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".

What the signal really is: what we want

Sampling Interval

What we think the signal is: too often, what we get

Sampled Points

Nyquist Criterion

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

Anti-aliasing

4x

16x
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:

1. **Supersampling:** Pick some number of unique sub-pixels within a pixel, render the image at each of these individual sub-pixels (including depth and stencil tests), then average them together. This results in lots of renders.

2. **Multisampling:** Perform a single color render for the one entire pixel. Then, pick some number of unique sub-pixels within that pixel and perform depth and stencil tests there. Assign the single color to all the sub-pixels that made it through the depth and stencil tests. Note: per-sample depth and stencil tests are performed first to decide which color renders actually should be done.

Consider Two Triangles Whose Edges Pass Through the Same Pixel

**Supersampling**

\[
\text{Final Pixel Color} = \sum \text{Color sample from subpixel}
\]

# Fragment Shader calls = 8

**Multisampling**

\[
\text{Final Pixel Color} = 3 \times \text{One color sample from A} + 5 \times \text{One color sample from B}
\]

# Fragment Shader calls = 2
VkPipelineMultisampleStateCreateInfo vpmsci;
vpmsci.sType = VK_STRUCTURE_TYPE_PIPELINE_MULTISAMPLE_STATE_CREATE_INFO;
vpmsci.pNext = nullptr;
vpmsci.flags = 0;
vpmsci.rasterizationSamples = VK_SAMPLE_COUNT_8_BIT;
vpmsci.sampleShadingEnable = VK_TRUE;
vpmsci.minSampleShading = 0.5f;
vpmsci.pSampleMask = (VkSampleMask *)nullptr;
vpmsci.alphaToCoverageEnable = VK_FALSE;
vpmsci.alphaToOneEnable = VK_FALSE;

VkGraphicsPipelineCreateInfo vgpci;
vgpci.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;
vgpci.pNext = nullptr;
...;
vgpci.pMultisampleState = &vpmsci;

result = vkCreateGraphicsPipelines(LogicalDevice, VK_NULL_HANDLE, 1, IN &, vgpci, PALLOCATOR, OUT pGraphicsPipeline);

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);