Logical Devices

Vulkan: Overall Block Diagram

Instance

Physical Device

Logical Device

Vulkan: a More Typical (and Simplified) Block Diagram

Application

Instance

Physical Device

Logical Device

Queue

Command Buffer

Looking to See What Device Layers are Available

const char * myDeviceLayers[] = {
    //"VK_LAYER_LUNARG_api_dump",
    //"VK_LAYER_LUNARG_core_validation",
    //"VK_LAYER_LUNARG_image",
    "VK_LAYER_LUNARG_object_tracker",
    "VK_LAYER_LUNARG_parameter_validation",
    //"VK_LAYER_NV_optimus"
};

const char * myDeviceExtensions[] = {
    "VK_KHR_surface",
    "VK_KHR_win32_surface",
    "VK_EXT_debug_report",
    //"VK_KHR_swapchains"
};

// see what device layers are available:
uint32_t layerCount;
vkEnumerateDeviceLayerProperties(PhysicalDevice, &layerCount, (VkLayerProperties *)nullptr);
VkLayerProperties * deviceLayers = new VkLayerProperties[layerCount];
result = vkEnumerateDeviceLayerProperties(PhysicalDevice, &layerCount, deviceLayers);

Looking to See What Device Extensions are Available

// see what device extensions are available:
uint32_t extensionCount;
vkEnumerateDeviceExtensionProperties(PhysicalDevice, deviceLayers[i].layerName, &extensionCount, (VkExtensionProperties *)nullptr);
VkExtensionProperties * deviceExtensions = new VkExtensionProperties[extensionCount];
result = vkEnumerateDeviceExtensionProperties(PhysicalDevice, deviceLayers[i].layerName, &extensionCount, deviceExtensions);

What Device Layers and Extensions are Available

3 physical device layers enumerated:
0x00400038   1  'VK_LAYER_NV_optimus'  'NVIDIA Optimus layer'
0 device extensions enumerated for 'VK_LAYER_NV_optimus':

0x00400033   1  'VK_LAYER_LUNARG_object_tracker'  'LunarG Validation Layer'
0 device extensions enumerated for 'VK_LAYER_LUNARG_object_tracker':

0x00400033   1  'VK_LAYER_LUNARG_parameter_validation'  'LunarGValidation Layer'
0 device extensions enumerated for 'VK_LAYER_LUNARG_parameter_validation':

mjb – September 18, 2018
Computer Graphics
float queuePriorities[1] =
{
  1.
};

VkDeviceQueueCreateInfo vdqci;
vdqci.sType = VK_STRUCTURE_TYPE_DEVICE_QUEUE_CREATE_INFO;
vdqci.pNext = nullptr;
vdqci.flags = 0;
vdqci.queueFamilyIndex = 0;
vdqci.queueCount = 1;
vdqci.pQueueProperties = queuePriorities;

VkDeviceCreateInfo vdci;
vdci.sType = VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO;
vdci.pNext = nullptr;
vdci.flags = 0;
vdci.queueCreateInfoCount = 1; // # of device queues
vdci.pQueueCreateInfos = &vdqci; // array of VkDeviceQueueCreateInfo/s
vdci.enabledLayerCount = sizeof(myDeviceLayers) / sizeof(char *);
vdcienabledLayerCount = 0;
vdci.ppEnabledLayerNames = myDeviceLayers;
vdci.enabledExtensionCount = 0;
vdci.ppEnabledExtensionNames = (const char **)nullptr; // no extensions
vdci.enabledExtensionCount = sizeof(myDeviceExtensions) / sizeof(char *);
vdci.ppEnabledExtensionNames = myDeviceExtensions;
vdci.pEnabledFeatures = &PhysicalDeviceFeatures;

result = vkCreateLogicalDevice( PhysicalDevice, &vdci, PALLOCATOR, OUT &LogicalDevice );

// get the queue for this logical device:
vkGetDeviceQueue( LogicalDevice, 0, 0, OUT &Queue ); // 0, 0 = queueFamilyIndex, queueIndex