### Physical Devices

- **Application**
- **Instance**
- **Physical Device**
- **Logical Device**

**Vulkan: Overall Block Diagram**

**Vulkan: a More Typical (and Simplified) Block Diagram**

```cpp
uint32_t count;
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT (VkPhysicalDevice *)nullptr );
VkPhysicalDevice * physicalDevices = new VkPhysicalDevice[ count ];
result = vkEnumeratePhysicalDevices( Instance,     &count,          physicalDevices );
```

- **How many total there are**
- **Where to put them**

This way of querying information is a recurring OpenCL and Vulkan pattern (get used to it):
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;
}
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] );
Here's What I Got

vkEnumeratePhysicalDevices:
Device 0:
  API version: 4194360
  Driver version: 4194360
  Vendor ID: 0x10de
  Device ID: 0x1b06
  Physical Device Type: 2 = (Discrete GPU)
Pipeline Cache Size: 13
Device #0 selected ("GeForce GTX 1080 Ti")

Physical Device Features:
  geometryShader = 1
tessellationShader = 1
multiDrawIndirect = 1
wideLines = 1
largePoints = 1
multiViewport = 1
occlusionQueryPrecise = 1
pipelineStatisticsQuery = 1
shaderFloat64 = 1
shaderInt64 = 0
shaderInt16 = 0

Here's What I Got

11 Memory Types:
  Memory 0: DeviceLocal
  Memory 1: DeviceLocal
  Memory 2: HostVisible HostCoherent
  Memory 3: HostVisible
  Memory 4: HostCached
  Memory 5: HostCached
  Memory 6: HostCached
  Memory 7: DeviceLocal
  Memory 8: DeviceLocal
  Memory 9: HostVisible HostCoherent
  Memory 10: HostVisible HostCoherent HostCached

2 Memory Heaps:
  Heap 0: size = 0xb7c00000 DeviceLocal
  Heap 1: size = 0xfac00000

Here's What I Got

Asking About the Physical Device's Different Memories

vkGetPhysicalDeviceMemoryProperties
outs:
vkGetPhysicalDeviceMemoryProperties(PhysicalDevice, OUT &vpdmp);
for(unsigned int i = 0; i < vpdmp.memoryTypeCount; i++) {
  VkMemoryType vmt = vpdmp.memoryTypes[i];
  printf("Memory %d: ", i);
  if((vmt.propertyFlags & VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT) != 0)    printf("DeviceLocal");
  if((vmt.propertyFlags & VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT) != 0)    printf("HostVisible");
  if((vmt.propertyFlags & VK_MEMORY_PROPERTY_HOST_COHERENT_BIT) != 0)    printf("HostCoherent");
  if((vmt.propertyFlags & VK_MEMORY_PROPERTY_HOST_CACHED_BIT) != 0)    printf("HostCached");
  printf("\n");
}

Asking About the Physical Device's Queue Families

uint32_t count = -1;
vkGetPhysicalDeviceQueueFamilyProperties(PhysicalDevice, &count, OUT vqfp);
fprintf("Found %d Queue Families:\n", count);
for(unsigned int i = 0; i < count; i++) {
  printf("%d: queueCount = %d  ;   ", i, vqfp[i].queueCount);
  if((vqfp[i].queueFlags & VK_QUEUE_GRAPHICS_BIT) != 0)       printf("Graphics");
  if((vqfp[i].queueFlags & VK_QUEUE_COMPUTE_BIT) != 0)       printf("Compute ");
  if((vqfp[i].queueFlags & VK_QUEUE_TRANSFER_BIT) != 0)       printf("Transfer");
  printf("\n");
}
Here's What I Got

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