Jessica Spokoyny
June 11, 2017
Project #7A
OpenCL / OpenGL Particle System

1. My Machine:

I ran this program in Visual Studio 2015 from a windows 10 laptop. My program's functions are contained in 2 files called spokoynj.cpp and spokoynj.cl. The executable is called spokoynjParticles.exe.

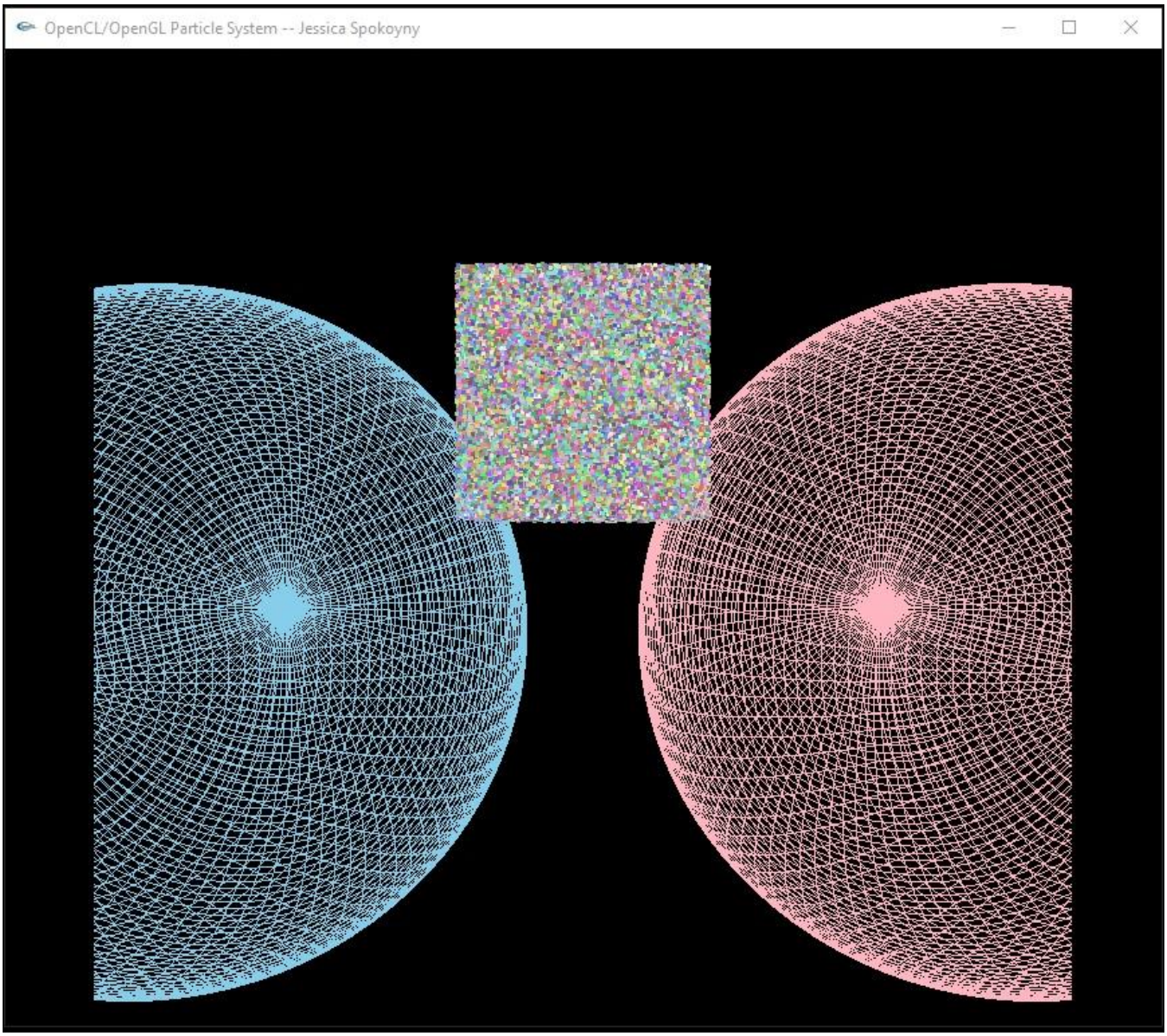
The program can be executed by opening the solution in Visual Studio or simply running the executable.

