The Graphics Pipeline

Vulkan: A Pipeline Records the Following Items:

- PipelineLayout: DescriptorSets, PushConstants
- Which Shaders are going to be used
- Per-vertex input attributes: location, binding, format, offset
- Per-vertex input bindings: binding, stride, inputRate
- Assembly topology
- Viewport: x, y, w, h, nearDepth, farDepth
- Scissoring: x, y, w, h
- Rasterization: cullMode, polygonMode, frontFace, lineWidth
- Depth: depthTestEnable, depthWriteEnable, depthCompareOp
- Stencil: stencilTestEnable, stencilOpStateFront, stencilOpStateBack
- Blending: blendEnable, srcColorBlendFactor, dstColorBlendFactor, colorBlendOp
- DynamicState: which states can be set dynamically (bound to the command buffer, outside the Pipeline)

**Bold/Italics** indicates that this state item can also be set with Dynamic Variables

Creating a Graphics Pipeline from a lot of Pieces
Creating a Typical Graphics Pipeline

Link in the Per-Vertex Attributes

Options for vpiasci.topology

What is “Primitive Restart Enable”?
One Really Good use of Restart Enable is in Drawing Terrain Surfaces with Triangle Strips

Triangle Strip #0:

Triangle Strip #1:

Triangle Strip #2:

...
The controls blending between the output of the fragment shader and the input to the color attachments.
Group all of the individual state information and create the pipeline

Putting it all Together! (finally…)

Later on, we will Bind the Graphics Pipeline to the Command Buffer when Drawing

vkCmdBindPipeline(CommandBuffers[nextImageIndex], VK_PIPELINE_BIND_POINT_GRAPHICS, GraphicsPipeline);