



Vulkan.
Physical Devices

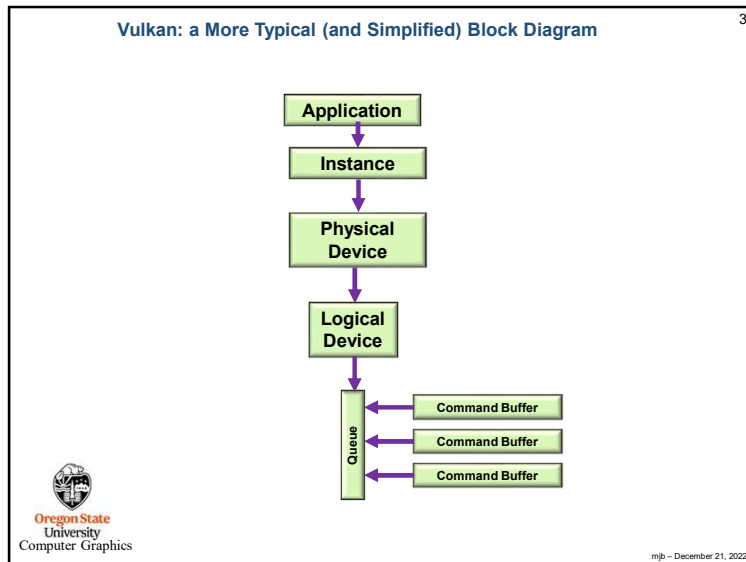
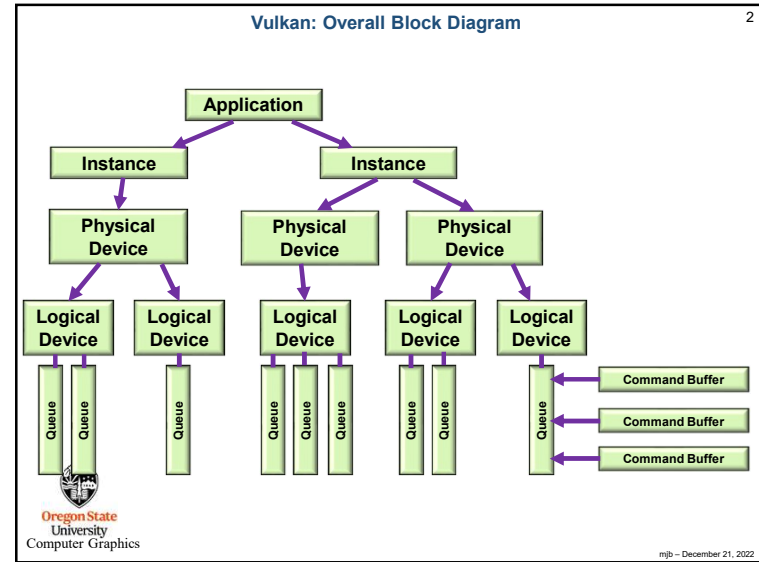


Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

CC BY-NC-ND
This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



PhysicalDevices.pptx mjb - December 21, 2022



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      Where to
there are           put them
result = vkEnumeratePhysicalDevices( Instance, &count, nullptr );
result = vkEnumeratePhysicalDevices( Instance, &count, physicalDevices );
  
```

Oregon State University Computer Graphics mjb - December 21, 2022

Vulkan: Identifying the Physical Devices 5

```

graph TD
    Application --> Instance
    Instance --> PhysicalDevice[Physical Device]
    PhysicalDevice --> LogicalDevice[Logical Device]
    LogicalDevice --> Device
    
```

```

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\n" );
    return VK_SHOULD_EXIT;
}

fprintf( FpDebug, "\n%d physical devices found.\n", 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\n", PhysicalDeviceCount );
    return VK_SHOULD_EXIT;
}
            
```

Oregon State University
Computer Graphics

mjb - December 21, 2022

Which Physical Device to Use, I 6

```

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\n", i );
        return VK_SHOULD_EXIT;
    }

    fprintf( FpDebug, "\n\nDevice %2d:\n", i );
    fprintf( FpDebug, "\tAPI version: %d\n", vpdp.apiVersion );
    fprintf( FpDebug, "\tDriver version: %d\n", vpdp.driverVersion );
    fprintf( FpDebug, "\tVendor ID: 0x%04x\n", vpdp.vendorID );
    fprintf( FpDebug, "\tDevice ID: 0x%04x\n", vpdp.deviceID );
    fprintf( FpDebug, "\tPhysical Device Type: %d =", vpdp.deviceType );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU ) fprintf( FpDebug, " (Discrete GPU)\n" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU ) fprintf( FpDebug, " (Integrated GPU)\n" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_VIRTUAL_GPU ) fprintf( FpDebug, " (Virtual GPU)\n" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU ) fprintf( FpDebug, " (CPU)\n" );
    fprintf( FpDebug, "\tDevice Name: %s\n", vpdp.deviceName );
    fprintf( FpDebug, "\tPipeline Cache Size: %d\n", vpdp.pipelineCacheUID[0] );
}
            
