Project #7

OpenCL / OpenGL Particle System

Haoxiang Wang; Student ID: 932359049

1. Machine Running the Test

I used my local machine to run all of the tests. The machine doesn't have an NVidia independent graphic cards but has an Intel HD Graphics 4000 graphic card instead. The program ran perfectly on this machine.

2. Color Setting

I set the color changing dynamically with the position of the particles. I got the original color information and updated with the multiply result of original position data and the time step by using the following equation:

$$Color_{updated} = Color_{origin} + Position_{origin} * DT$$

3. Screen Capture Images

The following images show the screen captures different time steps from different angles.

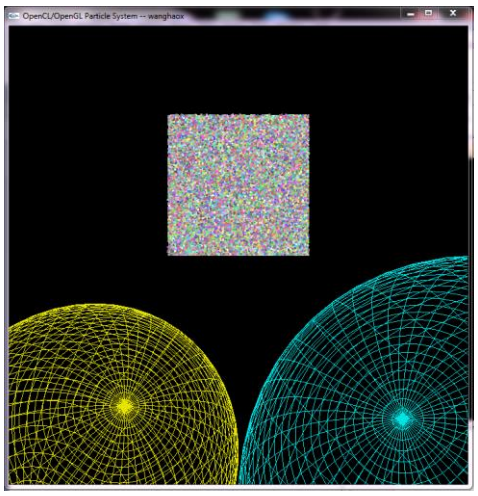


Fig 1. Original View

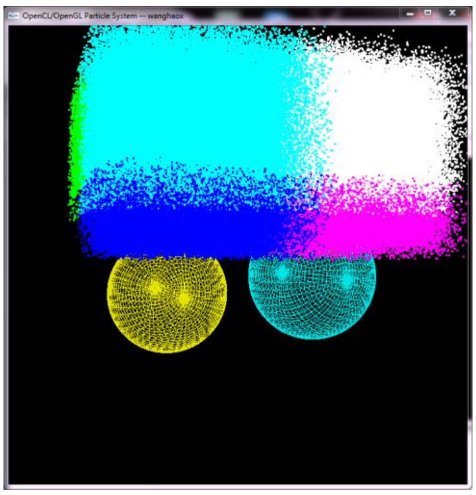


Fig 2. Beginning

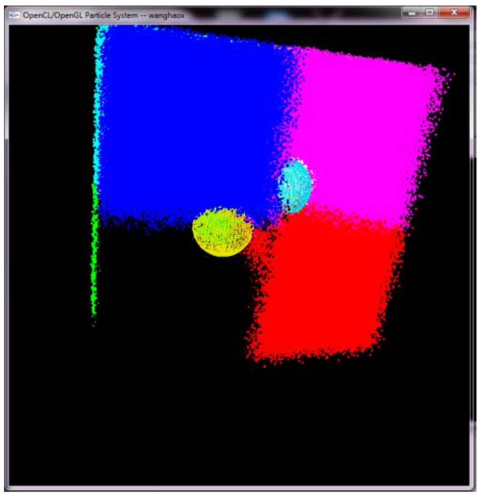


Fig 3. Bouncing (bottom view)

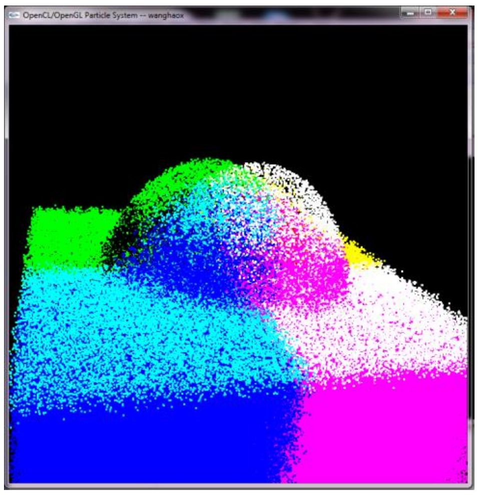


Fig 4. Bouncing (top view)

