CIS565 GPU Programming and Architecture

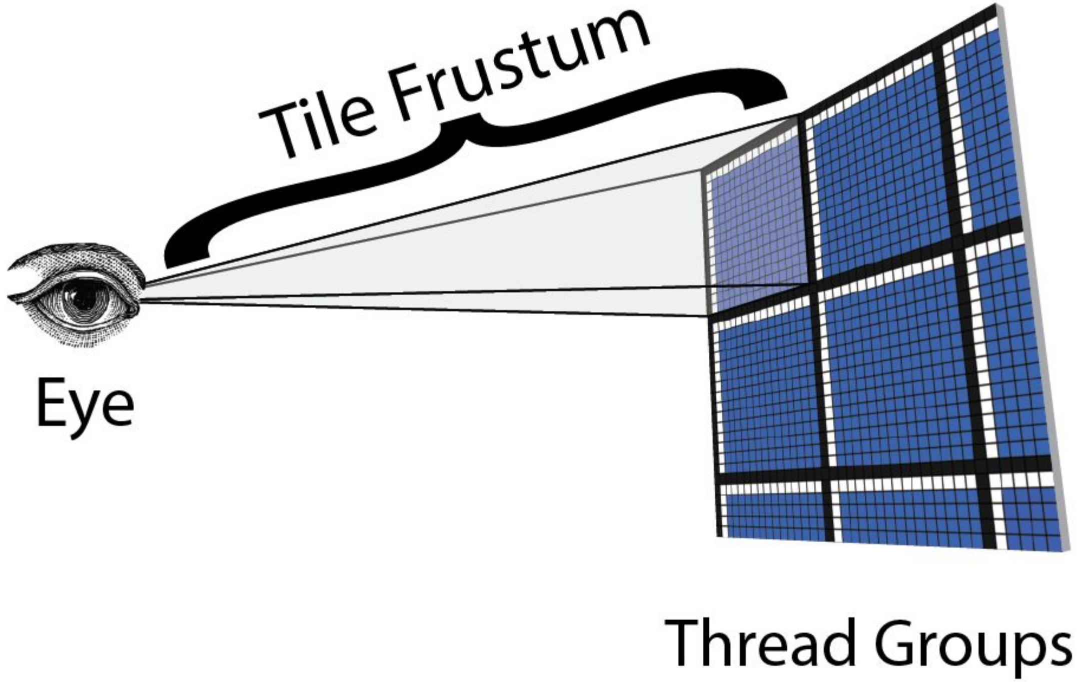
Final Project Milestone I

Forward + with Vulkan

Zimeng Yang & Liang Peng

Instructor: Patrick Cozzi

Forward + Rendering

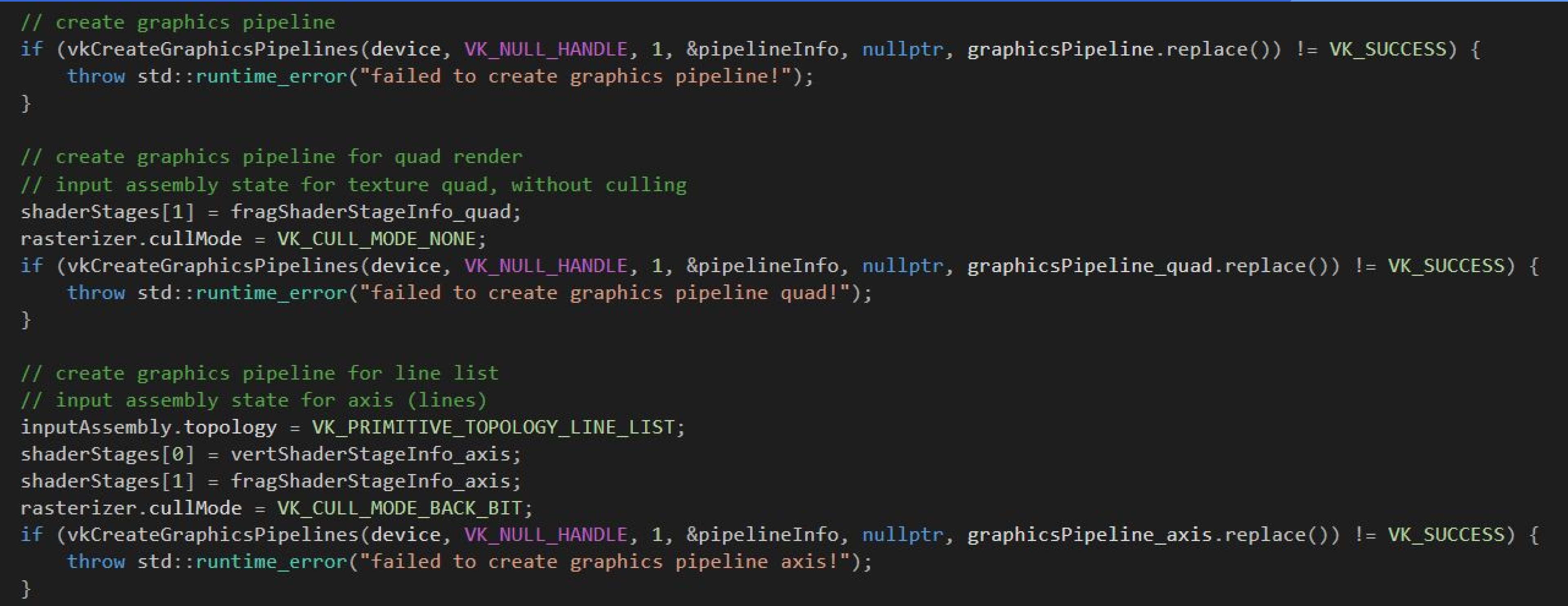


Finished Tasks

- Walked through Vulkan tutorial
- Environment setup and basic pipeline
- Multiple pipeline, vertex buffer, index buffer, and texture; model loading
- Implemented camera to change view direction