2. Dynamic Color Changes:

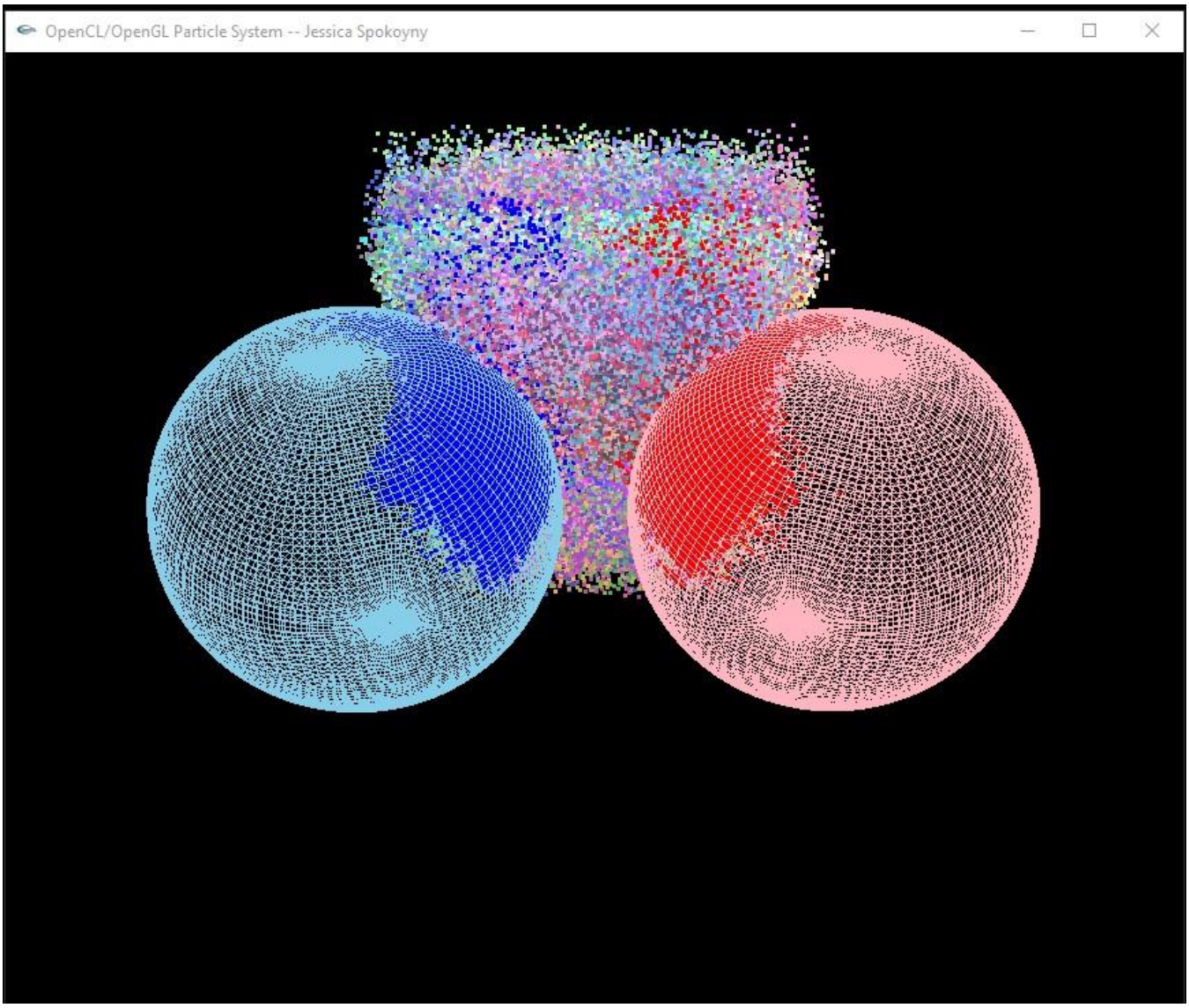
My project starts with a light blue wire sphere, a light pink wire sphere, and a multicolored cube overlapping the two spheres. When a particle bounces off of the light blue sphere, the particles turns dark blue in color and similarly, when a particle bounces off of the light pink sphere, it turns red. This color change is evidenced by the resulting dark blue particles in the light blue sphere and the red particles in the pink sphere.

