Vulkan: Overall Block Diagram

Application
Instance
Physical Device
Logical Device

Queue
Command Buffer

Vulkan: a More Typical (and Simplified) Block Diagram

Application
Instance
Physical Device
Logical Device

Command Buffer

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

Querying the Number of Physical Devices

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

result = vkEnumeratePhysicalDevices( Instance, &count, nullptr );
result = vkEnumeratePhysicalDevices( Instance, &count, physicalDevices );
**Vulkan: Identifying the Physical Devices**

```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, \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
", PhysicalDeviceCount );
    return VK_SHOULD_EXIT;
}
```

**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 );
    if( result != VK_SUCCESS ) {
        fprintf( FpDebug, "Could not get the physical device properties of device %d
", i );
        return VK_SHOULD_EXIT;
    }
    fprintf( FpDebug, \n\Device %2d:
", i );
    fprintf( FpDebug, \tAPI version: %d\n", vpdp.apiVersion );
    fprintf( FpDebug, \tDriver version: %d\n", vpdp.apiVersion );
    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 )  f printf( 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 )   f printf( FpDebug, " (Virtual GPU)\n" );
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU )           f printf( FpDebug, " (CPU)\n" );
    fprintf( FpDebug, \tDevice Name: %s\n", vpdp.deviceName );
    fprintf( FpDebug, \tPipeline Cache Size: %d\n", vpdp.pipelineCacheSize );
}
```

**Asking About the Physical Device's Features**

```c
VkPhysicalDeviceProperties PhysicalDeviceFeatures;
vkGetPhysicalDeviceProperties( IN PhysicalDevice, OUT &PhysicalDeviceFeatures );

fprintf(FpDebug, \nPhysical Device Features:
" );
fprintf(FpDebug, \ngeometryShader = %2d\n", PhysicalDeviceFeatures.geometryShader );
fprintf(FpDebug, \ntessellationShader = %2d\n", PhysicalDeviceFeatures.tessellationShader );
fprintf(FpDebug, \nmultiDrawIndirect = %2d\n", PhysicalDeviceFeatures.multiDrawIndirect );
fprintf(FpDebug, \nwideLines = %2d\n", PhysicalDeviceFeatures.wideLines );
fprintf(FpDebug, \nlargePoints = %2d\n", PhysicalDeviceFeatures.largePoints );
fprintf(FpDebug, \nocclusionQueryPrecise = %2d\n", PhysicalDeviceFeatures.occlusionQueryPrecise );
fprintf(FpDebug, \npipelineStatisticsQuery = %2d\n", PhysicalDeviceFeatures.pipelineStatisticsQuery );
fprintf(FpDebug, \nshaderFloat64 = %2d\n", PhysicalDeviceFeatures.shaderFloat64 );
fprintf(FpDebug, \nshaderInt64 = %2d\n", PhysicalDeviceFeatures.shaderInt64 );
fprintf(FpDebug, \nshaderInt16 = %2d\n", PhysicalDeviceFeatures.shaderInt16 );
```

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

```c
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
    occlusionQueryPrecise = 1
    pipelineStatisticsQuery = 1
    shaderFloat64 = 1
    shaderInt64 = 1
    shaderInt16 = 0
```
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
Here’s What the Intel HD Graphics 520 Produced

// need some logical here to decide which physical device to select:
if (vpdp.deviceType == VK_PHYSICALDEVICE_TYPE_DISCRETE_GPU)
  discreteSelect = i;
else if (vpdp.deviceType == VK_PHYSICALDEVICE_TYPE_INTEGRATED_GPU)
  integratedSelect = i;
else
  fprintf(FpDebug, "Could not select a Physical Device\n");
  return VK_SHOULD_EXIT;
Which Physical Device to Use, II

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

Here’s What I Got

11 Memory Types:
Memory 0: 0x0a70d00000 Device Local
Memory 1: 0x0a70d00000 Device Local
Memory 2: 0x0a70d00000 Device Local
Memory 3: 0x0a70d00000 Device Local
Memory 4: 0x0a70d00000 Device Local
Memory 5: 0x0a70d00000 Device Local
Memory 6: 0x0a70d00000 Device Local
Memory 7: 0x0a70d00000 Device Local
Memory 8: 0x0a70d00000 Device Local
Memory 9: 0x0a70d00000 Device Local
Memory 10: 0x0a70d00000 Device Local

2 Memory Heaps:
Heap 0: size = 0x0a70d00000 Device Local
Heap 1: size = 0x0a70d00000 Device Local

Asking About the Physical Device’s Different Memories

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

1 Memory Types:
Memory 0:  Device Local
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, "%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");
}

Here's What I Got

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