Multipass Rendering

Mike Bailey
mjb@cs.oregonstate.edu
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

Multipass Rendering uses Attachments -- What is a Vulkan Attachment Anyway?

"[An attachment is] an image associated with a renderpass that can be used as the input or output of one or more of its subpasses."

— Vulkan Programming Guide

An attachment can be written to, read from, or both.

Subpass
Attachment
Framebuffer
Subpass
Attachment
Subpass
Attachment
Framebuffer

Back in Our Single-pass Days

So far, we've only performed single-pass rendering, within a single Vulkan RenderPass.

Here comes a quick reminder of how we did that.

Afterwards, we will extend that.

Back in Our Single-pass Days, I

So far, we've only performed single-pass rendering, within a single Vulkan RenderPass.

Back in Our Single-pass Days, II

So far, we've only performed single-pass rendering, but within a single Vulkan RenderPass, we can also have several subpasses, each of which is feeding information to the next subpass or subpasses.

In this case, we will look at following up a 3D rendering with some image processing on the outcome.
Multispectral Algorithm to Render and then Image Process

Original  | Sharpened  | Edge Detected
---|---|---
No Noise  |  |  
Noise  |  |  

No Noise

Noise

VkAttachmentReference outputReference;

VkAttachmentReference depthReference;

VkAttachmentReference colorReference;

vsdp[1].srcSubpass = 0; // 3D rendering

depthReference.attachment = 1;

colorReference.attachment = 0;

vrpci.pAttachments =

vrpci.attachmentCount = 3; // color, depth/stencil, output

vrpci.flags = 0;

vrpci.pNext = nullptr;

vrpci.sType = VK_STRUCTURE_TYPE_RENDER_PASS_CREATE_INFO;

vrpci.pSubpasses =

vrpci.subpassCount = 2;

vrpci.pDependencies =

vrpci.dependencyCount = 1;

vsdp[0].dependencyFlags = VK_DEPENDENCY_BY_REGION_BIT;

vsdp[0].dstAccessMask = VK_ACCESS_SHADER_READ_BIT;

vsdp[0].srcAccessMask = VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT;

vsdp[0].dstStageMask = VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT;

vsdp[0].dstSubpass = 1; // image processing

vkCmdBindPipeline( CommandBuffers[nextImageIndex], VK_PIPELINE_BIND_POINT_GRAPHICS,

vkCmdBeginRenderPass

// first subpass is automatically started here

vkCmdEndRenderPass

// second subpass is started here – doesn't need any new drawing vkCmd's

vkCmdNextSubpass

vkCmdDraw( CommandBuffers[nextImageIndex], vertexCount, instanceCount, firstVertex, firstInstance );

vkCmdBindVertexBuffers( CommandBuffers[nextImageIndex], 0, 1, vBuffers, offsets );

vkCmdBindDescriptorSets( CommandBuffers[nextImageIndex], VK_PIPELINE_BIND_POINT_GRAPHICS,

GraphicsPipelineLayout, 0, 4, DescriptorSets, 0, (uint32_t *) nullptr );

GraphicsPipeline );

vad[0].format = VK_FORMAT_B8G8R8A8_SRGB;

vad[0].flags = 0;

vad[0].finalLayout = VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL;

vad[0].initialLayout = VK_IMAGE_LAYOUT_UNDEFINED;

vad[0].stencilStoreOp = VK_ATTACHMENT_STORE_OP_DONT_CARE;

vad[0].stencilLoadOp = VK_ATTACHMENT_LOAD_OP_DONT_CARE;

vad[0].storeOp = VK_ATTACHMENT_STORE_OP_STORE;

vad[0].loadOp = VK_ATTACHMENT_LOAD_OP_CLEAR;

vad[0].samples = VK_SAMPLE_COUNT_1_BIT;

vad[1].format = VK_FORMAT_D32_SFLOAT_S8_UINT;

vad[1].flags = 0;

vad[1].pPreserveAttachments = (uint32_t *) nullptr;

vad[1].preserveAttachmentCount = 0;

vad[1].pDepthStencilAttachment = (VkAttachmentReference *) nullptr;

vad[1].pResolveAttachments = (VkAttachmentReference *)nullptr;

vad[1].pColorAttachments = &outputReference;

vad[1].colorAttachmentCount = 1;

vad[1].pInputAttachments = colorReference;

vad[1].inputAttachmentCount = 1;

vad[1].pipelineBindPoint = VK_PIPELINE_BIND_POINT_GRAPHICS;

vad[2].flags = 0;

vad[2].pPreserveAttachments = (uint32_t *) nullptr;

vad[2].preserveAttachmentCount = 0;

vad[2].pDepthStencilAttachment = &depthReference;

vad[2].pResolveAttachments = (VkAttachmentReference *)nullptr;

vad[2].pColorAttachments = colorReference;

vad[2].pInputAttachments = (VkAttachmentReference *)nullptr;

vad[2].pipelineBindPoint = VK_PIPELINE_BIND_POINT_GRAPHICS;

vad[2].finalLayout = VK_IMAGE_LAYOUT_PRESENT_SRC_KHR;

vad[2].initialLayout = VK_IMAGE_LAYOUT_UNDEFINED;

vad[2].stencilStoreOp = VK_ATTACHMENT_STORE_OP_DONT_CARE;

vad[2].stencilLoadOp = VK_ATTACHMENT_LOAD_OP_DONT_CARE;

vad[2].samples = VK_SAMPLE_COUNT_1_BIT;

vad[2].flags = 0;