Logical Devices

Vulkan: Overall Block Diagram

Application

Instance

Instance

Physical Device

Physical Device

Physical Device

Logical Device

Logical Device

Logical Device

Logical Device

Queue

Queue

Queue

Queue

Queue

Queue

Command Buffer

Command Buffer

Command Buffer
Vulkan: a More Typical (and Simplified) Block Diagram

Application
   ↓
Instance
   ↓
Physical Device
      ↓
Logical Device
         ↓
Queue
         ← Command Buffer
         ← Command Buffer
         ← Command Buffer

Looking to See What Device Layers are Available

```c
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_object_tracker",
   // "VK_LAYER_LUNARG_parameter_validation",
   // "VK_LAYER_NV_optimus"
};

const char * myDeviceExtensions[] = {
   "VK_KHR_surface",
   // "VK_KHR_win32_window",
   "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

```c
// 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

4 physical device layers enumerated:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00401063</td>
<td>VK_LAYER_NV_optimus</td>
<td>NVIDIA Optimus layer</td>
</tr>
<tr>
<td>0x00401072</td>
<td>VK_LAYER_LUNARG_core_validation</td>
<td>LunarG Validation Layer</td>
</tr>
<tr>
<td>0x00401072</td>
<td>VK_LAYER_LUNARG_object_tracker</td>
<td>LunarG Validation Layer</td>
</tr>
<tr>
<td>0x00401072</td>
<td>VK_LAYER_LUNARG_parameter_validation</td>
<td>LunarG Validation Layer</td>
</tr>
</tbody>
</table>
Vulkan: Creating a Logical Device

```c
float queuePriorities[1] =
{
  1,
};
VkDeviceQueueCreateInfo vdqi;
vdqi.sType = VK_STRUCTURE_TYPE_DEVICE_QUEUE_CREATE_INFO;
vdqi.pNext = nullptr;
vdqi.flags = 0;
vdfqi.queueFamilyIndex = 0;
vdfqi.queueCount = 1;
vdfqi.pQueueProperties = queuePriorities;
result = vkCreateLogicalDevice( PhysicalDevice, IN &vdci, PALLOCATOR, OUT &LogicalDevice );
```

Vulkan: Creating the Logical Device’s Queue

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