




# OpenCL Events




**Oregon State University**  
Mike Bailey  
mjb@cs.oregonstate.edu

This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)







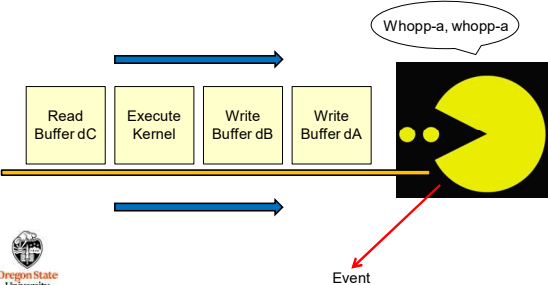
opencl.events.pdf


mjb - March 27, 2021

1

# OpenCL Events

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





Oregon State University  
Computer Graphics

mjb - March 27, 2021

2

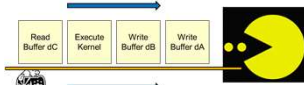
## From the OpenCL Notes: 11. Enqueue the Kernel Object for Execution


```

size_t globalWorkSize[ 3 ] = { NUM_ELEMENT, 1, 1 };
size_t localWorkSize[ 3 ] = { LOCAL_SIZE, 1, 1 };

status = clEnqueueNDRangeKernel( cmdQueue, kernel, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, NULL );

```





Oregon State University  
Computer Graphics

mjb - March 27, 2021

3

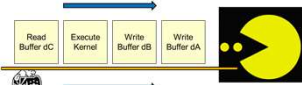
## Creating an Event


```

cl_event waitKernelA, waitKernelB, waitKernelC;

status = clEnqueueNDRangeKernel( cmdQueue, kernel, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, &waitKernelC );

```





Oregon State University  
Computer Graphics

mjb - March 27, 2021

4

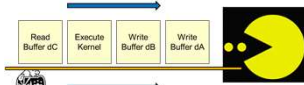
## Waiting for Events from Previously-Executed Kernels

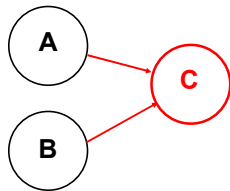
```


cl_event waitKernelA, waitKernelB, waitKernelC;
...
cl_event dependenciesAB[ 2 ];
dependenciesAB[ 0 ] = waitKernelA;
dependenciesAB[ 1 ] = waitKernelB;

status = clEnqueueNDRangeKernel( cmdQueue, kernelC, 1, NULL, globalWorkSize, localWorkSize, 2, dependenciesAB, NULL );

```







Oregon State University  
Computer Graphics

mjb - March 27, 2021

5

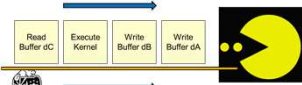
## Creating an Execution Graph Structure

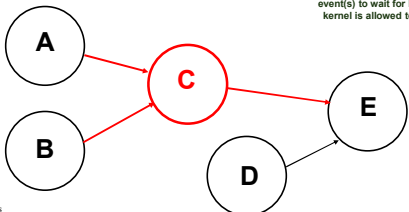
```


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

status = clEnqueueNDRangeKernel( cmdQueue, kernelC, 1, NULL, globalWorkSize, localWorkSize, 2, dependenciesAB, &waitKernelC );

```







Oregon State University  
Computer Graphics

mjb - March 27, 2021

6

### Creating the Full Execution Graph Structure

```

cl_event waitKernelA, waitKernelB, waitKernelC, waitKernelD;

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 );

```

Oregon State University Computer Graphics

7

### Waiting for One Event

```

cl_event waitKernelA, waitKernelB;

...

status = clEnqueueNDRangeKernel( cmdQueue, kernelC, 1, NULL, globalWorkSize, localWorkSize, 1, &waitKernelA, NULL );

```

event(s) to wait for

Oregon State University Computer Graphics

8

### Placing a Barrier in the Command Queue

```

status = clEnqueueBarrier( cmdQueue );

```

Note: this *cannot* throw its own event

This does not complete until all commands enqueued before it have completed.

Oregon State University Computer Graphics

9

### Placing an Event Marker in the Command Queue

```

cl_event waitMarker;
status = clEnqueueMarker( cmdQueue, &waitMarker );

```

Note: this *can* throw its own event

This does not complete until all commands enqueued before it have completed.

This is just like a barrier, but it can throw an event to be waited for.

Oregon State University Computer Graphics

10

### Waiting for Events Without Enqueuing Another Command

```

status = clWaitForEvents( 2, dependencies );

```

event(s) to wait for

This **blocks** until the specified events are thrown, so use it carefully!

Oregon State University Computer Graphics

11

### 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" );
}

```

Call this before starting the timer, before ending the timer, and before retrieving data from an array computed in an OpenCL program.

Oregon State University Computer Graphics

12

13

### 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_int eventStatus;
status = clGetEventInfo( waitKernelC, CL_EVENT_COMMAND_EXECUTION_STATUS, sizeof(cl_int),
                        &eventStatus, NULL );
    
```

↑  
cl\_int is what type  
CL\_EVENT\_COMMAND\_EXECUTION\_STATUS  
returns



CL\_EVENT\_COMMAND\_EXECUTION\_STATUS  
returns one of these

↓

```

CL_QUEUED
CL_SUBMITTED
CL_RUNNING
CL_COMPLETE
    
```

Note that this is 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.



igp - March 27, 2011