




1




Pipeline Barriers



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



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



Oregon State University
Computer Graphics

PipelineBarriers.pptx mjb – February 26, 2023


1

2

Why Do We Need Pipeline Barriers?

A series of `vkCmdxxx()` calls are meant to run “flat-out”, that is, as fast as the Vulkan runtime can get them executing. But, many times, that is not desirable because the output of one command might be needed as the input to a subsequent command.

Pipeline Barriers solve this problem by declaring which stages of the hardware pipeline in subsequent `vkCmdyyy()` calls need to wait until which stages in previous `vkCmdxxx()` calls are completed.



Oregon State University
Computer Graphics

mjb – February 26, 2023

2

Potential Memory Race Conditions that Pipeline Barriers can Prevent

3

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

Note: there is no problem with Read-after-Read (R-a-R) as no data gets changed.



mjb – February 26, 2023

3

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

4

```
vkCmdBeginConditionalRendering
vkCmdBeginDebugUtilsLabel
vkCmdBeginQuery
vkCmdBeginQueryIndexed
vkCmdBeginRendering
vkCmdBeginRenderPass
vkCmdBeginRenderPass2
vkCmdBeginTransformFeedback
vkCmdBindDescriptorSets
vkCmdBindIndexBuffer
vkCmdBindInvocationMask
vkCmdBindPipeline
vkCmdBindPipelineShaderGroup
vkCmdBindShadingRateImage
vkCmdBindTransformFeedbackBuffers
vkCmdBindVertexBuffers
vkCmdBindVertexBuffers2
vkCmdBlitImage
```

```
vkCmdBlitImage2
vkCmdBuildAccelerationStructure
vkCmdBuildAccelerationStructuresIndirect
vkCmdBuildAccelerationStructures
vkCmdClearAttachments
vkCmdClearColorImage
vkCmdClearDepthStencilImage
vkCmdCopyAccelerationStructure
vkCmdCopyAccelerationStructureToMemory
vkCmdCopyBuffer
vkCmdCopyBuffer2
vkCmdCopyBufferToImage
vkCmdCopyBufferToImage2
vkCmdCopyImage
vkCmdCopyImage2
vkCmdCopyImageToBuffer
vkCmdCopyImageToBuffer2
vkCmdCopyMemoryToAccelerationStructure
```



mjb – February 26, 2023

4

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

5

```

vkCmdCopyQueryPoolResults
vkCmdCuLaunchKernelX
vkCmdDebugMarkerBegin
vkCmdDebugMarkerEnd
vkCmdDebugMarkerInsert
vkCmdDispatch
vkCmdDispatchBase
vkCmdDispatchIndirect
vkCmdDraw
vkCmdDrawIndexed
vkCmdDrawIndexedIndirect
vkCmdDrawIndexedIndirectCount
vkCmdDrawIndirect
vkCmdDrawIndirectByteCount
vkCmdDrawIndirectCount
vkCmdDrawMeshTasksIndirectCount
vkCmdDrawMeshTasksIndirect
vkCmdDrawMeshTasks

```

```

vkCmdDrawMulti
vkCmdDrawMultiIndexed
vkCmdEndConditionalRendering
vkCmdEndDebugUtilsLabel
vkCmdEndQuery
vkCmdEndQueryIndexed
vkCmdEndRendering
vkCmdEndRenderPass
vkCmdEndRenderPass2
vkCmdEndTransformFeedback
vkCmdExecuteCommands
vkCmdExecuteGeneratedCommands
vkCmdFillBuffer
vkCmdInsertDebugUtilsLabel
vkCmdNextSubpass
vkCmdNextSubpass2
vkCmdPipelineBarrier
vkCmdPipelineBarrier2

```



mjb - February 26, 2023

5

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

6

```

vkCmdPreprocessGeneratedCommands
vkCmdPushConstants
vkCmdPushDescriptorSet
vkCmdPushDescriptorSetWithTemplate
vkCmdResetEvent
vkCmdResetEvent2
vkCmdResetQueryPool
vkCmdResolveImage
vkCmdResolveImage2
vkCmdSetBlendConstants
vkCmdSetCheckpoint
vkCmdSetCoarseSampleOrder
vkCmdSetCullMode
vkCmdSetDepthBias
vkCmdSetDepthBiasEnable
vkCmdSetDepthBounds
vkCmdSetDepthBoundsTestEnable
vkCmdSetDepthCompareOp

```

```

vkCmdSetDepthTestEnable
vkCmdSetDepthWriteEnable
vkCmdSetDeviceMask
vkCmdSetDiscardRectangle
vkCmdSetEvent
vkCmdSetEvent2
vkCmdSetExclusiveScissor
vkCmdSetFragmentShadingRateEnum
vkCmdSetFragmentShadingRate
vkCmdSetFrontFace
vkCmdSetLineStipple
vkCmdSetLineWidth
vkCmdSetLogicOp
vkCmdSetPatchControlPoints
vkCmdSetPrimitiveRestartEnable
vkCmdSetPrimitiveTopology
vkCmdSetRasterizerDiscardEnable
vkCmdSetRayTracingPipelineStackSize

