Vulkan: Identifying the Physical Devices

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

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

```c
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;
} for( unsigned int i = 0; i < PhysicalDeviceCount; i++ ) {
    VkPhysicalDeviceProperties vpdp;
    vkGetPhysicalDeviceProperties( IN physicalDevices[i], OUT &vpdp);
    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.pipelineCacheSize);
}
```

Vulkan: Which Physical Device to Use, I

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

Vulkan: Querying the Number of Physical Devices

```c
result = vkEnumeratePhysicalDevices( Instance, OUT &count, (VkPhysicalDevice *)nullptr );
if( result != VK_SUCCESS || count <= 0 ) {
    fprintf( FpDebug, "Could not count the physical devices
    return VK_SHOULD_EXIT;
}
```

Vulkan: Overall Block Diagram

- **Instance**
- **Physical Device**
- **Logical Device**
- **Command Buffer**

Vulkan: Overall Block Diagram

- **Application**
- **Instance**
- **Physical Device**
- **Logical Device**
- **Command Buffer**

Vulkan: a More Typical (and Simplified) Block Diagram

- **Application**
- **Instance**
- **Physical Device**
- **Logical Device**
- **Queue**
- **Command Buffer**

Vulkan: Identifying the Physical Devices

- **Instance**
- **Physical Device**
- **Logical Device**
- **Command Buffer**

Vulkan: Which Physical Device to Use, I

```c
int discreteSelect = -1;
int integratedSelect = -1;
for( unsigned int i = 0; i < PhysicalDeviceCount; i++ ) {
    VkPhysicalDeviceProperties vpdp;
    vkGetPhysicalDeviceProperties( IN physicalDevices[i], OUT &vpdp);
    fprintf( FpDebug, "Could not get the physical device properties of device %d
", i );
    return VK_SHOULD_EXIT;
} fprintf( FpDebug, "Could not enumerate the %d physical devices
", PhysicalDeviceCount );
```
Which Physical Device to Use, II

// need some logic here to decide which physical device to select:
if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU )
    which = integratedSelect;
else if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU )
    which = discreteSelect;
else if( integratedSelect >= 0 )
    which = integratedSelect;
else if( discreteSelect >= 0 )
    which = discreteSelect;
else
    fprintf( FpDebug, "Could not select a Physical Device
" );
return VK_SHOULD_EXIT;

PhysicalDevice = physicalDevices[which];

Asking About the Physical Device’s Features

PhysicalDevice Features:

API version: 4194360
Pipeline Cache Size: 13
Physical Device Type: 2 = (Discrete GPU)
Device ID: 0x1b06
Vendor ID: 0x10de

Asking About the Physical Device’s Different Memories

Memory Types:

11 Memory Types:
Memory 0:
Memory 1:
Memory 2:
Memory 3:
Memory 4:
Memory 5:
Memory 6:
Memory 7:
Memory 8:
Memory 9:
Memory 10:

Memory Heaps:
Heap 0:
Heap 1:

Asking About the Physical Device’s Queue Families

Queue Families:

Found %d Queue Families:

10}
Here's What I Got

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