Depth Buffer

Depth Buffers, creating images, and using multiple Render Pass attachments

Depth Buffer

- Currently our scenes can’t handle depth.
  - If we have one object behind another, it may be drawn in front if it is drawn second out of the two objects.
- To fix this, we use a “Depth Buffer”
- A Depth Buffer keeps track of an object’s distance from the camera.
- If at a pixel, the object being drawn is measured to be further away than the last object drawn at that pixel… then it must be *behind* the last object! So the new object should not be drawn.
- Depth Buffers keep track of this information by having a second image that holds the depth (how far away the closest object is) at each pixel, in place of colour data.

Depth Buffer Image

- First need to create a new image to hold the Depth data.
- Images we've used so far were created when Swapchain was created. Swapchain does not create Depth image, so needs to be created manually.
- Can be done so with VkImageCreateInfo that takes information about the image.
- However, this struct does not create the image data itself, only the header describing that image.
- Must designate Device Memory for actual image.
- Device Memory is then bound to the VkImage, and an Image View is generated from this.
- We do not need to map any memory to it though! The depth values are calculated by the Pipeline itself.

A New Render Pass Attachment

- Much like the Swapchain images, the Depth image will be an attachment that is transitioned at each stage of the render pass.
- New attachment starts as undefined, transitions to VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL and then stays in this form.
- Dependencies do not need to change as the transitions happen at same time as Swapchain images.
- Subpass does not refer to the depth image in the colour attachment, but instead through the pDepthStencilAttachment.

A New Framebuffer Attachment

- The Depth Buffer's Image View is then attached to the Framebuffer.
- It is important to ensure the Framebuffer attachments and Render Pass attachments line up!
- If Render Pass has Colour Attachment at 0 and Depth Attachment at 1, then the Framebuffer will also need this arrangement.
- Otherwise: The Depth will output to the Colour, and the Colour to the Depth! Not good!

Updating the Pipeline

- Recall when creating the Pipeline, the struct: VkPipelineDepthStencilStateCreateInfo
- Can now set this to enable depth testing.
- **depthTestEnable:** Enables ability to check depth values and choose whether to replace pixel or not.
- **depthWriteEnable:** Enables ability to replace depth values in depth buffer. Disabling this can be used if you have a pre-written template you wish to compare depth values too, without altering it.
- **depthCompareOp:** Logical operation to determine if a depth value is in front of another.

Updating the Commands

- Lastly, don’t forget to update the clear colour when recording commands!
- Depth is set to clear at start of every render pass so values of previous passes don’t affect current.
- Render Pass Begin Info needs to update pClearValues to hold both colour attachment clear, and depth clear.
- IMPORTANT: Make sure the position of the clears matches the order of the render pass and framebuffer attachments!
- Otherwise the colour attachment will be cleared with the depth value, and the depth attachment with the colour value.

Summary

- Depth Buffer is an image recording the points closest to the camera. Used to ensure objects overlap correctly.
- Create a new image and attach to memory.
- Setup new Render Pass Attachment for Depth and give to subpass.
- Add Image as attachment to Framebuffer. Ensure order of attachments matches order of Render Pass attachments.
- Update Pipeline to enable depth testing and writing.
- Update Command Buffers to use clear value for Depth Attachment.

See you next video!