```



mjb - February 26, 2023

6

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

7

```

vkCmdSetSampleLocations
vkCmdSetScissor
vkCmdSetScissorWithCount
vkCmdSetStencilCompareMask
vkCmdSetStencilOp
vkCmdSetStencilReference
vkCmdSetStencilTestEnable
vkCmdSetStencilWriteMask
vkCmdSetVertexInput
vkCmdSetViewport
vkCmdSetViewportShadingRatePalette
vkCmdSetViewportWithCount
vkCmdSetViewportWScaling

```

```

vkCmdSubpassShading
vkCmdTraceRaysIndirect2
vkCmdTraceRaysIndirect
vkCmdTraceRays
vkCmdUpdateBuffer
vkCmdWaitEvents
vkCmdWaitEvents2
vkCmdWriteAccelerationStructuresProperties
vkCmdWriteBufferMarker2
vkCmdWriteBufferMarker
vkCmdWriteTimestamp
vkCmdWriteTimestamp2

```



mjb - February 26, 2023

7

vkCmdPipelineBarrier() Function Call

8

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

```

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

```

Guarantee that *this* pipeline stage is completely done being used by the previous vkCmdxxx before ...

... allowing *this* pipeline stage to be used by the next vkCmdyyy

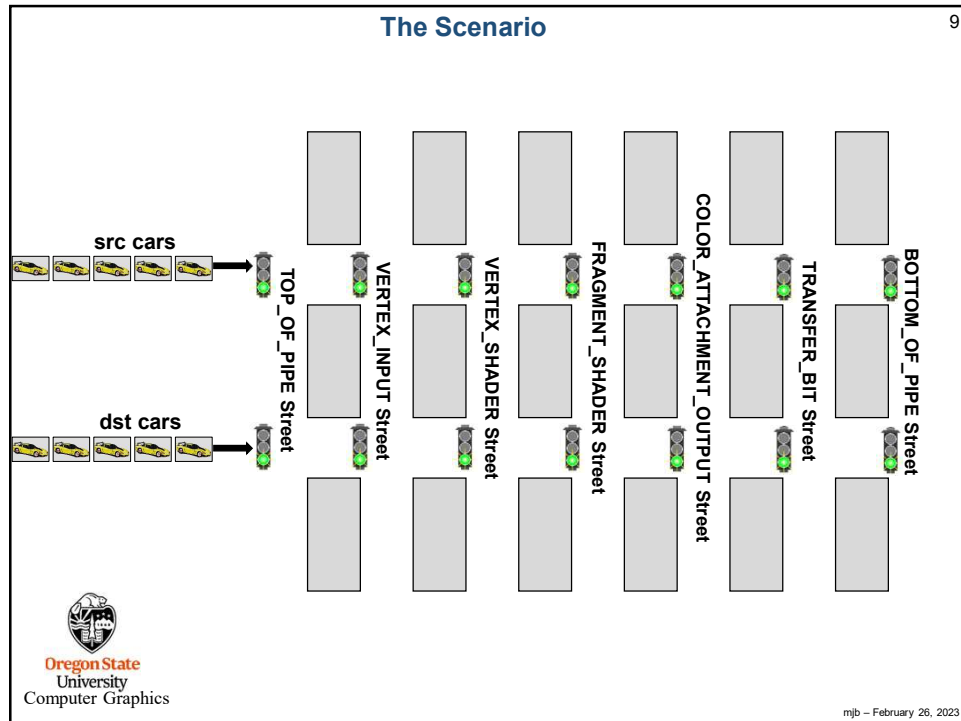
Defines what data we will be blocking on or un-blocking on



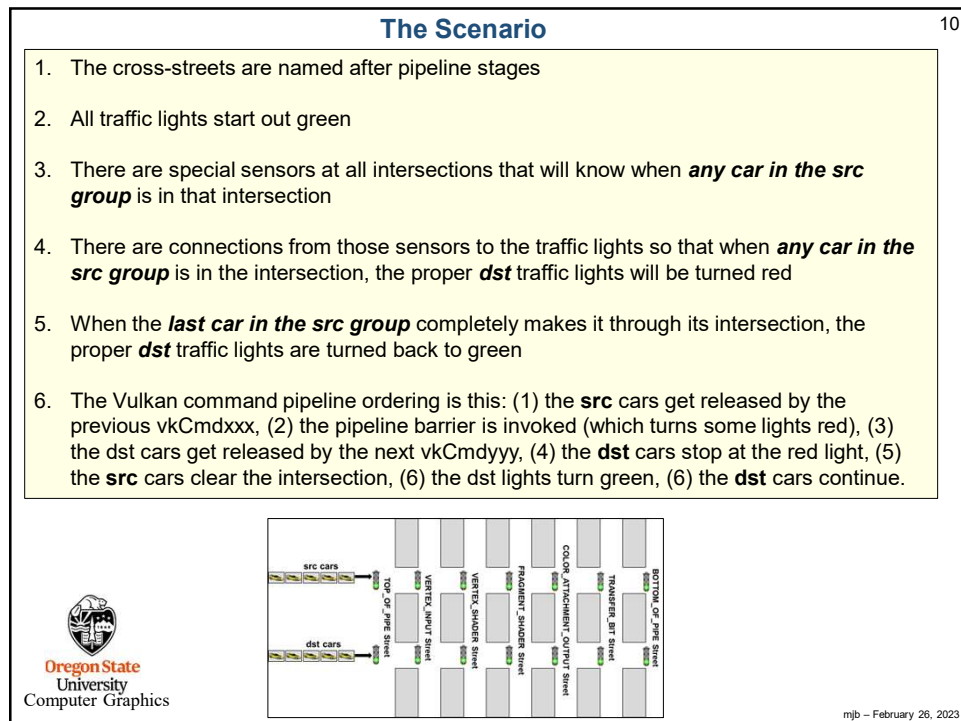
The hope is maximize the number of unblocked stages:
produce data *early* and consume data *late*

