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

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

VkQueryPoolCreateInfo vqpci;

vkQueryPool occlusionQueryPool;
result = vkCreateQueryPool( LogicalDevice, IN &vqpci, PALLOCATOR, OUT &occlusionQueryPool );

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 );

vkCmdCopyQueryPoolResults( CommandBuffer, occlusionQueryPool, 0, buffer, 0, stride, flags );

Resetting, Filling, and Examining a Query Pool

vkQueryPoolPipelineStatistics bitmask;

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 );

vkCmdCopyQueryPoolResults( CommandBuffer, occlusionQueryPool, 0, buffer, 0, stride, flags );
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[NUM_STATS];
result = vkGetQueryPoolResults( LogicalDevice, statisticsQueryPool, 0, 1,
sizeof(uint32_t) * NUM_STATS, counts, 0, VK_QUERY_RESULT_WAIT_BIT );
vkCmdCopyQueryPoolResults( CommandBuffer, occlusionQueryPool, 0, 1,
buffer, 0, 0, VK_QUERY_RESULT_WAIT_BIT );
```

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

```cpp
uint64_t nanosecondsCount;
result = vkGetQueryPoolResults( LogicalDevice, timestampQueryPool, 0, 1,
sizeof(uint64_t), &nanosecondsCount, 0, VK_QUERY_RESULT_64_BIT | VK_QUERY_RESULT_WAIT_BIT );
vkCmdCopyQueryPoolResults( CommandBuffer, timestampQueryPool, 0, 1,
buffer, 0, 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.

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