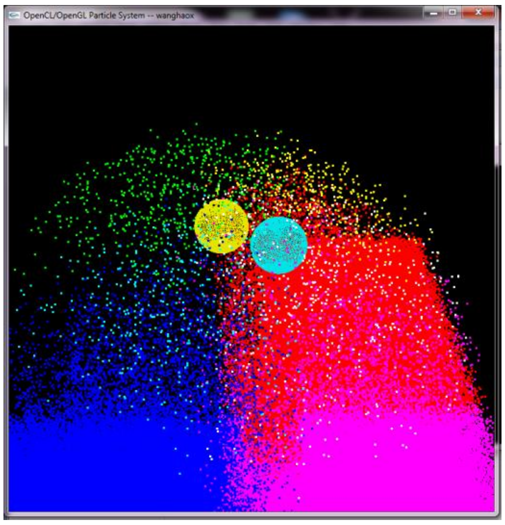


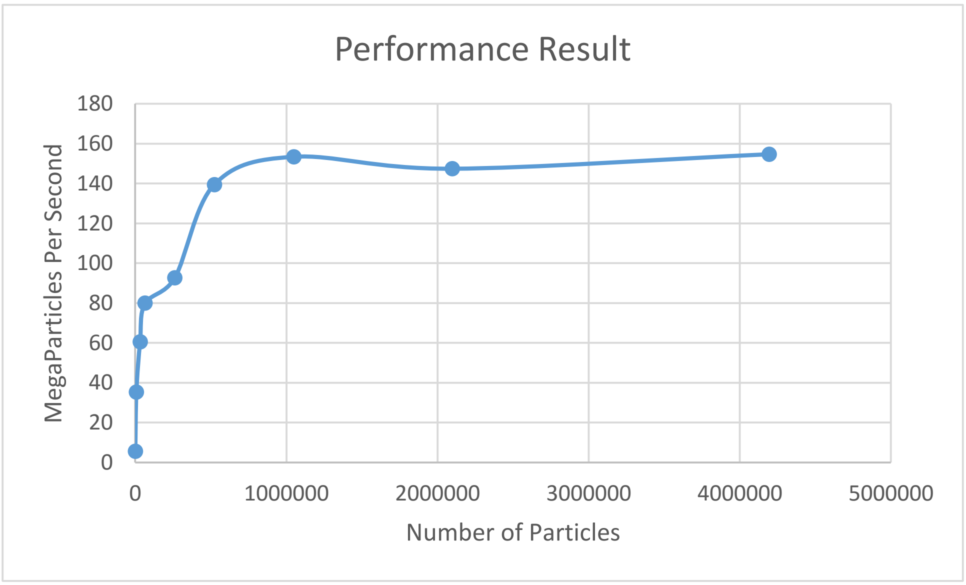
Fig 5. Finishing Bounce

4. Tables and Graphs

The table and graph are really simple this time. The table only contains the variable, which is number of particles, and the performance result. The local work size is fixed to 32 this time, and the unit of the performance is MegaParticles Per Second. The table is shown below.

NUM_Particles	1024	8192	32768	65536	262144	524288	1048576	2097152	4194304
Performance	5.59	35.39	60.56	79.92	92.59	139.43	153.34	147.35	154.6

The graph created based on the table is shown below. There only exists one curve in it, and it represents the performance results with variety number of particles. The pattern and explanation will be introduced in the following sections.



5. Pattern

The pattern in the graph is really obvious. The performance first increases when the number of particles increases. Then the performance stays in a similar range and doesn’t increase with the number pf particles anymore. This pattern looks really similar to one pattern we had in project #6, and that is multiply-reduction performance versus global work size result.

6. Pattern Explanation

As it mentioned in the last section, the pattern is pretty similar compared to one of the results we had in the last object. The reason for this pattern is also similar to the one in project #6. Since the local work size is fixed to 32, when the global work size, which is number of particles in this project, is small, not every work group in the GPU is used to do the computation. Under this condition, the performance increases with the global work size. When the number of particles is large enough, every work group in the GPU is used to do the computation, then the performance stays in the similar range since it could not be higher any more.

7. GPU Parallel Programming Explanation

Using OpenCL to interact with OpenGL in order to deal with graphics problem is really efficient. The project such as particle system requires a computation on a really large size of work. Only using OpenGL will be really hard to get the job done. Fortunately we have OpenCL, which is capable of doing massive of computations. OpenCL has built in functions that can create buffers for OpenGL does directly read and write. OpenCL can also manipulate the data in these buffers, so that we could use parallel computing in the OpenGL project. By including parallel computing, we could definitely get improvements in graphics implementations.