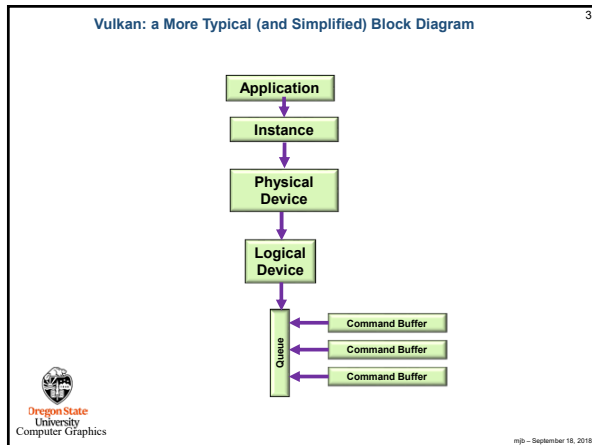
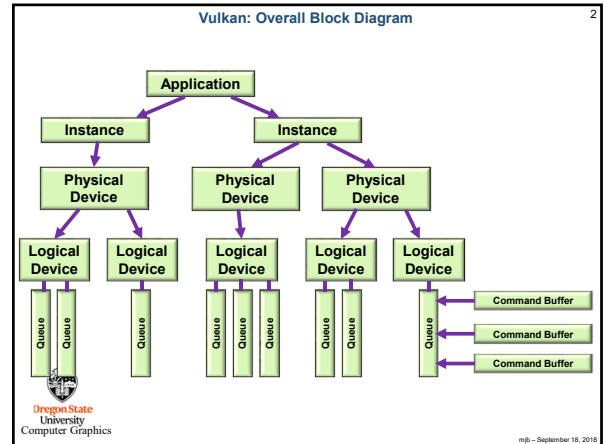


**Vulkan.**  
Physical Devices

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

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

Oregon State University Computer Graphics  
PhysicalDevices.pptx  
mb - September 18, 2018



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

Oregon State University Computer Graphics  
mb - September 18, 2018

### Vulkan: Identifying the 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!\n" );
    return VK_SHOULD_EXIT;
}
fprintf( FpDebug, "%i 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 %i physical devices!\n", PhysicalDeviceCount );
    return VK_SHOULD_EXIT;
}
  
```

Oregon State University Computer Graphics  
mb - September 18, 2018

### Which Physical Device to Use, I

```

int discreteSelect = -1;
int integratedSelect = -1;
for( unsigned int i = 0; i < PhysicalDeviceCount; i++ )
{
    VkPhysicalDeviceProperties vppd;
    vkGetPhysicalDeviceProperties( IN physicalDevices[ i ], OUT &vppd );
    if ( result != VK_SUCCESS )
    {
        fprintf( FpDebug, "Could not get the physical device properties of device %d!\n", i );
        return VK_SHOULD_EXIT;
    }

    fprintf( FpDebug, " %i\nDevice %2d!\n", i );
    fprintf( FpDebug, "API version: %d\n", vppd.apiVersion );
    fprintf( FpDebug, "Driver version: %d\n", vppd.driverVersion );
    fprintf( FpDebug, "Vendor ID: 0x%04x\n", vppd.vendorID );
    fprintf( FpDebug, "Device ID: 0x%04x\n", vppd.deviceID );
    fprintf( FpDebug, "Physical Device Type: %d\n", vppd.deviceType );
    if ( vppd.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU ) fprintf( FpDebug, " (Discrete GPU)\n" );
    if ( vppd.deviceType == VK_PHYSICAL_DEVICE_TYPE_VIRTUAL_GPU ) fprintf( FpDebug, " (Integrated GPU)\n" );
    if ( vppd.deviceType == VK_PHYSICAL_DEVICE_TYPE_VIRTUAL_GPU ) fprintf( FpDebug, " (Virtual GPU)\n" );
    if ( vppd.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU ) fprintf( FpDebug, " (CPU)\n" );
    fprintf( FpDebug, "Device Name: %s\n", vppd.deviceName );
    fprintf( FpDebug, "Pipeline Cache Size: %d\n", vppd.pipelineCacheUID );
}
  
```


Oregon State University Computer Graphics  
mb - September 18, 2018

### Asking About the Physical Device's Features

```

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 );
    
```



mj - September 18, 2018


### Here's What the NVIDIA 1080ti Produced

```

vkEnumeratePhysicalDevices:

Device 0:
  API version: 4194360
  Driver version: 4194360
  Vendor ID: 0x10de
  Device ID: 0x1b06
  Physical Device Type: 2 = (Discrete GPU)
  Device Name: GeForce GTX 1080 Ti
  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 = 1
  shaderInt16 = 0
    
```



mj - September 18, 2018


### Here's What the Intel HD Graphics 520 Produced

```

vkEnumeratePhysicalDevices:

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
    
```



mj - September 18, 2018


### Which Physical Device to Use, II

```

// need some logical here to decide which physical device to select:
if( vpd.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU )
  discreteSelect = i;

if( vpd.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;
}
    
```



mj - September 18, 2018


### Asking About the Physical Device's Different Memories

```

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

fprintf( FpDebug, "\nId 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, "\nId Memory Heaps:\n", vpdmp.memoryHeapCount );
for( unsigned int i = 0; i < vpdmp.memoryHeapCount; i++ )
{
  fprintf( FpDebug, "Heap %2d: ", i );
  VkMemoryHeap vmt = vpdmp.memoryHeaps[i];
  fprintf( FpDebug, " size = %0x%08bc", (unsigned long int)vmt.size );
  if( ( vmt.flags & VK_MEMORY_HEAP_DEVICE_LOCAL_BIT ) != 0 ) fprintf( FpDebug, " DeviceLocal" ); // only one in use
  fprintf( FpDebug, "\n" );
}
    
```




mj - September 18, 2018

### Here's What I Got

```

11 Memory Types:
Memory 0:
Memory 1:
Memory 2:
Memory 3:
Memory 4:
Memory 5:
Memory 6:
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
    
```




mj - September 18, 2018

### 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, "%d: queueCount = %d ; ", 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" );
}
```




Oregon State University  
Computer Graphics

mjb - September 18, 2018

### Here's What I Got 14

Found 3 Queue Families:

0: queueCount = 16 ;	Graphics Compute Transfer
1: queueCount = 1 ;	Transfer
2: queueCount = 8 ;	Compute



Oregon State University  
Computer Graphics

mjb - September 18, 2018