3. Visual Demonstration:

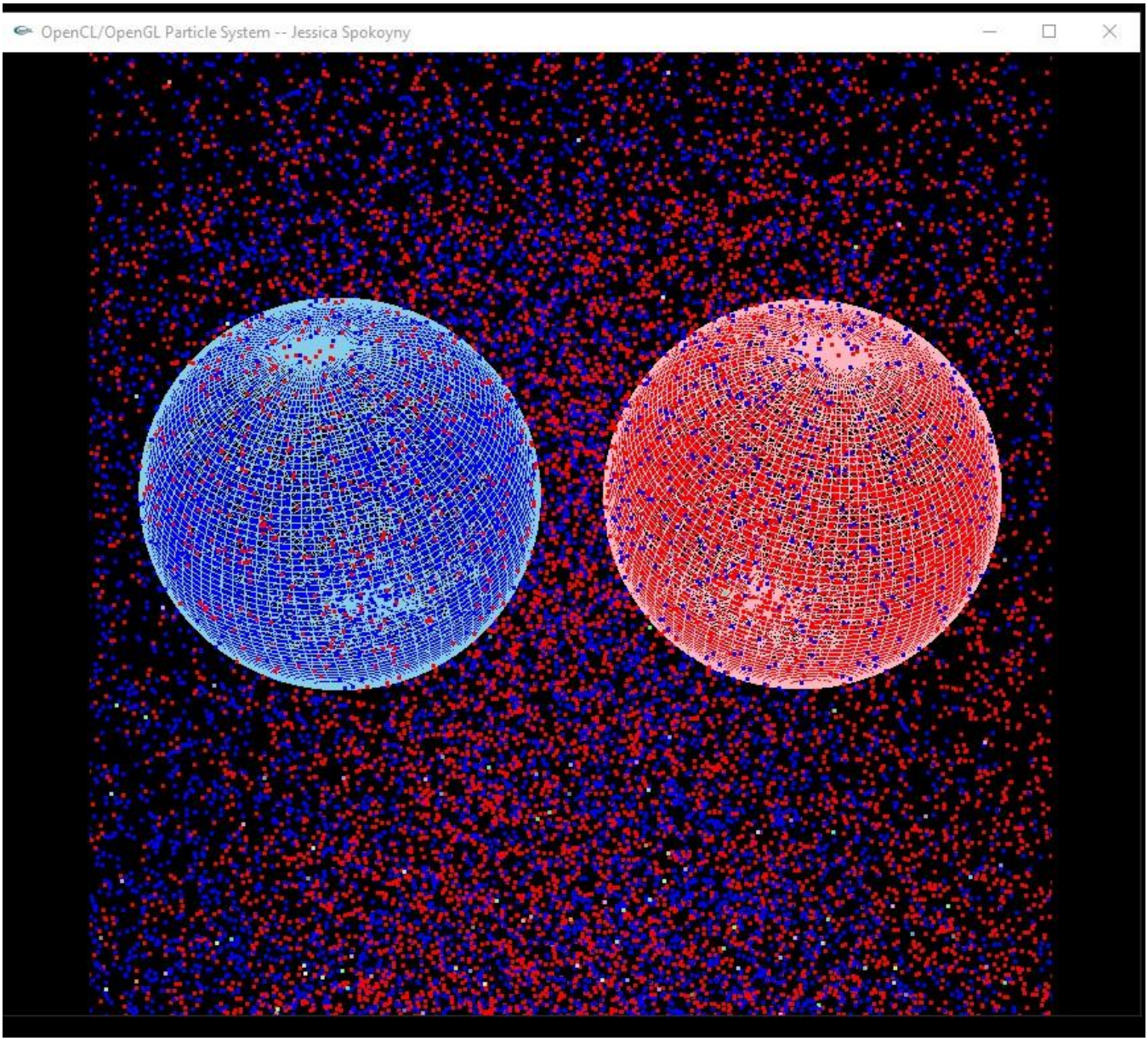
The following Image shows the initial state, 2 spheres (1 pink and 1 light blue) with a cube of multicolored particles overlapping them:



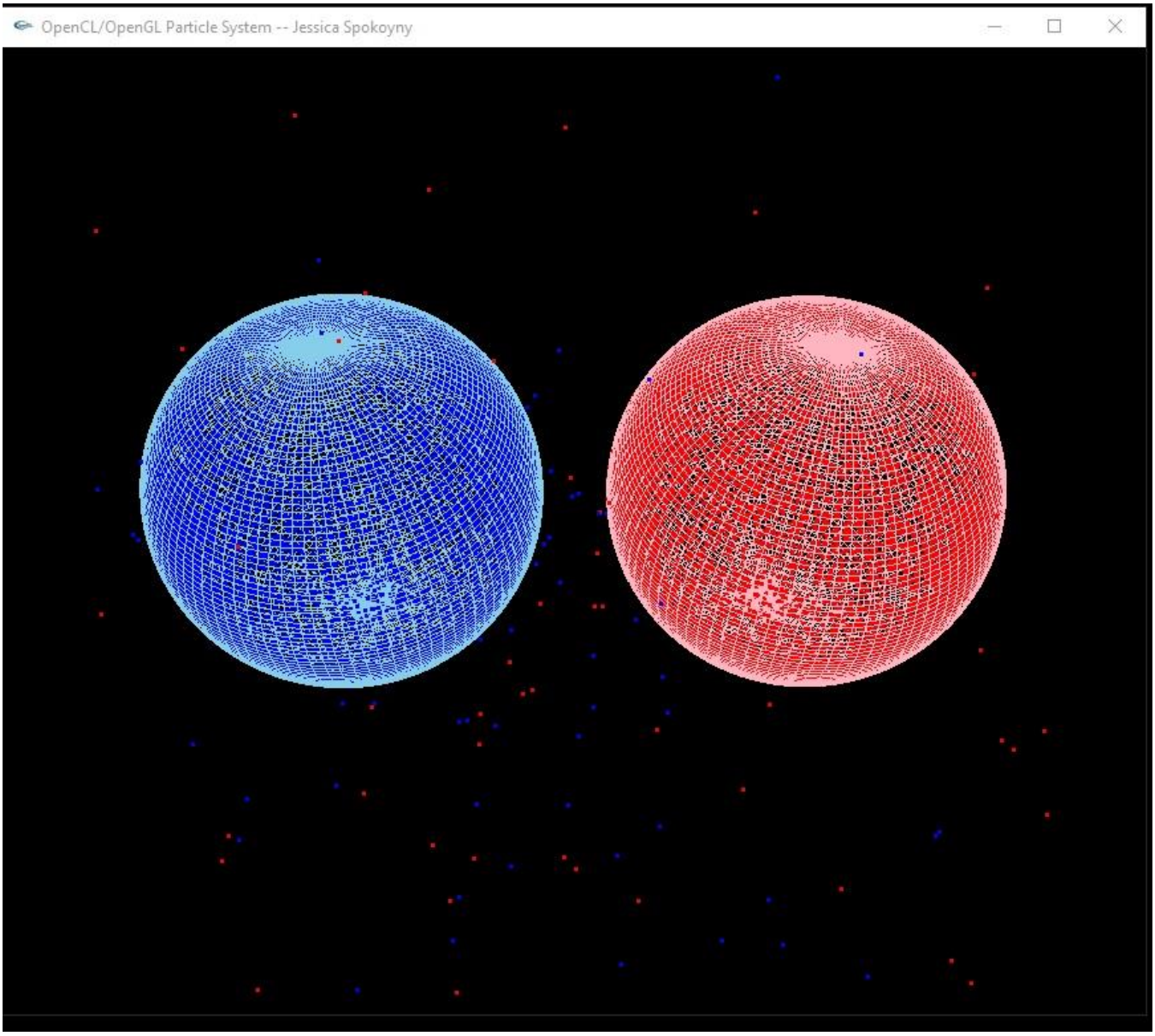
The next image shows the spheres zoomed out, after the explosion has started. We can see that the particles that bounced off the light blue sphere have turned dark blue and those that bounced off the pink sphere have turned red:



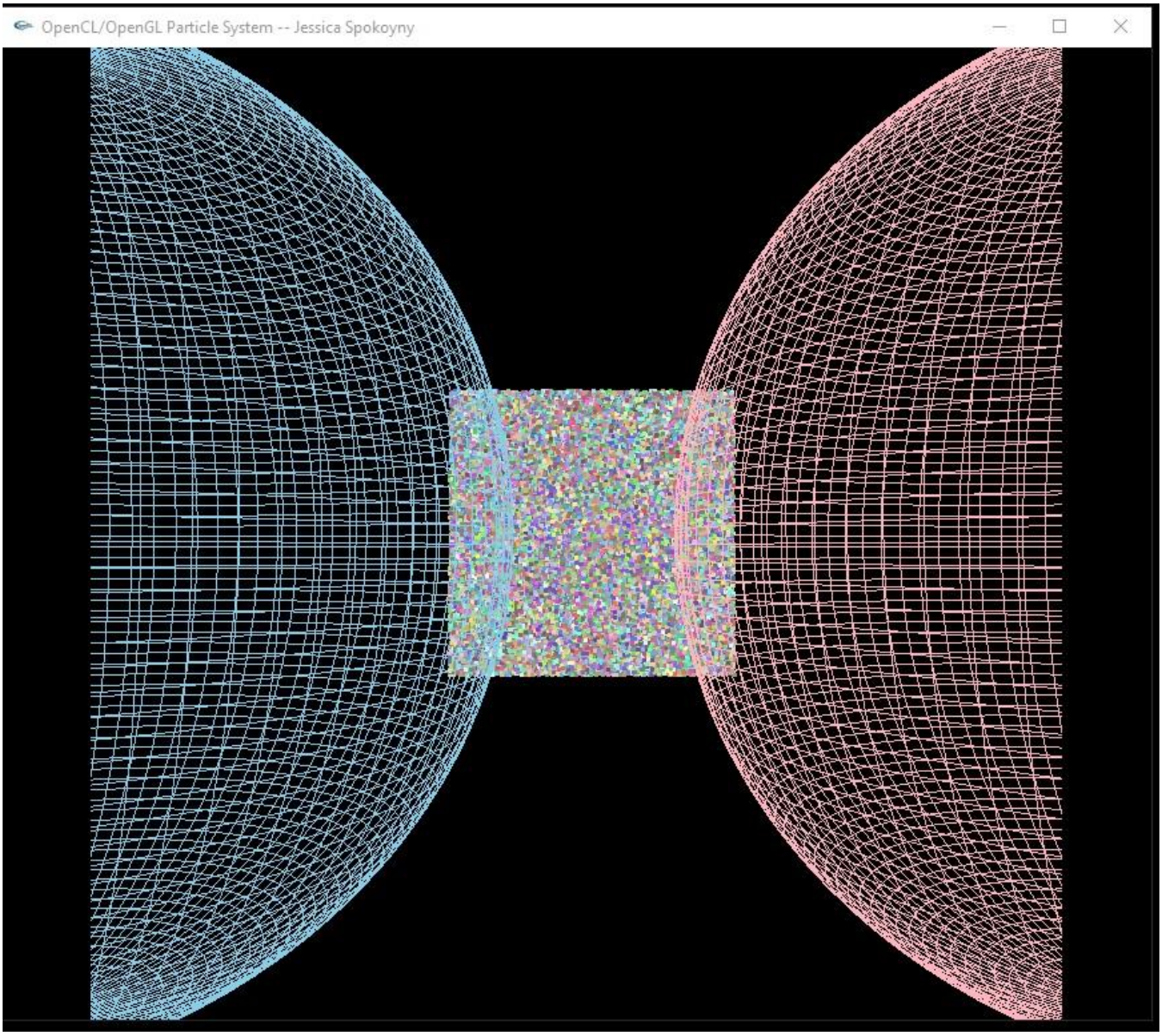
We see here after further bounces that all of the particles have become either dark blue or dark red:



