Vulkan: Identifying the 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);

Querying the Number of Physical Devices

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
    return VK_SHOULD_EXIT;
}

Which Physical Device to Use, I

Which Physical Device to Use, II

Vulkan: Overall Block Diagram

Vulkan: a More Typical (and Simplified) Block Diagram

Vulkan: Identifying the Physical Devices
Which Physical Device to Use, II

```c
fprintf( FpDebug, "Could not select a Physical Device
" );
return VK_SHOULD_EXIT;
```

Here's What the NVIDIA RTX 2080 Ti Produced

```c
vkEnumeratePhysicalDevices:
Device 0:
  API version: 4198499
  Driver version: 4198499
  Vendor ID: 0x10de
  Device ID: 0x1e04
  Device Name: 'RTX 2080 Ti'
  Physical Device Type: 2 = (Discrete GPU)
  Pipeline Cache Size: 236

Physical Device Features:
  geometryShader = 1
  tesselationShader = 1
  multiDrawIndirect = 1
  multiViewport = 1
  largePoints = 1
  wideLines = 1
  multiDrawIndirect = 1
  geometryShader = 1

Performance:

Host Memory Types:
  size = 0x12300000

Memory Heaps:
  Heap 1:  size = 0xb7c00000
  Heap 0:  size = 0xfac00000
```

Here's What the Intel HD Graphics 520 Produced

```c
vkEnumeratePhysicalDevices:
Device 0:
  API version: 4194360
  Driver version: 4194360
  Vendor ID: 0x8086
  Device ID: 0x1916
  Device Name: 'Intel(R) HD Graphics 520'
  Physical Device Type: 1 = (Integrated GPU)
  Pipeline Cache Size: 213

Physical Device Features:
  geometryShader = 1
  multiDrawIndirect = 1
  largePoints = 1
  wideLines = 1
  multiDrawIndirect = 1
  geometryShader = 1

Performance:

Host Memory Types:
  size = 0x2000000

Memory Heaps:
  Heap 0:  size = 0x2000000
```

Asking About the Physical Device’s Features

```c
vkGetPhysicalDeviceFeatures( IN PhysicalDevice, OUT &PhysicalDeviceFeatures );
```

Here's What I Got

```c
Physical Device Features:
  geometryShader = 1
  multiDrawIndirect = 1
  largePoints = 1
  wideLines = 1
  multiDrawIndirect = 1
  geometryShader = 1
```

Asking About the Physical Device’s Different Memories

```c
vkGetPhysicalDeviceMemoryProperties( IN PhysicalDevice, OUT &vpdmp );
```

Here's What I Got

<table>
<thead>
<tr>
<th>Memory Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory 0:</td>
</tr>
<tr>
<td>Memory 1:</td>
</tr>
<tr>
<td>Memory 2:</td>
</tr>
<tr>
<td>Memory 3:</td>
</tr>
<tr>
<td>Memory 4:</td>
</tr>
<tr>
<td>Memory 5:</td>
</tr>
<tr>
<td>Memory 6:</td>
</tr>
<tr>
<td>Memory 7:</td>
</tr>
<tr>
<td>Memory 8:</td>
</tr>
<tr>
<td>Memory 9:</td>
</tr>
<tr>
<td>HostViableHostCoherent HostCached</td>
</tr>
<tr>
<td>HostViableHostCoherent HostCached</td>
</tr>
</tbody>
</table>

2 Memory Heaps:
  Heap 0:  size = 0x80000000
  Heap 1:  size = 0x80000000
Asking About the Physical Device's Queue Families

uint32_t count = -1;
vkGetPhysicalDeviceQueueFamilyProperties(IN PhysicalDevice, &count, OUT (VkQueueFamilyProperties *)nullptr);
fprintf(FpDebug, "Found %d Queue Families:");

VkQueueFamilyProperties *vqfp = new VkQueueFamilyProperties[count];
vkGetPhysicalDeviceQueueFamilyProperties(IN PhysicalDevice, &count, OUT vqfp);
for(unsigned int i = 0; i < count; i++){
fprintf(FpDebug, "\t%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, "\n");
}

Here's What I Got

- Found 3 Queue Families:
  - 0: queueCount = 16  ;   Graphics Compute Transfer
  - 1: queueCount =  2  ;   Transfer
  - 2: queueCount =  8  ;   Compute

Here's What I Got