```

Oregon State University
Computer Graphics

mjb - December 21, 2022

Which Physical Device to Use, II 7

```

// need some logical here to decide which physical device to select:

if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU )
    discreteSelect = i;

if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU )
    integratedSelect = i;
}

int which = -1;
if( discreteSelect >= 0 )
{
    which = discreteSelect;
    PhysicalDevice = physicalDevices[which];
}
else if( integratedSelect >= 0 )
{
    which = integratedSelect;
    PhysicalDevice = physicalDevices[which];
}
else
{
    fprintf( FpDebug, "Could not select a Physical Device\n" );
    return VK_SHOULD_EXIT;
}
            
```

Oregon State University
Computer Graphics

mjb - December 21, 2022

Asking About the Physical Device's Features 8

```

VkPhysicalDeviceProperties PhysicalDeviceFeatures;
vkGetPhysicalDeviceFeatures( IN PhysicalDevice, OUT &PhysicalDeviceFeatures );

fprintf( FpDebug, "\nPhysical Device Features:\n" );
fprintf( FpDebug, "geometryShader = %2d\n", PhysicalDeviceFeatures.geometryShader );
fprintf( FpDebug, "tessellationShader = %2d\n", PhysicalDeviceFeatures.tessellationShader );
fprintf( FpDebug, "multiDrawIndirect = %2d\n", PhysicalDeviceFeatures.multiDrawIndirect );
fprintf( FpDebug, "wideLines = %2d\n", PhysicalDeviceFeatures.wideLines );
fprintf( FpDebug, "largePoints = %2d\n", PhysicalDeviceFeatures.largePoints );
fprintf( FpDebug, "multiViewport = %2d\n", PhysicalDeviceFeatures.multiViewport );
fprintf( FpDebug, "occlusionQueryPrecise = %2d\n", PhysicalDeviceFeatures.occlusionQueryPrecise );
fprintf( FpDebug, "pipelineStatisticsQuery = %2d\n", PhysicalDeviceFeatures.pipelineStatisticsQuery );
fprintf( FpDebug, "shaderFloat64 = %2d\n", PhysicalDeviceFeatures.shaderFloat64 );
fprintf( FpDebug, "shaderInt64 = %2d\n", PhysicalDeviceFeatures.shaderInt64 );
fprintf( FpDebug, "shaderInt16 = %2d\n", PhysicalDeviceFeatures.shaderInt16 );
            
```

Oregon State University
Computer Graphics


mjb - December 21, 2022

Here's What the NVIDIA A6000 Produced

Init03PhysicalDeviceAndGetQueueFamilyProperties

Device 0:
 API version: 4206797
 Driver version: 4206797
 Vendor ID: 0x10de
 Device ID: 0x2230
 Physical Device Type: 2 = (Discrete GPU)
 Device Name: NVIDIA RTX A6000
 Pipeline Cache Size: 72
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




mjb - December 21, 2022

Here's What the Intel HD Graphics 520 Produced

Init03PhysicalDeviceAndGetQueueFamilyProperties

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



mjb - December 21, 2022


Asking About the Physical Device's Different Memories

```

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

fprintf( FpDebug, "In%d Memory Types:\n", 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, "In%d Memory Heaps:\n", vpdmp.memoryHeapCount );
for( unsigned int i = 0; i < vpdmp.memoryHeapCount; i++ )
{
    fprintf( FpDebug, "Heap %d: ", i );
    VkMemoryHeap vmh = vpdmp.memoryHeaps[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" );
}
    
```




mjb - December 21, 2022

Here's What I Got on the A6000's

6 Memory Types:
 Memory 0: DeviceLocal
 Memory 1: HostVisible HostCoherent
 Memory 2: HostVisible HostCoherent HostCached
 Memory 3: HostVisible HostCoherent HostCached
 Memory 4: DeviceLocal HostVisible HostCoherent
 Memory 5: DeviceLocal

4 Memory Heaps:
 Heap 0: size = 0xdbb00000 DeviceLocal
 Heap 1: size = 0xfd504000
 Heap 2: size = 0x0d600000 DeviceLocal
 Heap 3: size = 0x02000000 DeviceLocal



mjb - December 21, 2022

Asking About the Physical Device's Queue Families

13

```

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

```



mjb - December 21, 2022

Here's What I Got on the A6000's

14

```

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

```



mjb - December 21, 2022