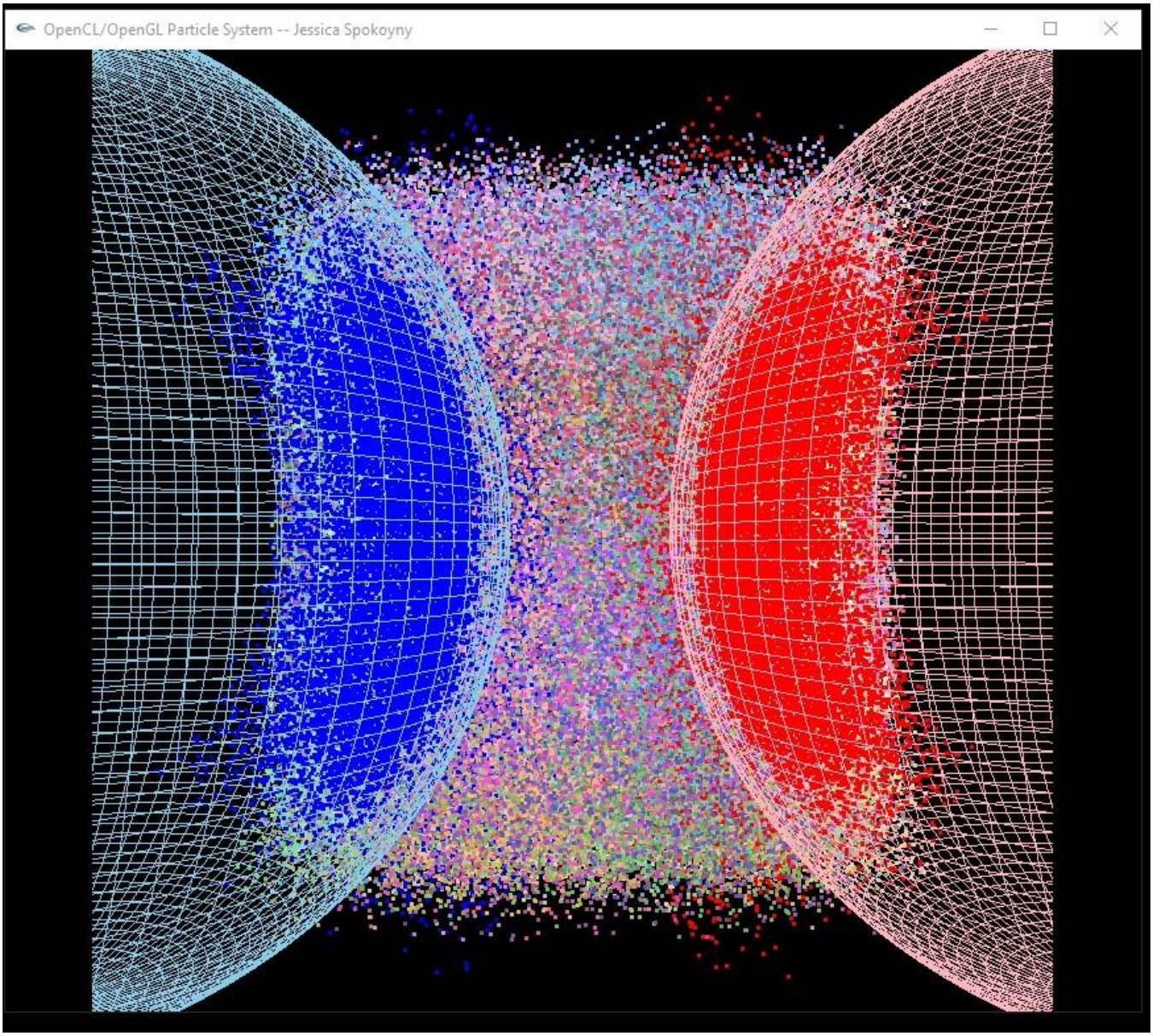
The next image shows after most of the particles have fallen, we are left with dark blue particles bouncing inside the light blue sphere and red particles bouncing inside the pink sphere:



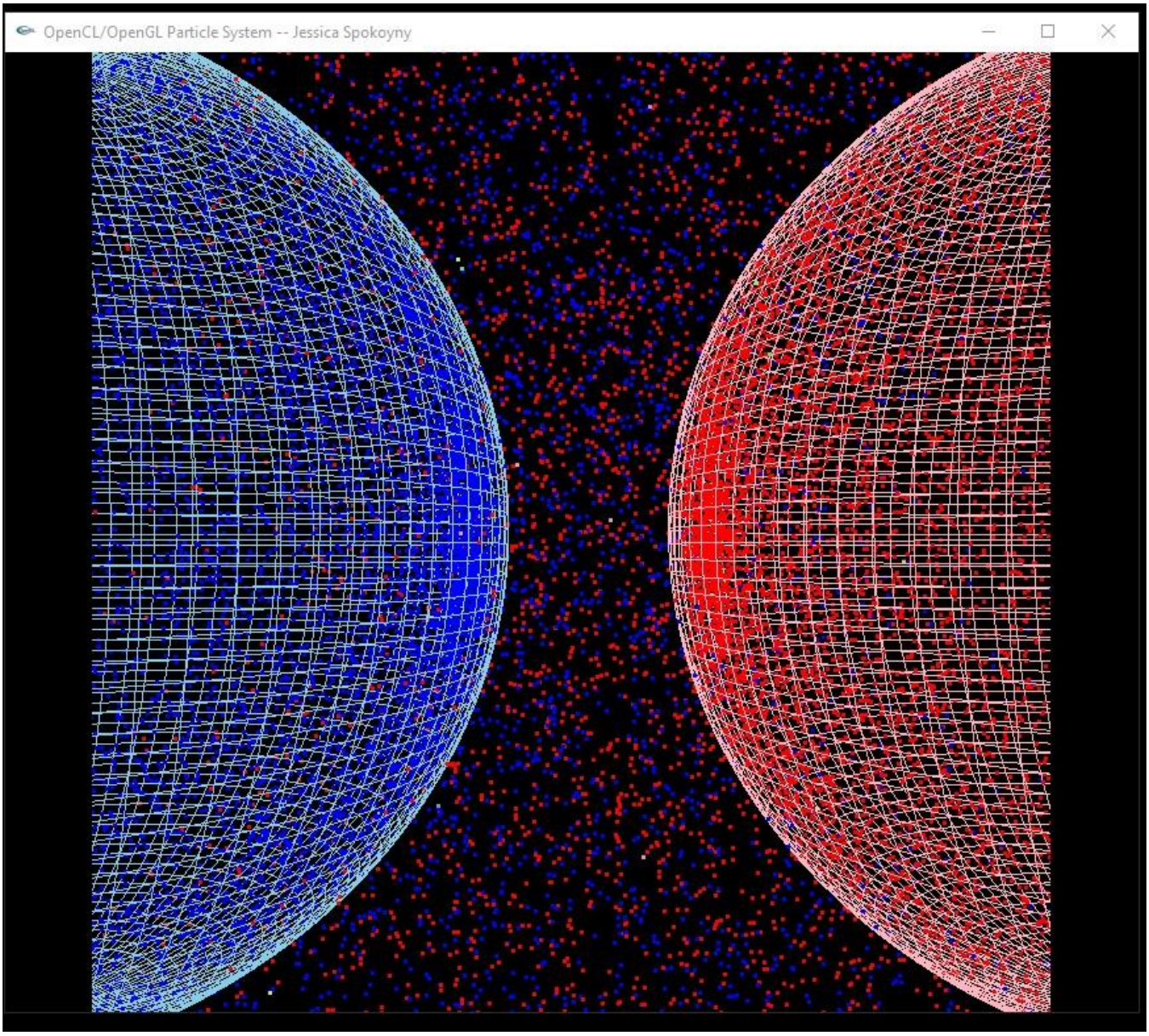
Next is a more close-up view of the demonstration which shows a clear overlapping of the cube with the spheres:



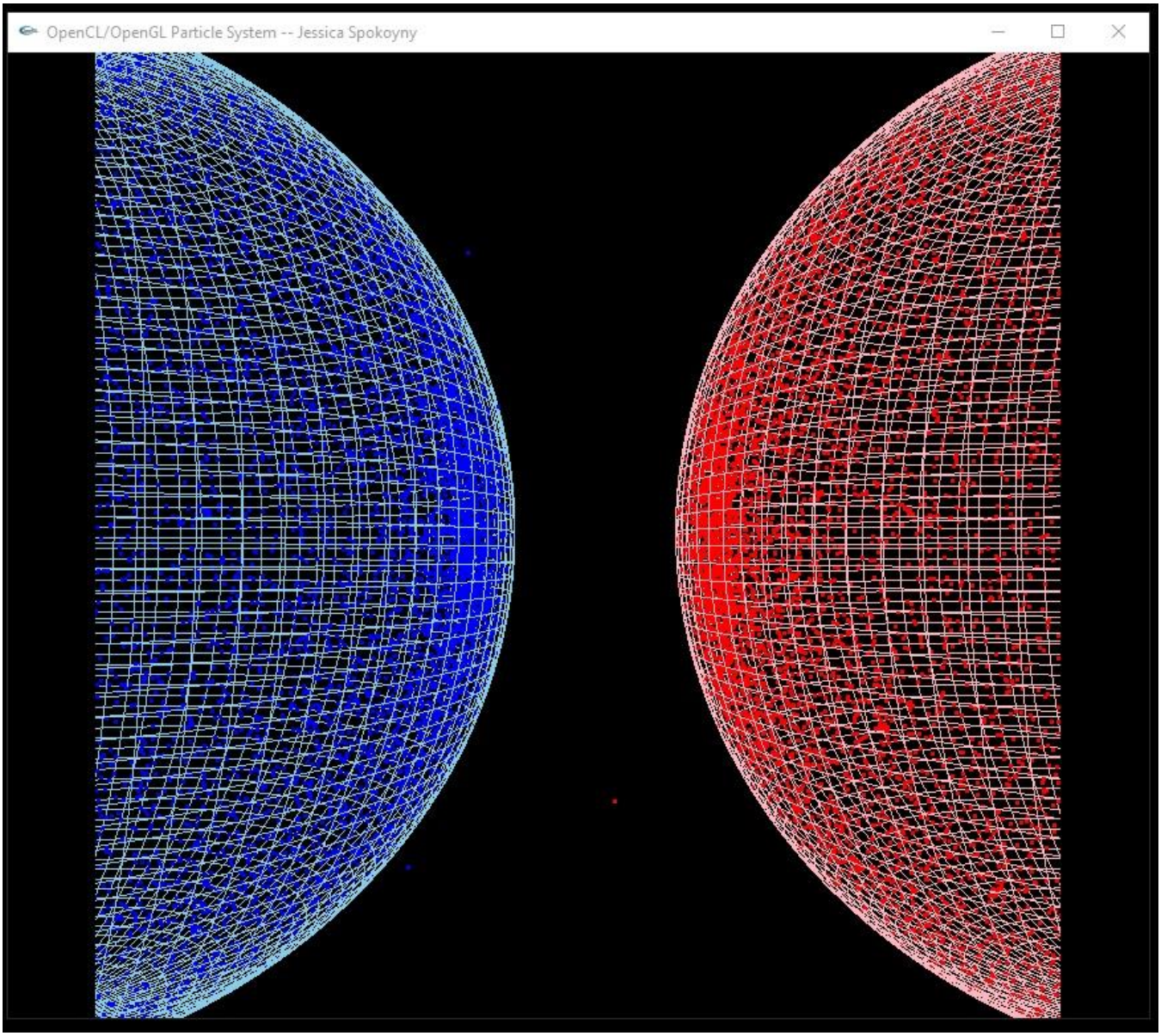
Again after the initial explosion, we see the light blue sphere filling up with blue particles and the pink sphere filling up with red particles:



Next, we can see that all of the particles bouncing between the spheres have turned either red or blue:



And finally, all that's left are dark blue particles bouncing inside the light blue sphere and red particles bouncing inside the pink sphere:

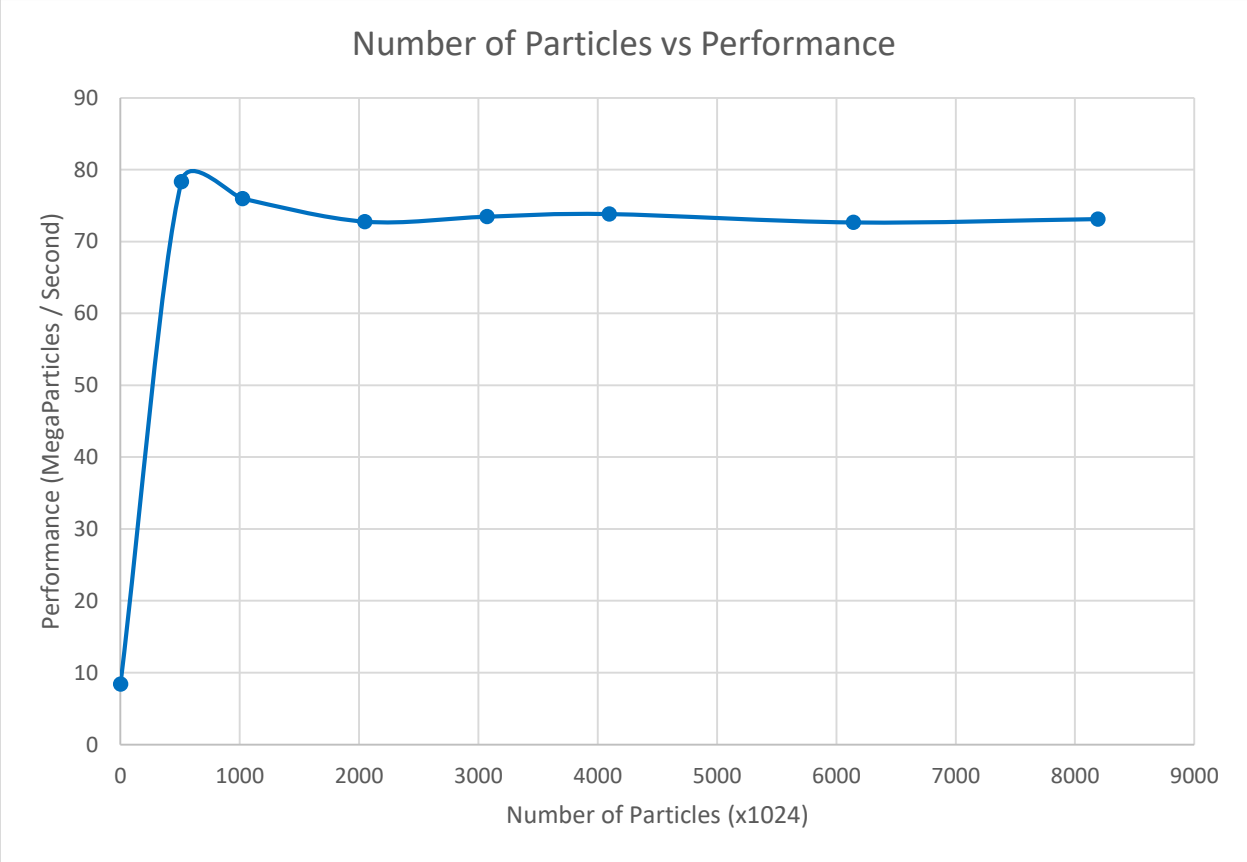


4. My performance results:

With Local Size set to 32:

