Ramez Nahas
I.D#: 26718108
Assignment 3 Report 11 November, 2019

Overview:

The central idea was to limit memory allocations/deallocations and movement of data between host and device. For that, 2 buffers were allocated on the device: one for the balls and one for the unique pairs of balls. All device computation is then done directly using data from the device's global memory. When that computation is done, the updated positions and velocities of the balls are sent back to host which will draw the new frame.

init_pairs(ball, ball*, size_t) kernel:**

This kernel handles initializing the allocated buffer in device memory to store the unique pairs of balls. When this kernel is called, the device memory was already allocated from host with a buffer containing all the different balls. But unlike the allocation and initialization of balls in device memory, which is done exclusively from host function calls, the initialization of pairs has to be done in device. The reason is that the buffer of pairs is an array of pointers to ball. However those pointers are the memory addresses of the balls in device memory and not in host memory.

Other kernels:

The other kernels don't do anything new or different from assignment 2. It is the same logic to do the ball-wall and ball-ball collision.

The only noteworthy thing to note is all the kernels' execution configurations:

- init_pairs is called using 1 block per grid, and 1 thread per block (i.e computation is done serially)
- wall_bounce is called using 1 block per grid for every 256 balls, with 256 threads per block. In other words, between 1 and 256 balls, we use one block of 256 threads. Between 257 and 512 balls, we use 2 grids of 256 threads each, and so on.
- ball_bounce uses the same execution configuration as wall_bounce, but works on the number of pairs instead of number of balls. So 1-256 pairs gives 1 block of 256 threads. 257-512 gives 2 blocks of 256 threads, and so on.

Flow:

We first call the wall_bounce kernel, then the ball_bounce kernel. Once these computations are done, the device sends the updated values to host, which uses those values for the new frame.