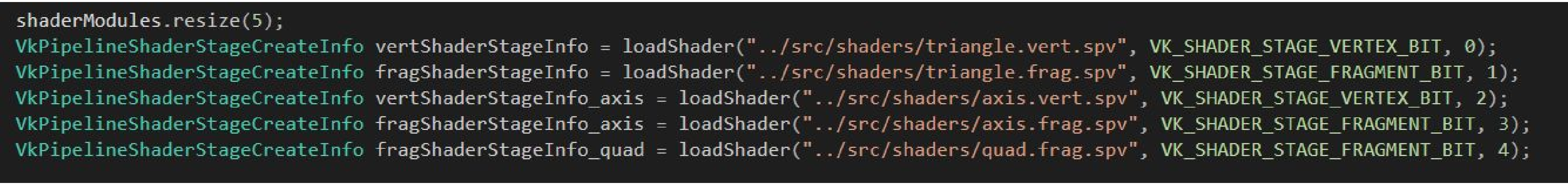
GIF

Working Code

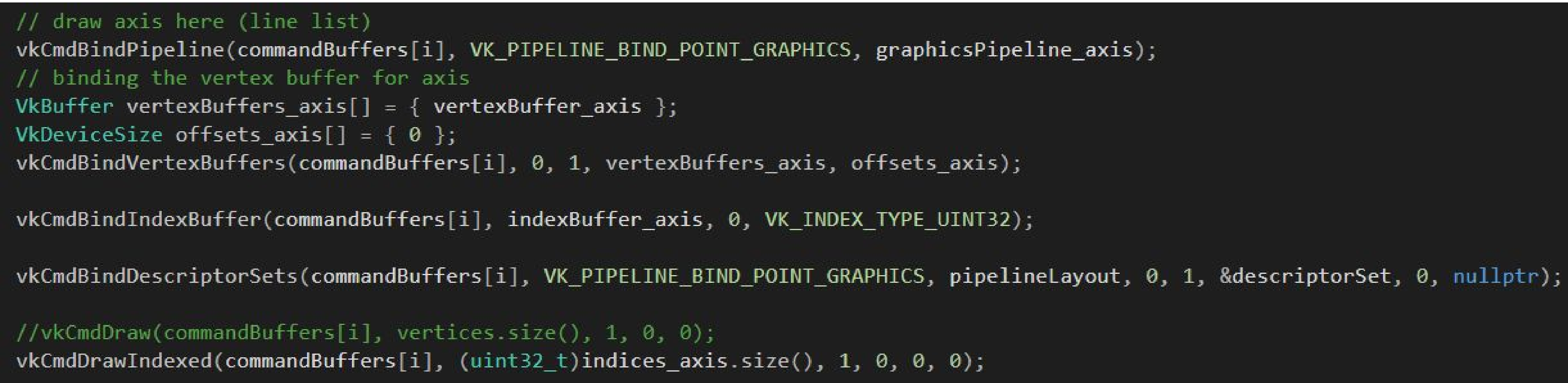


```
// create graphics pipeline
if (vkCreateGraphicsPipelines(device, VK_NULL_HANDLE, 1, &pipelineInfo, nullptr, graphicsPipeline.replace()) != VK_SUCCESS) {
    throw std::runtime_error("failed to create graphics pipeline!");
}

// create graphics pipeline for quad render
// input assembly state for texture quad, without culling
shaderStages[1] = fragShaderStageInfo_quad;
rasterizer.cullMode = VK_CULL_MODE_NONE;
if (vkCreateGraphicsPipelines(device, VK_NULL_HANDLE, 1, &pipelineInfo, nullptr, graphicsPipeline_quad.replace()) != VK_SUCCESS) {
    throw std::runtime_error("failed to create graphics pipeline quad!");
}

// create graphics pipeline for line list
// input assembly state for axis (lines)
inputAssembly.topology = VK_PRIMITIVE_TOPOLOGY_LINE_LIST;
shaderStages[0] = vertShaderStageInfo_axis;
shaderStages[1] = fragShaderStageInfo_axis;
rasterizer.cullMode = VK_CULL_MODE_BACK_BIT;
if (vkCreateGraphicsPipelines(device, VK_NULL_HANDLE, 1, &pipelineInfo, nullptr, graphicsPipeline_axis.replace()) != VK_SUCCESS) {
    throw std::runtime_error("failed to create graphics pipeline axis!");
}
```

Working Code



```
shaderModules.resize(5);
VkPipelineShaderStageCreateInfo vertShaderStageInfo = loadShader("../src/shaders/triangle.vert.spv", VK_SHADER_STAGE_VERTEX_BIT, 0);
VkPipelineShaderStageCreateInfo fragShaderStageInfo = loadShader("../src/shaders/triangle.frag.spv", VK_SHADER_STAGE_FRAGMENT_BIT, 1);