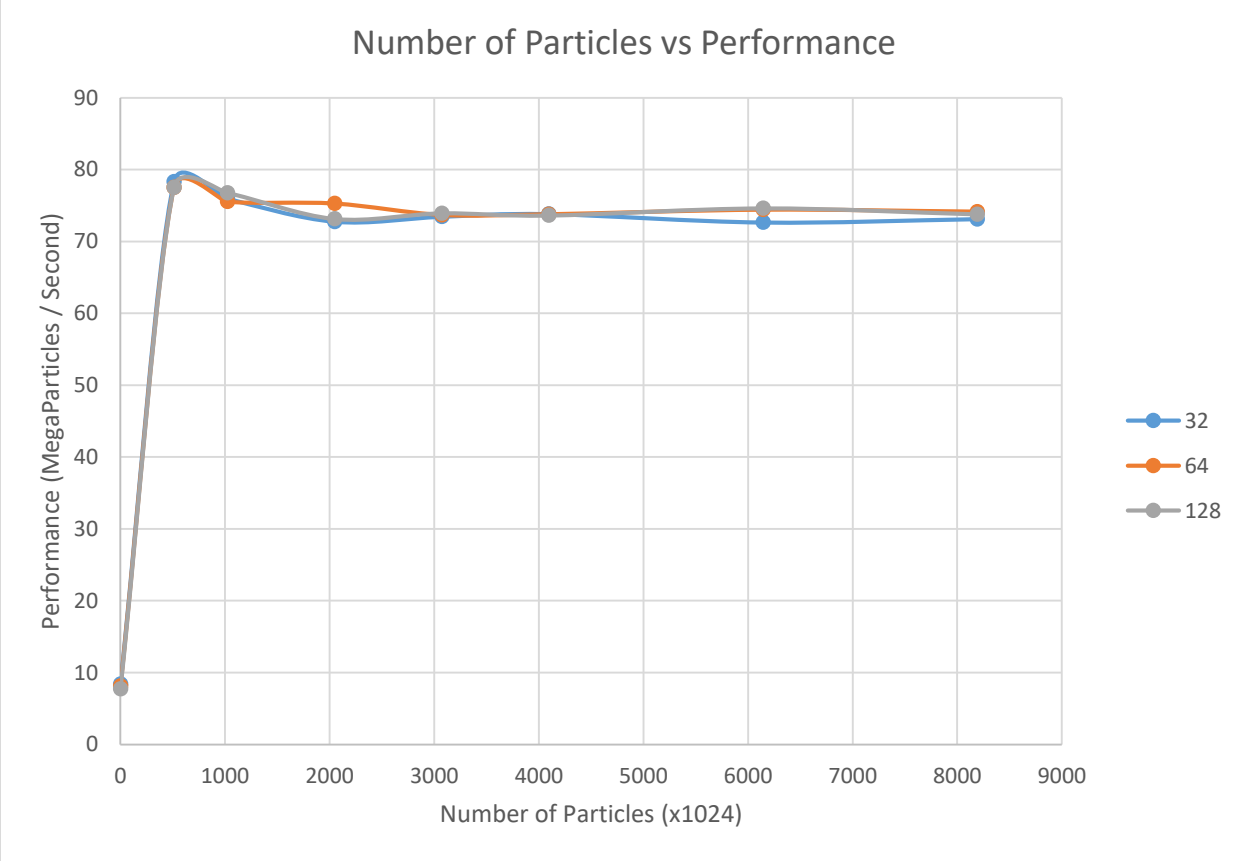
NUMBER OF PARTICLES (x1024)	PERFORMANCE (MegaParticles / Second)
1	8.4097
512	78.319
1024	75.9632
2048	72.752
3072	73.4398
4096	73.8073
6144	72.6399
8192	73.1102

Graph of Performance:



5. Patterns:

We can see from the graph above that as the total number of particles increases, the performance first shoots up rapidly, then stabilizes to an (almost) constant value. I also measured the performance with local sizes of 64 and 128 and the performance patters were almost identical (see below)



6. Explanation of Patterns:

Using OpenGL increases performance by creating a vertex buffer holding the objects to be drawn, then sending it to the graphics card all at once where it can be drawn quickly. If there aren't many particles (or things to draw) the benefits of using OpenGL are diminished. Clearly when we increase the number of particles from 1,024 to 524,288 we see a drastic improvement.

7. Meaning for GPU Parallel Computing:

We can conclude from these results that the larger the global work size, the more valuable OpenGL is to boost performance, up to a certain level. Once that level is reached, we have a

stable performance, regardless of new particles added. For local work size, we see approximately the same results. However, a very small local size is pointless because the benefits don’t compensate for the amount of overhead.