### Getting Information Back from the Graphics System

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 – 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).

#### Occlusion Query

Occlusion Queries count the number of fragments drawn between the `vkCmdBeginQuery` and the `vkCmdEndQuery`.

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 – 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).

#### Pipeline Statistics Query

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

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

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

```c
// 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
```