VkPipelineShaderStageCreateInfo vertShaderStageInfo_axis = loadShader("../src/shaders/axis.vert.spv", VK_SHADER_STAGE_VERTEX_BIT, 2);
VkPipelineShaderStageCreateInfo fragShaderStageInfo_axis = loadShader("../src/shaders/axis.frag.spv", VK_SHADER_STAGE_FRAGMENT_BIT, 3);
VkPipelineShaderStageCreateInfo fragShaderStageInfo_quad = loadShader("../src/shaders/quad.frag.spv", VK_SHADER_STAGE_FRAGMENT_BIT, 4);
```



```
// draw axis here (line list)
vkCmdBindPipeline(commandBuffers[i], VK_PIPELINE_BIND_POINT_GRAPHICS, graphicsPipeline_axis);
// binding the vertex buffer for axis
VkBuffer vertexBuffers_axis[] = { vertexBuffer_axis };
VkDeviceSize offsets_axis[] = { 0 };
vkCmdBindVertexBuffers(commandBuffers[i], 0, 1, vertexBuffers_axis, offsets_axis);

vkCmdBindIndexBuffer(commandBuffers[i], indexBuffer_axis, 0, VK_INDEX_TYPE_UINT32);

vkCmdBindDescriptorSets(commandBuffers[i], VK_PIPELINE_BIND_POINT_GRAPHICS, pipelineLayout, 0, 1, &descriptorSet, 0, nullptr);

//vkCmdDraw(commandBuffers[i], vertices.size(), 1, 0, 0);
vkCmdDrawIndexed(commandBuffers[i], (uint32_t)indices_axis.size(), 1, 0, 0, 0);
```

Upcoming Milestones

- Milestone II
  - Basic forward rendering
  - Include compute shader stage in pipeline
  - Compute grid frustum for each tile
- Milestone III
  - Compute light list for each tile
  - Compute lighting in each tile
- Final Presentation
  - Performance Analysis

Thank you!