Antialiasing and Multisampling

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

Aliasing

"Aliasing" is a signal-processing term for "under-sampled compared with the frequencies in the signal".

Nyquist Criterion

"The Nyquist [sampling] rate is twice the maximum component frequency of the function [i.e., signal] being sampled." — Wikipedia

Anti-aliasing

MultiSampling

Multisampling is a computer graphics technique to improve the quality of your output image by looking inside every pixel to see what the rendering is doing there.

There are two approaches to this:

1. Supersampling: Pick some number of unique points within a pixel, render the image into each of these sub-pixels (including depth and stencil tests), then average them together.

2. Multisampling: Pick some number of unique points within each pixel and perform a depth and stencil render there. Then, perform a single color render for that pixel. Assign that RGBA to all the sub-pixels that made it through the depth and stencil tests.
Consider Two Triangles Whose Edges Pass Through the Same Pixel

Supersampling

Supersampling

Multisampling

Setting up the Image
VkPipelineMultisampleStateCreateInfo vpmsci;

vpmsci.minSampleShading = 0.5;

At least this fraction of samples will get their own fragment shader calls (as long as they pass the depth and stencil tests).

0. produces simple multisampling
(0.,1.) produces partial supersampling
1. Produces complete supersampling

VkAttachmentDescription vad[2];

vad[0].format = VK_FORMAT_B8G8R8A8_SRGB;

vad[0].samples = VK_SAMPLE_COUNT_8_BIT;

vad[0].loadOp = VK_ATTACHMENT_LOAD_OP_CLEAR;

vad[0].storeOp = VK_ATTACHMENT_STORE_OP_STORE;

vad[0].stencilLoadOp = VK_ATTACHMENT_LOAD_OP_DONT_CARE;

vad[0].stencilStoreOp = VK_ATTACHMENT_STORE_OP_DONT_CARE;

vad[0].initialLayout = VK_IMAGE_LAYOUT_UNDEFINED;

vad[0].finalLayout = VK_IMAGE_LAYOUT_PRESENT_SRC_KHR;

vad[0].flags = 0;

vad[1].format = VK_FORMAT_D32_SFLOAT_S8_UINT;

vad[1].samples = VK_SAMPLE_COUNT_8_BIT;

vad[1].loadOp = VK_ATTACHMENT_LOAD_OP_CLEAR;

vad[1].storeOp = VK_ATTACHMENT_STORE_OP_DONT_CARE;

vad[1].stencilLoadOp = VK_ATTACHMENT_LOAD_OP_DONT_CARE;

vad[1].stencilStoreOp = VK_ATTACHMENT_STORE_OP_DONT_CARE;

vad[1].initialLayout = VK_IMAGE_LAYOUT_UNDEFINED;

vad[1].finalLayout = VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL;

vad[1].flags = 0;

VkAttachmentReference colorReference;

colorReference.attachment = 0;

colorReference.layout = VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL;

VkAttachmentReference depthReference;

depthReference.attachment = 1;

depthReference.layout = VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL;

VkSubpassDescription vsd;

vsd.flags = 0;

vsd.pipelineBindPoint = VK_PIPELINE_BIND_POINT_GRAPHICS;

vsd.inputAttachmentCount = 0;

vsd.pInputAttachments = (VkAttachmentReference *)nullptr;

vsd.colorAttachmentCount = 1;

vsd.pColorAttachments = &colorReference;

vsd.pResolveAttachments = (VkAttachmentReference *)nullptr;

vsd.pDepthStencilAttachment = &depthReference;

vsd.preserveAttachmentCount = 0;

vsd.pPreserveAttachments = (uint32_t *)nullptr;

VkRenderPassCreateInfo vrpci;

vrpci.sType = VK_STRUCTURE_TYPE_RENDER_PASS_CREATE_INFO;

vrpci.pNext = nullptr;

vrpci.flags = 0;

vrpci.attachmentCount = 2;              // color and depth/stencil

vrpci.pAttachments = vad;

vrpci.subpassCount = 1;

vrpci.pSubpasses = &vsd;

vrpci.dependencyCount = 0;

vrpci.pDependencies = (VkSubpassDependency *)nullptr;

result = vkCreateRenderPass( LogicalDevice, IN &vrpci, PALLOCATOR, OUT &RenderPass );

VkSubpassDescription vspc;

vspc.flags = 0;

vspc.pipelineBindPoint = VK_PIPELINE_BIND_POINT_GRAPHICS;

vspc.inputAttachmentCount = 0;

vspc.pInputAttachments = (VkAttachmentReference *)nullptr;

vspc.colorAttachmentCount = 0;

vspc.colorAttachment = 0;

vspc.pColorAttachments = (VkAttachmentReference *)nullptr;

vspc.pResolveAttachments = (VkAttachmentReference *)nullptr;

vspc.pDepthStencilAttachment = 0;

vspc.pDepthStencilAttachment = 0;

vspc.preserveAttachmentCount = 0;

vspc.pPreserveAttachments = (uint32_t *)nullptr;

result = vkCreateRenderPass( LogicalDevice, IN &vspc, PALLOCATOR, OUT &RenderPass );

VkAttachmentReference colorReference2;

colorReference2.attachment = 2;

colorReference2.layout = VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL;

VkAttachmentReference depthReference2;

depthReference2.attachment = 3;

depthReference2.layout = VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL;

VkSubpassDescription vspc2;

vspc2.flags = 0;

vspc2.pipelineBindPoint = VK_PIPELINE_BIND_POINT_GRAPHICS;

vspc2.inputAttachmentCount = 0;

vspc2.pInputAttachments = (VkAttachmentReference *)nullptr;

vspc2.colorAttachmentCount = 0;

vspc2.colorAttachment = 0;

vspc2.pColorAttachments = (VkAttachmentReference *)nullptr;

vspc2.pResolveAttachments = (VkAttachmentReference *)nullptr;

vspc2.pDepthStencilAttachment = 0;

vspc2.pDepthStencilAttachment = 0;

vspc2.preserveAttachmentCount = 0;

vspc2.pPreserveAttachments = (uint32_t *)nullptr;

result = vkCreateRenderPass( LogicalDevice, IN &vspc2, PALLOCATOR, OUT &RenderPass2 );

VkImageSubresourceRange vsr;

vsr.aspectMask = VK_IMAGE_ASPECT_COLOR_BIT;

vsr.mipLevel = 0;

vsr.arrayLayers = 0;

result = vkCreateImageSubpass( LogicalDevice, IN &vsr, PALLOCATOR, OUT &ImageSubpass );

result = vkCreateImageSubpass( LogicalDevice, IN &vsr, PALLOCATOR, OUT &ImageSubpass2 );

vkCreateResolveImage( cmdBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, 1, &vir );

vkCreateResolveImage( cmdBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, 1, &vir2 );

vkCreateResolveImage( cmdBuffer, srcImage, srcImageLayout, dstImage, dstImageLayout, 1, &vir3 );