Getting Information Back from the Graphics System

There are 3 types of Queries: **Occlusion, Pipeline Statistics, and Timestamp**

Vulkan requires you to first setup “Query Pools”, some for each specific type

This indicates that Vulkan thinks that Queries are time-consuming (relatively) to setup, and thus better to set them up in program-setup than in program-runtime
Setting up Query Pools

```c
VkQueryPoolCreateInfo vqpci;
vqpci.sType = VK_STRUCTURE_TYPE_QUERY_POOL_CREATE_INFO;
vqpci.pNext = nullptr;
vqpci.flags = 0;
vqpci.queryType = << one of: >>
    VK_QUERY_TYPE_OCCLUSION
    VK_QUERY_TYPE_PIPELINE_STATISTICS
    VK_QUERY_TYPE_TIMESTAMP
vqpci.queryCount = 3;
vqpci.pipelineStatistics = 0;  // bitmask of what stats you are querying for if you
                                // are doing a pipeline statistics query
VkQueryPool occlusionQueryPool;
result = vkCreateQueryPool( LogicalDevice, IN &vqpci, PALLOCATOR, OUT &occlusionQueryPool );

VkQueryPool statisticsQueryPool;
result = vkCreateQueryPool( LogicalDevice, IN &vqpci, PALLOCATOR, OUT &statisticsQueryPool );

VkQueryPool timestampQueryPool;
result = vkCreateQueryPool( LogicalDevice, IN &vqpci, PALLOCATOR, OUT &timestampQueryPool );
```

Resetting, Filling, and Examining a Query Pool

```c
vkCmdResetQueryPool( CommandBuffer, occlusionQueryPool, 0, 3 );
vkCmdBeginQuery( CommandBuffer, occlusionQueryPool, 0, VK_QUERY_CONTROL_PRECISE_BIT );
...
vkCmdEndQuery( CommandBuffer, occlusionQueryPool, 0 );
result = vkGetQueryPoolResults( LogicalDevice, occlusionQueryPool, 0, 1, DATASIZE, data, stride, flags );
    // VK_QUERY_RESULT_64_BIT
    // VK_QUERY_RESULT_WAIT_BIT
    // VK_QUERY_RESULT_WITH_AVAILABILITY_BIT
    // VK_QUERY_RESULT_PARTIAL_BIT
    // stride is # of bytes in between each result
vkCmdCopyQueryPoolResults( CommandBuffer, occlusionQueryPool, 0, 1, buffer, 0, stride, flags );
    // VK_QUERY_RESULT_64_BIT
    // VK_QUERY_RESULT_WAIT_BIT
    // VK_QUERY_RESULT_WITH_AVAILABILITY_BIT
    // VK_QUERY_RESULT_PARTIAL_BIT
    // stride is # of bytes in between each result
```
Occlusion Queries count the number of fragments drawn between the vkCmdBeginQuery and the vkCmdEndQuery that pass both the Depth and Stencil tests. This is commonly used to see what level-of-detail should be used when drawing a complicated object.

Some hints:

- Don’t draw the whole scene – just draw the object you are interested in
- Don’t draw the whole object – just draw a simple bounding volume at least as big as the object
- Don’t draw the whole bounding volume – cull away the back faces (two reasons: time and correctness)
- Don’t draw the colors – just draw the depths (especially if the fragment shader is time-consuming)

```cpp
uint32_t fragmentCount;
result = vkGetQueryPoolResults( LogicalDevice, occlusionQueryPool, 0, 1,
                                 sizeof(uint32_t), &fragmentCount, 0, VK_QUERY_RESULT_WAIT_BIT );

vkCmdCopyQueryPoolResults( CommandBuffer, occlusionQueryPool, 0, 1,
                           buffer, 0, 0, VK_QUERY_RESULT_WAIT_BIT );
```

Pipeline Statistics Queries count how many of various things get done between the vkCmdBeginQuery and the vkCmdEndQuery.

```cpp
uint32_t counts[NUMSTATS];
result = vkGetQueryPoolResults( LogicalDevice, statisticsQueryPool, 0, 1,
                                 NUM_STATS*sizeof(uint32_t), counts, 0, VK_QUERY_RESULT_WAIT_BIT );

vkCmdCopyQueryPoolResults( CommandBuffer, occlusionQueryPool, 0, 1,
                           buffer, 0, 0, VK_QUERY_RESULT_WAIT_BIT );
```
Timestamp Query

Timestamp Queries count how many nanoseconds of time elapsed between the vkCmdBeginQuery and the vkCmdEndQuery.

```c
uint64_t nanosecondsCount;
result = vkGetQueryPoolResults( LogicalDevice, timestampQueryPool, 0, 1,
    sizeof(uint64_t), &nanosecondsCount, 0,
    VK_QUERY_RESULT_64_BIT | VK_QUERY_RESULT_WAIT_BIT);
```

```c
vkCmdCopyQueryPoolResults( CommandBuffer, timestampQueryPool, 0, 1,
    buffer, 0, 0,
    VK_QUERY_RESULT_64_BIT | VK_QUERY_RESULT_WAIT_BIT );
```

Timestamp Query

The vkCmdWriteTimeStamp( ) function produces the time between when this function is called and when the first thing reaches the specified pipeline stage.

Even though the stages are “bits”, you are supposed to only specify one of them

```c
vkCmdWriteTimeStamp( CommandBuffer, pipelineStages, timestampQueryPool, 0 );
```

```
VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT
VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT
VK_PIPELINE_STAGE_VERTEX_INPUT_BIT
VK_PIPELINE_STAGE_VERTEX_SHADER_BIT
VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT,
VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT
VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT,
VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT
VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT
VK_PIPELINE_STAGE_TRANSFER_BIT
VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT
VK_PIPELINE_STAGE_HOST_BIT
```