1/3/2020

Vulkan: Overall Block Diagram

Application

Instance

Physical Device

Logical Device

Command Buffer

Vulkan: Querying the Number of 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
" );
    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;
}
uint32_t count;
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT (VkPhysicalDevice *)nullptr );
VkPhysicalDevice * physicalDevices = new VkPhysicalDevice[ count ];
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT physicalDevices );

Vulkan: Identifying the Physical Devices

int discreteSelect = -1;
int integratedSelect = -1;
for( unsigned int i = 0; i < PhysicalDeviceCount; i++ ) {
    VkPhysicalDeviceProperties vpdp;
vkGetPhysicalDeviceProperties( IN physicalDevices[i], OUT &vpdp );
    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] );
}

Vulkan: Which Physical Device to Use, I

WhichPhysicalDevice() {
    VkPhysicalDeviceProperties vendor;
vkGetPhysicalDeviceProperties( PhysicalDevice[0], OUT &vendor );
    if( vendor.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU ) {
        fprintf( FpDebug, "Device is a Discrete GPU
" );
    } else if( vendor.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU ) {
        fprintf( FpDebug, "Device is an Integrated GPU
" );
    } else if( vendor.deviceType == VK_PHYSICAL_DEVICE_TYPE_VIRTUAL_GPU ) {
        fprintf( FpDebug, "Device is a Virtual GPU
" );
    } else if( vendor.deviceType == VK_PHYSICAL_DEVICE_TYPE_CPU ) {
        fprintf( FpDebug, "Device is a CPU
" );
    } else {
        fprintf( FpDebug, "Device is not a GPU or CPU
" );
    }
}
### Asking About the Physical Device’s Features

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

if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU ){
    // need some logical here to decide which physical device to select:
    if( vpdp.deviceType == VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU ){
        // do something
        vkEnumeratePhysicalDevices:
        API version: 4194360
        Driver version: 4194360
        Device Name: Intel(R) HD Graphics 520
        Physical Device: Intel(R) HD Graphics 520
        Pipeline Cache Size: 213
        Device #0 selected ('Intel(R) HD Graphics 520')
    }
}

Here’s What the Intel HD Graphics 520 Produced

- shaderInt16 = 1
- shaderInt64 = 1
- shaderFloat64 = 1
- pipelineStatisticsQuery = 1
- occlusionQueryPrecise = 1
- multiViewport = 1
- largePoints = 1
- wideLines = 1
- multiDrawIndirect = 1
- tessellationShader = 1
- geometryShader = 1

Physical Device Features:

Device Features:

- Memory 10: HostVisible HostCoherent HostCached
- Memory 9: HostVisible HostCoherent HostCached
- Memory 8: HostVisible HostCoherent HostCached
- Memory 7: HostVisible HostCoherent HostCached
- Memory 6: HostVisible HostCoherent HostCached
- Memory 5: HostVisible HostCoherent HostCached
- Memory 4: HostVisible HostCoherent HostCached
- Memory 3: HostVisible HostCoherent HostCached
- Memory 2: HostVisible HostCoherent HostCached
- Memory 1: HostVisible HostCoherent HostCached
- Memory 0: HostVisible HostCoherent HostCached
```

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

```c
vkGetMemoryProperties(IN PhysicalDevice, OUT MemoryProperties);
VkPhysicalDeviceMemoryProperties PhysicalDeviceMemoryProperties;

Memory Properties:

- Memory 10: HostVisible HostCoherent HostCached
- Memory 9: HostVisible HostCoherent HostCached
- Memory 8: HostVisible HostCoherent HostCached
- Memory 7: HostVisible HostCoherent HostCached
- Memory 6: HostVisible HostCoherent HostCached
- Memory 5: HostVisible HostCoherent HostCached
- Memory 4: HostVisible HostCoherent HostCached
- Memory 3: HostVisible HostCoherent HostCached
- Memory 2: HostVisible HostCoherent HostCached
- Memory 1: HostVisible HostCoherent HostCached
- Memory 0: HostVisible HostCoherent HostCached
```

### Here’s What I Got

11 Memory Types:

- Memory 0: DeviceLocal
- Memory 1: DeviceLocal
- Memory 2: DeviceLocal
- Memory 3: DeviceLocal
- Memory 4: DeviceLocal
- Memory 5: DeviceLocal
- Memory 6: DeviceLocal
- Memory 7: DeviceLocal
- Memory 8: DeviceLocal
- Memory 9: DeviceLocal
- Memory 10: DeviceLocal

2 Memory Heaps:

Heap 0: size = 0x00000000
Heap 1: size = 0x00000000
uint32_t count = -1;
vkGetPhysicalDeviceQueueFamilyProperties(IN PhysicalDevice, &count, OUT (VkQueueFamilyProperties *)nullptr);
fprintf(FpDebug, "Found %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");
}

Asking About the Physical Device's Queue Families

Here's What I Got

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