Getting Information Back from the Graphics System

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(s) you are interested in.
- Don’t draw the whole object – just draw a simple bounding volume at least as big as the object(s).
- Don’t draw the whole bounding volume – cut 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).

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

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

Vulkan requires you to first setup “Query Pools”, one 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.

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

Setting up Query Pools

```c
VkQueryPoolCreateInfo vqpci;
vqpci.sType = VK_STRUCTURE_TYPE_QUERY_POOL_CREATE_INFO;
vqpci.pipelineStatistics = 0; // bitmask of what stats you are querying for if you are doing a pipeline statistics query
vqpci.queryCount = 1;
vqpci.queryType = VK_QUERY_TYPE_PIPELINE_STATISTICS; // Pipeline Statistics Query
vqpci.flags = 0;
vqpci.pNext = nullptr;
result = vkCreateQueryPool( LogicalDevice, &vqpci, PALLOCATOR, &queryPool );
```

```c
#define DATASIZE 128
uint32_t counts[NUM_STATS];
result = vkGetQueryPoolResults( LogicalDevice, statisticsQueryPool, 0, 1, DATASIZE*sizeof(uint32_t), &counts, 0, VK_QUERY_RESULT_WAIT_BIT );
```

Resetting, Filling, and Examining a Query Pool

```c
vkCmdResetQueryPool( CommandBuffer, occlusionQueryPool, 0, 1 );
vkCmdBeginQuery( CommandBuffer, occlusionQueryPool, 0, 1, VK_QUERY_CONTROL_PRECISE_BIT );
vkCmdEndQuery( CommandBuffer, occlusionQueryPool, 0 );
```

```c
result = vkGetQueryPoolResults( LogicalDevice, occlusionQueryPool, 0, 1, DATASIZE*sizeof(uint32_t), &counts, 0, VK_QUERY_RESULT_WAIT_BIT );
```

 Pipeline Statistics Query

Occlusion Query

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

```c
uint32_t counts[NUM_STATS];
result = vkGetQueryPoolResults( LogicalDevice, statisticsQueryPool, 0, 1, DATASIZE*sizeof(uint32_t), &counts, 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);
```

### 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, not "or" multiple ones together.

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

### Pipeline Stages

- VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT
- VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
- VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT
- VK_PIPELINE_STAGE_HOST_BIT
- VK_PROCESSING_STAGE_BIT
- VK_QUERY_RESULT_BIT