mjb - February 26, 2023

8



9



10

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

mjb – February 26, 2023

11

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

mjb – February 26, 2023

12

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




mjb – February 26, 2023

13

Pipeline Stages and what Access Operations are Allowed

	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
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																	
VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT				•		•	•										
VK_PIPELINE_STAGE_TRANSFER_BIT												•	•				
VK_PIPELINE_STAGE_HOST_BIT														•	•		




mjb – February 26, 2023

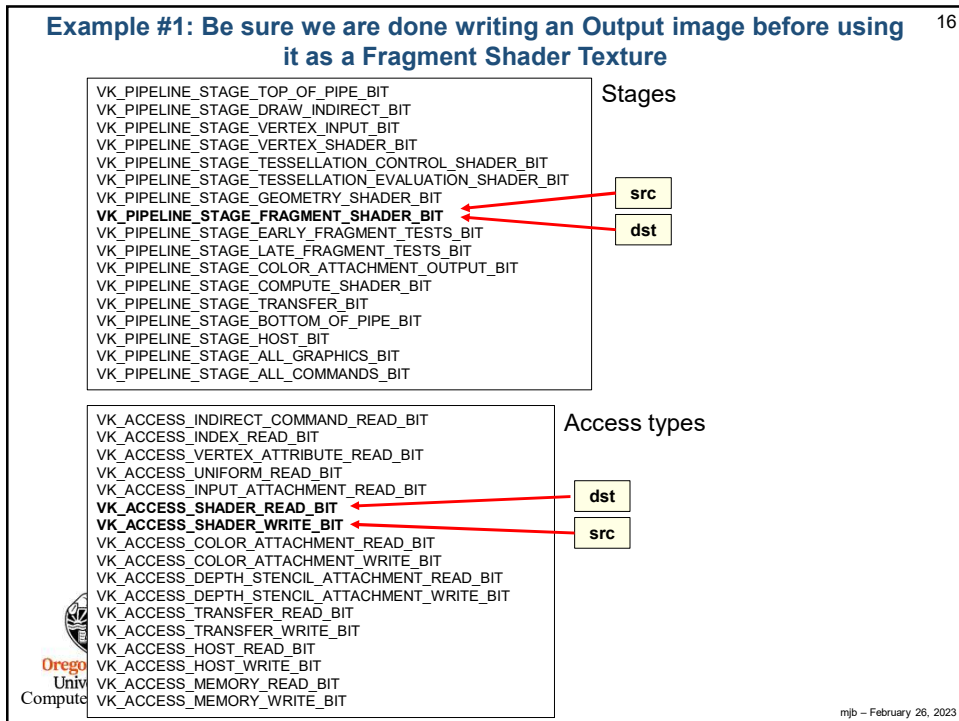
14

Access Operations and what Pipeline Stages they can be used In

	1	2	3	4	5	6	7	8	9	10	11	12
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_EARLY_FRAGMENT_TESTS_BIT												
VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT												
VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT												
VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT												
VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT												
VK_PIPELINE_STAGE_TRANSFER_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												

 mjb - February 26, 2023

15



16

Example #2: Setting a Pipeline Barrier so the Compute Shader Waits for the Drawing to Finish 19

```

VkBufferMemoryBarrier          vbmb;
    vbmb.sType = VK_STRUCTURE_TYPE_BUFFER_MEMORY_BARRIER;
    vbmb.pNext = nullptr;
    vbmb.srcAccessFlags = VK_ACCESS_SHADER_WRITE_BIT;
    vbmb.dstAccessFlags = VK_ACCESS_SHADER_READ_BIT;
    vbmb.srcQueueFamilyIndex = 0;
    vbmb.dstQueueFamilyIndex = 0;
    vbmb.buffer =
    vbmb.offset = 0;
    vbmb.size = NUM_PARTICLES * sizeof( glm::vec4 );

const uint32 bufferMemoryBarrierCount = 1;

vkCmdPipelineBarrier
(
    commandBuffer,
    VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT,
    VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT,
    VK_DEPENDENCY_BY_REGION_BIT,
    0, nullptr, bufferMemoryBarrierCount, IN &vbmb, 0, nullptr
);

```