Physical Devices

Application

Instance

Physical Device

Logical Device

Command Buffer

Vulkan: Overall Block Diagram

Instance

Physical Device

Logical Device

Queue

Command Buffer

Vulkan: a More Typical (and Simplified) Block Diagram

Application

Instance

Physical Device

Logical Device

Command Buffer

Querying the Number of Physical Devices

uint32_t count;
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT (VkPhysicalDevice *)nullptr );

VkPhysicalDevice * physicalDevices = new VkPhysicalDevice[ count ];
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT physicalDevices );

This way of querying information is a recurring OpenCL and Vulkan pattern (get used to it):

How many total there are
Where to put them

result = vkEnumeratePhysicalDevices( Instance, &count, nullptr );
result = vkEnumeratePhysicalDevices( Instance, &count, physicalDevices );
VkResult result = VK_SUCCESS;
result = vkEnumeratePhysicalDevices( Instance, OUT &PhysicalDeviceCount, (VkPhysicalDevice *)nullptr );
if( result != VK_SUCCESS || PhysicalDeviceCount <= 0 )
{
    fprintf( FpDebug, "Could not count the physical devices
" );
    return VK_SHOULD_EXIT;
}
fprintf(FpDebug, "
%d physical devices found.
", PhysicalDeviceCount);
VkPhysicalDevice * physicalDevices = new VkPhysicalDevice[ PhysicalDeviceCount ];
result = vkEnumeratePhysicalDevices( Instance, OUT &PhysicalDeviceCount, OUT physicalDevices );
if( result != VK_SUCCESS )
{
    fprintf( FpDebug, "Could not enumerate the %d physical devices
", PhysicalDeviceCount );
    return VK_SHOULD_EXIT;
}

int discreteSelect = -1;
int integratedSelect = -1;
for( unsigned int i = 0; i < PhysicalDeviceCount; i++ )
{
    VkPhysicalDeviceProperties vpdp;
    vkGetPhysicalDeviceProperties( IN physicalDevices[i], OUT &vpdp );
    if( result != VK_SUCCESS )
    {
        fprintf( FpDebug, "Could not get the physical device properties of device %d
", i );
        return VK_SHOULD_EXIT;
    }
    fprintf( FpDebug, "

Device %2d:
", i );
    fprintf( FpDebug, "	API version: %d
", vpdp.apiVersion );
    fprintf( FpDebug, "	Driver version: %d
", vpdp.apiVersion );
    fprintf( FpDebug, "	Vendor ID: 0x%04x
", vpdp.vendorID );
    fprintf( FpDebug, "	Device ID: 0x%04x
", vpdp.deviceID );
    fprintf( FpDebug, "	Physical Device Type: %d =",
", vpdp.deviceType );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU )
        fprintf( FpDebug, " (Discrete GPU)
" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU )
        fprintf( FpDebug, " (Integrated GPU)
" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_VIRTUAL_GPU )
        fprintf( FpDebug, " (Virtual GPU)
" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU )
        fprintf( FpDebug, " (CPU)
" );
    fprintf( FpDebug, "	Device Name: %s
", vpdp.deviceName );
    fprintf( FpDebug, "	Pipeline Cache Size: %d
", vpdp.pipelineCacheUUID[0] );
}/
Which Physical Device to Use, I

Which Physical Device to Use, II

Asking About the Physical Device’s Features
Here's What the NVIDIA A6000 Produced

<table>
<thead>
<tr>
<th>Device 0:</th>
</tr>
</thead>
<tbody>
<tr>
<td>API version: 4206797</td>
</tr>
<tr>
<td>Driver version: 4206797</td>
</tr>
<tr>
<td>Vendor ID: 0x10de</td>
</tr>
<tr>
<td>Device ID: 0x2230</td>
</tr>
<tr>
<td>Physical Device Type: 2 = (Discrete GPU)</td>
</tr>
<tr>
<td>Device Name: NVIDIA RTX A6000</td>
</tr>
<tr>
<td>Pipeline Cache Size: 72</td>
</tr>
</tbody>
</table>

Device #0 selected ('NVIDIA RTX A6000')

Physical Device Features:
- geometryShader = 1
- tessellationShader = 1
- multiDrawIndirect = 1
- wideLines = 1
- largePoints = 1
- multiViewport = 1
- occlusionQueryPrecise = 1
- pipelineStatisticsQuery = 1
- shaderFloat64 = 1
- shaderInt64 = 1
- shaderInt16 = 1

Here's What the Intel HD Graphics 520 Produced

<table>
<thead>
<tr>
<th>Device 0:</th>
</tr>
</thead>
<tbody>
<tr>
<td>API version: 4194360</td>
</tr>
<tr>
<td>Driver version: 4194360</td>
</tr>
<tr>
<td>Vendor ID: 0x8086</td>
</tr>
<tr>
<td>Device ID: 0x1916</td>
</tr>
<tr>
<td>Physical Device Type: 1 = (Integrated GPU)</td>
</tr>
<tr>
<td>Device Name: Intel(R) HD Graphics 520</td>
</tr>
<tr>
<td>Pipeline Cache Size: 213</td>
</tr>
</tbody>
</table>

