Subpasses

Multiple subpasses using Input Attachments

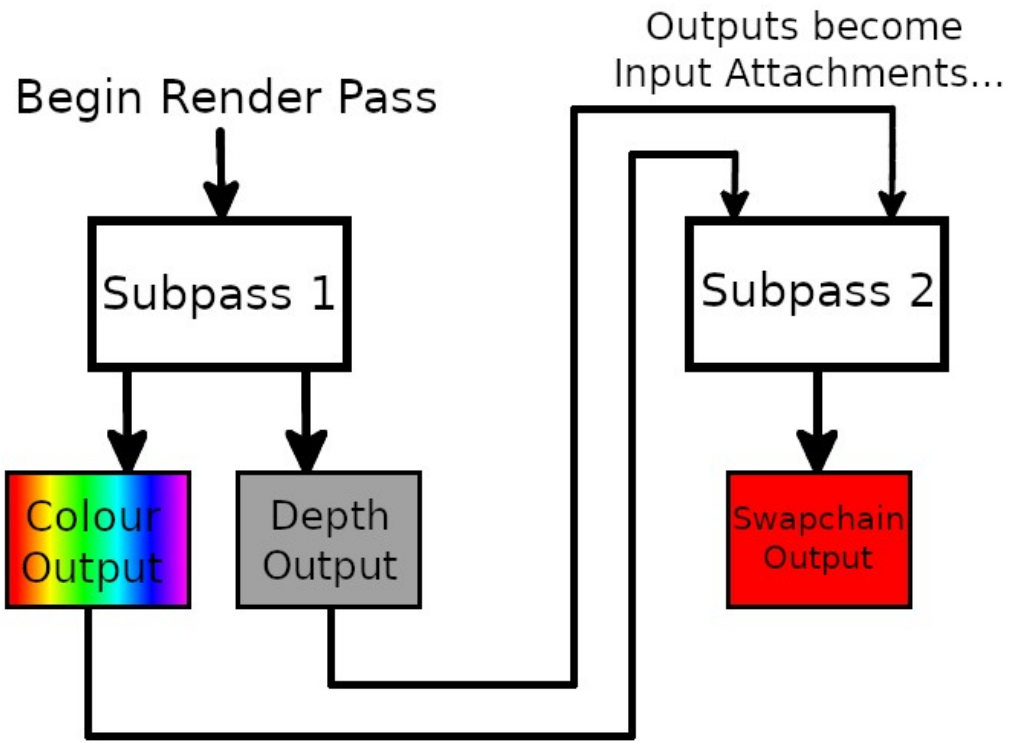
Subpasses

- Subpasses are individual passes within a single Render Pass
- Render Pass handles attachments and clears them (usually) at the start…
- … and then one or more subpasses operate on those attachments in succession.
- Subpasses can reuse attachments and even take attachments from past subpasses as input!
- However, reading from input attachments only allows reading from current fragment coordinate
  - So if Fragment Shader is currently working on Fragment (5, 3) then it can only access data on input attachment at position (5, 3)
  - This differs from samplers which can access *any* part of an image at any point in time
- Good for things such as Deferred Rendering!

Structure of Basic Two Subpasses

- Using 3 attachments now:
  - 2 Colour Attachments (Colour Attachment + Swapchain Attachment)
  - 1 Depth Attachment
- 1. First Subpass outputs to Colour Attachment and Depth Attachment
- 2. Colour Attachment used as Input Attachment for Second Subpass
- 3. Second Subpass reads from Colour Attachment and outputs to Swapchain Attachment
- Note: Can also use Depth Attachment as second Input Attachment. Our code will do this.

Structure of Basic Two Subpasses



Reusing Attachments As Input

- An Input Attachment is an attachment that is read from, as opposed to written to.
- Can be in any data format (e.g. colour or depth)
- Just use different reference to attachment for second subpass!
- And make sure to update Subpass dependencies.

Descriptors

- Despite input attachment being passed into subpass, still need a Descriptor Set to access and represent the image data!
- Creation is exactly the same as before:
  - Descriptor Set Layout describing data layout
  - Descriptor Pool to allocate from
  - Bind Descriptor Set to data
- Using VK_DESCRIPTOR_TYPE_INPUT_ATTACHMENT type now!
- Bound as image, with layout set as VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL

Pipeline and Shaders

- Recall: When creating Graphics Pipeline, we defined the subpass it would be used for. Could only designate one though!
- Need to create a new Pipeline with new shaders
  - First Pipeline: Used by first Subpass to render scene normally and output colour and depth data.
  - Second Pipeline: Used by second Subpass to take in input data from first subpass. Will then render one large triangle that fills the screen, forcing Fragment operations on every Fragment, so we can copy across data pixel-by-pixel.

Input Attachment in Shaders

- Will be used in Fragment Shader to copy across each pixel
- layout (input_attachment_index = 0, binding = 0) uniform subpassInput inputColour;
- Uses type “subpassInput”
- Must specify which input attachment in layout, as well as binding!
- Pixel can then be accessed using “subpassLoad” function:
  - subpassLoad(inputColour).rgba
  - **Remember:** Will automatically access fragment that relates to current fragment. Can’t specify specific fragment.

Summary

- Subpasses allow multiple passes in a single Render Pass.
- Allow use of Input Attachments… but only able to access Fragment coordinates that match coordinates of current Fragment being rendered!
- Reuse attachments by creating multiple references.
- Still need to represent data with Descriptor Sets
- Also need a new Pipeline for each subpass!
- Can then access input attachment fragments inside shaders

See you next video!