## Physical Devices

### Vulkan: Overall Block Diagram

1. **Application** → **Instance** → **Physical Device** → **Logical Device** → **Command Buffer**

### Querying the Number of Physical Devices

```cpp
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.

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;
}
```

### Identifying the Physical Devices

```cpp
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
        return VK_SHOULD_EXIT;
    }
    fprintf( FpDebug, 

Device %2d:
    fprintf( FpDebug, 
	API version: %d
    fprintf( FpDebug, 
	Driver version: %d
    fprintf( FpDebug, 
	Vendor ID: 0x%04x
    fprintf( FpDebug, 
	Device ID: 0x%04x
    fprintf( FpDebug, 
	Physical Device Type: %d =
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU )  f printf( 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 )   f printf( FpDebug, " (Virtual GPU)
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU )           f printf( FpDebug, " (CPU)
    fprintf( FpDebug, 
	Device Name: %s
    fprintf( FpDebug, 
	Pipeline Cache Size: %d
```

### Which Physical Device to Use, I

```cpp
Vulkan: Identifying the Physical Devices

Vulkan: a More Typical (and Simplified) Block Diagram

Application
    Instance
        Physical Device
            Logical Device
                Command Buffer

Vulkan: Overall Block Diagram

Application
    Instance
        Physical Device
            Logical Device
                Command Buffer

Which Physical Device to Use, I

1. **App** → **Instance** → **Physical Device** → **Logical Device** → **Command Buffer**

## Physical Devices

### Querying the Number of Physical Devices

```cpp
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;
}
```

### Identifying the Physical Devices

```cpp
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
        return VK_SHOULD_EXIT;
    }
    fprintf( FpDebug, 

Device %2d:
    fprintf( FpDebug, 
	API version: %d
    fprintf( FpDebug, 
	Driver version: %d
    fprintf( FpDebug, 
	Vendor ID: 0x%04x
    fprintf( FpDebug, 
	Device ID: 0x%04x
    fprintf( FpDebug, 
	Physical Device Type: %d =
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU )  f printf( 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 )   f printf( FpDebug, " (Virtual GPU)
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU )           f printf( FpDebug, " (CPU)
    fprintf( FpDebug, 
	Device Name: %s
    fprintf( FpDebug, 
	Pipeline Cache Size: %d
```
Here's What the Intel HD Graphics 520 Produced

```
VKPhysicalDeviceProperties:
  PhysicalDevice: PhysicalDeviceProperties
  PhysicalDeviceType: VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU
  FeatureFlags: 0
  QueueFlags: 0
  QueueCount: 1
  Luid: 0
  SubGroupSize: 0
  MaxComputeUnits: 1
  MaxWorkGroupSize: 65536
  MaxPerStageWorkGroupInvocations: 65536
  MaxPerStageSharedMemory: 32768
  queueFamilyCount: 1

memoryTypeCount: 11

Memory Type 0:
  propertyFlags:
    VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT
    VK_MEMORY_PROPERTY_HOST_COHERENT_BIT
    VK_MEMORY_PROPERTY_HOST_CACHED_BIT
  set: 0
  size: 0x0
  alignment: 0
  linearSize: 0

Memory Heap Count: 2

Heap 0:
  size: 0xb7c00000
  flags:
    VK_MEMORY_HEAP_DEVICE_LOCAL_BIT
  propertyFlags:
    VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT
    VK_MEMORY_PROPERTY_LAZILY_ALLOCATED_BIT
  linearSize: 0

Heap 1:
  size: 0x0
  flags:
    VK_MEMORY_HEAP_HOST_VISIBLE_BIT
    VK_MEMORY_HEAP_HOST_COHERENT_BIT
  propertyFlags:
    VK_MEMORY_PROPERTY_IDLE_DEVICE_LOCAL_BIT
    VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT
    VK_MEMORY_PROPERTY_HOST_COHERENT_BIT
    VK_MEMORY_PROPERTY_HOST_CACHED_BIT
  linearSize: 0
```

Here's What the NVIDIA 1080ti Produced

```
VKPhysicalDeviceProperties:
  PhysicalDevice: PhysicalDeviceProperties
  PhysicalDeviceType: VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU
  FeatureFlags: 0
  QueueFlags: 0
  QueueCount: 1
  Luid: 0
  SubGroupSize: 0
  MaxComputeUnits: 64
  MaxWorkGroupSize: 65536
  MaxPerStageWorkGroupInvocations: 65536
  MaxPerStageSharedMemory: 32768
  queueFamilyCount: 1

memoryTypeCount: 11

Memory Type 0:
  propertyFlags:
    VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT
    VK_MEMORY_PROPERTY_HOST_COHERENT_BIT
    VK_MEMORY_PROPERTY_HOST_CACHED_BIT
  set: 0
  size: 0x0
  alignment: 0
  linearSize: 0

Memory Heap Count: 2

Heap 0:
  size: 0x0
  flags:
    VK_MEMORY_HEAP_DEVICE_LOCAL_BIT
    VK_MEMORY_HEAP_DEVICE_LOCAL_BIT
  propertyFlags:
    VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT
    VK_MEMORY_PROPERTY_LAZILY_ALLOCATED_BIT
  linearSize: 0

Heap 1:
  size: 0x0
  flags:
    VK_MEMORY_HEAP_HOST_VISIBLE_BIT
    VK_MEMORY_HEAP_HOST_COHERENT_BIT
  propertyFlags:
    VK_MEMORY_PROPERTY_IDLE_DEVICE_LOCAL_BIT
    VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT
    VK_MEMORY_PROPERTY_HOST_COHERENT_BIT
    VK_MEMORY_PROPERTY_HOST_CACHED_BIT
  linearSize: 0
```

Which Physical Device to Use, II

```
int which = -1;
else
  which = integratedSelect;
else if( discreteSelect >= 0 )
  discreteSelect = i;
```

Here's What I Got

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

2 Memory Heaps:
  Heap 0: size = 0x1f00000
  Heap 1: size = 0x1f00000
```
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, "\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");
}

Asking About the Physical Device’s Queue Families

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

Here’s What I Got