Device #0 selected ('Intel(R) HD Graphics 520')

Physical Device Features:
- geometryShader = 1
- tessellationShader = 1
- multiDrawIndirect = 1
- wideLines = 1
- largePoints = 1
- multiViewport = 1
- occlusionQueryPrecise = 1
- pipelineStatisticsQuery = 1
- shaderFloat64 = 1
- shaderInt64 = 1
- shaderInt16 = 1

Asking About the Physical Device's Different Memories

```
VkPhysicalDeviceMemoryProperties vpdmp;
vkGetPhysicalDeviceMemoryProperties( PhysicalDevice, OUT &vpdmp );

fprintf( FpDebug, "%d Memory Types:
", vpdmp.memoryTypeCount );
for( unsigned int i = 0; i < vpdmp.memoryTypeCount; i++ ){
    VkMemoryType vmt = vpdmp.memoryTypes[i];
    fprintf( FpDebug, "Memory %2d: ", i );
    if( ( vmt.propertyFlags & VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT       )   != 0 )    fprintf( FpDebug, " DeviceLocal" );
    if( ( vmt.propertyFlags & VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT       )     != 0 )    fprintf( FpDebug, " HostVisible" );
    if( ( vmt.propertyFlags & VK_MEMORY_PROPERTY_HOST_COHERENT_BIT )    != 0 )    fprintf( FpDebug, " HostCoherent" );
    if( ( vmt.propertyFlags & VK_MEMORY_PROPERTY_HOST_CACHED_BIT       )   != 0 )    fprintf( FpDebug, " HostCached" );
    if( ( vmt.propertyFlags & VK_MEMORY_PROPERTY_LAZILY_ALLOCATED_BIT ) != 0 )    fprintf( FpDebug, " LazilyAllocated" );
    fprintf(FpDebug, "\n" );
}

fprintf( FpDebug, "%d Memory Heaps:
", vpdmp.memoryHeapCount );
for( unsigned int i = 0; i < vpdmp.memoryHeapCount; i++ ){
    VkMemoryHeap vmh = vpdmp.memoryHeaps[i];
    fprintf( FpDebug, "Heap %d: ", i );
    fprintf(FpDebug, " size = 0x%08lx", (unsigned long int)vmh.size );
    if( ( vmh.flags & VK_MEMORY_HEAP_DEVICE_LOCAL_BIT  ) != 0 )     fprintf( FpDebug, " DeviceLocal" );    // only one in use
    fprintf(FpDebug, "\n" );
}
```

Here's What I Got on the A6000's

```
6 Memory Types:
Memory 0: DeviceLocal
Memory 2: HostVisible HostCoherent
Memory 3: HostVisible HostCoherent HostCached
Memory 4: DeviceLocal HostVisible HostCached
Memory 5: DeviceLocal

4 Memory Heaps:
Heap 0: size = 0xffffffff DeviceLocal
Heap 1: size = 0xffffffff DeviceLocal
Heap 2: size = 0xffffffff DeviceLocal
Heap 3: size = 0xffffffff DeviceLocal
```
uint32_t count = -1;
vkGetPhysicalDeviceQueueFamilyProperties( IN PhysicalDevice, &count, OUT (VkQueueFamilyProperties *)nullptr );
fprintf(FpDebug, "Found %d Queue Families:
", count);
VkQueueFamilyProperties *vqfp = new VkQueueFamilyProperties[count];
vkGetPhysicalDeviceQueueFamilyProperties( IN PhysicalDevice, &count, OUT vqfp );
for( unsigned int i = 0; i < count; i++ ){
    fprintf(FpDebug, "	%d: queueCount = %2d  ;   ", i, vqfp[i].queueCount);
    if( ( vqfp[i].queueFlags & VK_QUEUE_GRAPHICS_BIT ) != 0 ) fprintf(FpDebug, " Graphics" );
    if( ( vqfp[i].queueFlags & VK_QUEUE_COMPUTE_BIT ) != 0 ) fprintf(FpDebug, " Compute ");
    if( ( vqfp[i].queueFlags & VK_QUEUE_TRANSFER_BIT ) != 0 ) fprintf(FpDebug, " Transfer" );
    fprintf(FpDebug, "
");
}

Asking About the Physical Device's Queue Families

Here's What I Got on the A6000's

Found 3 Queue Families:

0: Queue Family Count = 16  ;   Graphics Compute Transfer
1: Queue Family Count = 2  ;   Transfer
2: Queue Family Count = 8  ;   Compute Transfer

Here's What I Got on the A6000's