OpenCL Events

An event is an object that communicates the status of OpenCL commands

From the OpenCL Notes:
11. Enqueue the Kernel Object for Execution

Creating an Event

Waiting for Events from Previously-Executed Kernels

Creating an Execution Graph Structure
Creating the Full Execution Graph Structure

```
cl_event waitKernelA, waitKernelB;
cl_event dependenciesAB[2];
dependenciesAB[0] = waitKernelA;
dependenciesAB[1] = waitKernelB;
cl_event dependenciesCD[2];
dependenciesCD[0] = waitKernelC;
dependenciesCD[1] = waitKernelD;
```

```
status = clEnqueueNDRangeKernel(cmdQueue, kernelA, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, &waitKernelA);
status = clEnqueueNDRangeKernel(cmdQueue, kernelB, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, &waitKernelB);
status = clEnqueueNDRangeKernel(cmdQueue, kernelC, 1, NULL, globalWorkSize, localWorkSize, 2, dependenciesAB, &waitKernelC);
status = clEnqueueNDRangeKernel(cmdQueue, kernelD, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, &waitKernelD);
status = clEnqueueNDRangeKernel(cmdQueue, kernelE, 1, NULL, globalWorkSize, localWorkSize, 2, dependenciesCD, NULL);
```

Waiting for One Event

```
cl_event waitKernelA, waitKernelB;
```

```
status = clEnqueueNDRangeKernel(cmdQueue, kernelA, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, &waitKernelA);
status = clEnqueueNDRangeKernel(cmdQueue, kernelB, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, &waitKernelB);
status = clEnqueueNDRangeKernel(cmdQueue, kernelC, 1, NULL, globalWorkSize, localWorkSize, 1, &waitKernelA, NULL);
```

Placing a Barrier in the Command Queue

```
status = clEnqueueBarrier(cmdQueue);
```

```
Placing an Event Marker in the Command Queue
```

```
void Wait(cl_command_queue queue)
{
    cl_event wait;
    cl_int status;
    status = clEnqueueMarker(queue, &wait);
    if(status == CL_SUCCESS)
        fprintf(stderr, "Wait: clEnqueueMarker failed\n");
    status = clWaitForEvents(1, &wait); // blocks until everything is done
    if(status == CL_SUCCESS)
        fprintf(stderr, "Wait: clWaitForEvents failed\n");
}
```

I Like Synchronizing Things This Way

```
// wait until all queued tasks have taken place:
void Wait(cl_command_queue queue)
{
    cl_event wait;
    cl_int status;
    status = clEnqueueMarker(queue, &wait);
    if(status == CL_SUCCESS)
        fprintf(stderr, "Wait: clEnqueueMarker failed\n");
    status = clWaitForEvents(1, &wait); // blocks until everything is done
    if(status == CL_SUCCESS)
        fprintf(stderr, "Wait: clWaitForEvents failed\n");
}
```
Getting Event Statuses Without Blocking

CL_EVENT_COMMAND_QUEUE
CL_EVENT_CONTEXT
CL_EVENT_COMMAND_TYPE
CL_EVENT_COMMAND_EXECUTION_STATUS

Specify one of these

CL_EVENT_COMMAND_EXECUTION_STATUS
returns one of these

CL_QUEUED
CL_SUBMITTED
CL_RUNNING
CL_COMPLETE

Note that this a nice way to check on event statuses without blocking. Thus, you could put this in a loop and go get some other work done in between calls.