Textures

Texture Loading, Samplers, and use with Descriptor Sets

Textures in Vulkan

- Textures in Vulkan are in two parts:
  - **The Image:** This is the image data itself in memory.
  - **The Sampler:** This is an object that accesses the image using a set of pre-defined methods. These methods cover such concepts as picking a point between two texels or beyond the edge of the image.
- Access these parts on Pipeline through Descriptor Sets
- Can do Image and Sampler as separate descriptors, but there is a descriptor type called “Combined Image Sampler” that combines the two concepts.

Loading in Image Data

- Need to load in image data before we can use it!
- To do this, we will use a lightweight library called stb_image
- stb_image is a single file so can be included straight into project directory
- Use stb_image to load in an image as raw data…
- … then copy this data to a buffer!
- Buffer and Device Memory created same way as always.
  - Create Buffer + Memory
  - Use memcpy to map image data to memory
  - Use Staging Buffers because Texture data won’t be changing, so can be stored on GPU without Host access. However, there is an issue...

Image Transition

- VkImage types are currently created with layout: VK_IMAGE_LAYOUT_UNDEFINED
- However to transfer an image from a Staging Buffer to a destination, they need to be in the form VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL
  - And after the transfer, it needs to be in the form VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL so that it can be used!
- Recall in Render Pass that attachments are transitioned automatically by subpass dependencies.
- … but textures aren't attachments, and so are not transitioned by the Render Pass!
- To do transitions manually, we need to use "Pipeline Barriers"

Pipeline Barrier

- Pipeline Barrier is similar to concept of Semaphore because it's used to synchronise GPU actions, but we can set it wherever we want.
- Quite literally a "barrier" in the pipeline to organise events and ensure no overlaps.
- Issued as a vkCmd to a queue and will work relative to commands issued before it and after it.
- Main advantage: Pipeline Barriers have additional functionality of forcing layout transitions!
- Image Memory Barrier restricts order of access to image resource… and also allows image layout transitions in that timespan.
- Simply state two points in time:
  - The time a transition can only occur after
  - The time a transition must occur before

The Sampler

- Now that we have our image data set up in a buffer, we need to setup the Sampler that will interact with that data.
- Fairly simple, just using VkSamplerCreateInfo type
  - Defines settings such as filtering (stretching and shrinking of texture on screen), addressing (handling values beyond the 0-1 texel range), and anisotropy.
- Worth noting that anisotropy requires a feature to be enabled on the Device.

The Descriptor Set

- Descriptor Set is created much in the same way as before…
- However Descriptor Set Layout will be of type VK_DESCRIPTOR_TYPE_COMBINED_IMAGE_SAMPLER and the stage it will connect to will be the Fragment Shader
- Descriptor Pool Size will also need to be of Combined Image Sampler type.
- Descriptor Write will instead use the "pImageInfo" value, as opposed to "pBufferInfo".
- pimageInfo is of type VkDescriptorImageInfo, which references both the image buffer, and the sampler being used

Usage of Descriptor Set

- Usage is also the same!
- Ensure vertex data is updated to hold texel values, and VkVertexInputAttributeDescription is updated to describe texel data.
- Bind Descriptor Set with combined image sampler…
- … and receive data in shader ready to use!
- Texture refered to using “sampler2D” type:

  layout(binding = 0) uniform sampler2D textureSampler;
- Note: If sampler is bound to separate Descriptor Set, then need to explicitly state which set is being used since default is 0.
  - e.g. layout(set=1, binding=0)
- To use sampler2D, the same as OpenGL version of GLSL: texture(textureSampler, texel);

Summary

- Textures need both an Image and a Sampler
- Images loaded in from external file, then stored in buffer
- Need to transition image layouts using Pipeline Barrier
- Sampler created by defining image sampling settings
- Image and Sampler are combined in a “Combined Image Sampler” descriptor
- Descriptor bound to Pipeline at Fragment Shader
- Fragment Shader reads texels using sampler

See you next video!