Dynamic State Variables
Creating a Pipeline with Dynamically Changeable State Variables

The graphics pipeline data structure is full of state information, and, as previously-discussed, is largely immutable, that is, the information contained inside it is fixed, and can only be changed by creating a new graphics pipeline data structure with new information.

That isn’t quite true. To a certain extent, Vulkan allows you to declare parts of the pipeline state changeable. This allows you to alter pipeline state information on the fly.

This is useful for managing state information that needs to change frequently. This also creates possible optimization opportunities for the Vulkan driver.
Creating a Pipeline

- VkGraphicsPipelineCreateInfo
  - Shader stages
    - VkSpecializationInfo
    - VkShaderModule
  - VkPipelineShaderStageCreateInfo
    - which stage (VERTEX, etc.)
  - VkPipelineVertexInputStateCreateInfo
    - VkVertexInputBindingDescription
      - binding
      - stride
      - inputRate
    - VkVertexInputAttributeDescription
      - location
      - binding
      - format
      - offset
  - VkPipelineInputAssemblyStateCreateInfo
    - VkPipelineInputAssemblyStateCreateInfo
      - Topology
      - cullMode
      - polygonMode
      - frontFace
      - lineWidth
  - VkPipelineRasterizationStateCreateInfo
    - VkPipelineRasterizationStateCreateInfo
      - depthTestEnable
      - depthWriteEnable
      - depthCompareOp
      - stencilTestEnable
      - stencilOpStateFront
      - stencilOpStateBack
      - blendEnable
      - srcColorBlendFactor
      - dstColorBlendFactor
      - colorBlendOp
      - srcAlphaBlendFactor
      - dstAlphaBlendFactor
      - alphaBlendOp
      - colorWriteMask
      - viewport
      - scissor
      - offset
      - extent
  - VkPipelineDepthStencilStateCreateInfo
    - VkPipelineColorBlendStateCreateInfo
    - VkPipelineColorBlendAttachmentState
    - VkPipelineDynamicStateCreateInfo
      - Array naming the states that can be set dynamically
  - VkPipelineDynamicStateCreateInfo
    - basePipelineHandle
    - basePipelineIndex
  - VkPipelineCreateInfo
    - VkPipelineLayout
  - RenderPass
    - basePipelineHandle
    - basePipelineIndex

vkCreateGraphicsPipeline()
The possible dynamic variables are shown in the `VkDynamicState` enum:

```c
VK_DYNAMIC_STATE_VIEWPORT
VK_DYNAMIC_STATE_SCISSOR
VK_DYNAMIC_STATE_LINE_WIDTH
VK_DYNAMIC_STATE_DEPTH_BIAS
VK_DYNAMIC_STATE_BLEND_CONSTANTS
VK_DYNAMIC_STATE_DEPTH_BOUNDS
VK_DYNAMIC_STATE_STENCIL_COMPARE_MASK
VK_DYNAMIC_STATE_STENCIL_WRITE_MASK
VK_DYNAMIC_STATE_STENCIL_REFERENCE
```
Creating a Pipeline

```c
VkDynamicState
{
    VK_DYNAMIC_STATE_VIEWPORT,
    VK_DYNAMIC_STATE_LINE_WIDTH
};

VkPipelineDynamicStateCreateInfo vpdsci;
    vpdsci.sType = VK_STRUCTURE_TYPE_PIPELINE_DYNAMIC_STATE_CREATE_INFO;
    vpdsci.pNext = nullptr;
    vpdsci.flags = 0;
    vpdsci.dynamicStateCount = sizeof(vds) / sizeof(VkDynamicState); // i.e., 2
    vpdsci.pDynamicStates = &vds;

VkGraphicsPipelineCreateInfo vgpci;
    . . .
    vgpci.pDynamicState = &vpdsci;
    . . .

vkCreateGraphicsPipelines( LogicalDevice, pipelineCache, 1, &vgpci, PALLOCATOR, &GraphicsPipeline );
```

If you declare certain state variables to be dynamic like this, then you **must** fill them in the command buffer! Otherwise, they are **undefined**.
Filling the Dynamic State Variables in the Command Buffer

First call:

```c
vkCmdBindPipeline( … );
```

Then, the command buffer-bound function calls to set these dynamic states are:

```c
vkCmdSetViewport( commandBuffer, firstViewport, viewportCount, pViewports );
vkCmdSetScissor( commandBuffer, firstScissor, scissorCount, pScissors );
vkCmdSetLineWidth( commandBuffer, linewidth );
vkCmdSetDepthBias( commandBuffer, depthBiasConstantFactor, depthBiasClamp, depthBiasSlopeFactor );
vkCmdSetBlendConstants( commandBuffer, blendConstants[4] );
vkCmdSetDepthBounds( commandBuffer, minDepthBounds, maxDepthBounds );
vkCmdSetStencilCompareMask( commandBuffer, faceMask, compareMask );
vkCmdSetStencilWriteMask( commandBuffer, faceMask, writeMask );
vkCmdSetStencilReference( commandBuffer, faceMask, reference );
```