_BIT
15. VK_PIPELINE_STAGE_HOST_BIT
16. VK_PIPELINE_STAGE_MEMORY_READ_BIT
17. VK_PIPELINE_STAGE_MEMORY_WRITE_BIT
### Access Operations and what Pipeline Stages they can be used In

<table>
<thead>
<tr>
<th>Access Operation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>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_DEPTH_STENCIL_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_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

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_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
dst

Stages

Access types

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

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

dst

(dst (no access setting needed))
The Scenario

src cars

TOP_OF_PIPE Street

VERTEX_INPUT Street

VERTEX_SHADER Street

BOTTOM_OF_PIPE Street

COLOR_ATTACHMENT_OUTPUT Street

TRANSFER_BIT Street

dst cars

FRAGMENT_SHADER Street

src cars

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

<table>
<thead>
<tr>
<th>VkImageLayout</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VK_IMAGE_LAYOUT_UNDEFINED</td>
<td>Used as a color attachment</td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_GENERAL</td>
<td>Read into a shader as a texture</td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL</td>
<td>Copy from</td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL</td>
<td>Copy to</td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_DEPTH_STENCIL_READ_ONLY_OPTIMAL</td>
<td>Show image to viewer</td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL</td>
<td></td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_TRANSFER_SRC_OPTIMAL</td>
<td></td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL</td>
<td></td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_PREINITIALIZED</td>
<td></td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_PRESENT_SRC_KHR</td>
<td></td>
</tr>
<tr>
<td>VK_IMAGE_LAYOUT_SHARED_PRESENT_KHR</td>
<td></td>
</tr>
</tbody>
</table>

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