Benjamin Fields
CS 575 Parallel Programming
Project 7a - OpenCL / OpenGL Particle System
June 13, 2017

I. Testing Platform

Host:

Windows Desktop:
Processor: Intel(R) Core(TM) i7-4770 CPU @ 3.40 GHz 3.40 GHz
Installed Memory: 12.0 GB
System type: 64-bit Operating System, x64-based processor
Runtime Environment: Visual Studio 2015

Device:

Nvidia GPU:
GeForce GTX 760 (192-bit)
Driver version: 381.65
CUDA Cores: 1152
Memory data rate: 5600 MHz
Memory interface: 192-bit
Memory bandwidth: 134.40 GB/s
Shared system memory: 6103 MB

II. Particle Color Dynamics

As per the requirements of this project we were required to include dynamic behavior regarding the color of the particles. The behavior needs to take into account any interactions with bumpers present in the scene. There are two circles present in the scene with specified colors of yellow and red. The particles are all created with the color white, but once the simulation begins the particles disperse. When a particle comes into contact with a sphere the particle changes to the color of the sphere that it came into contact with. This relationship can be seen in Figure 1 on the following page. The image shows the initial position of the particles followed by the progression as the simulation continues. In the heart of the simulation we can see that all particles that did not come into contact with a sphere are still white but all that bounced off a sphere changed colors to that of the sphere it came into contact with.

Benjamin Fields
CS 575 Parallel Programming
Project 7a - OpenCL / OpenGL Particle System
June 13, 2017

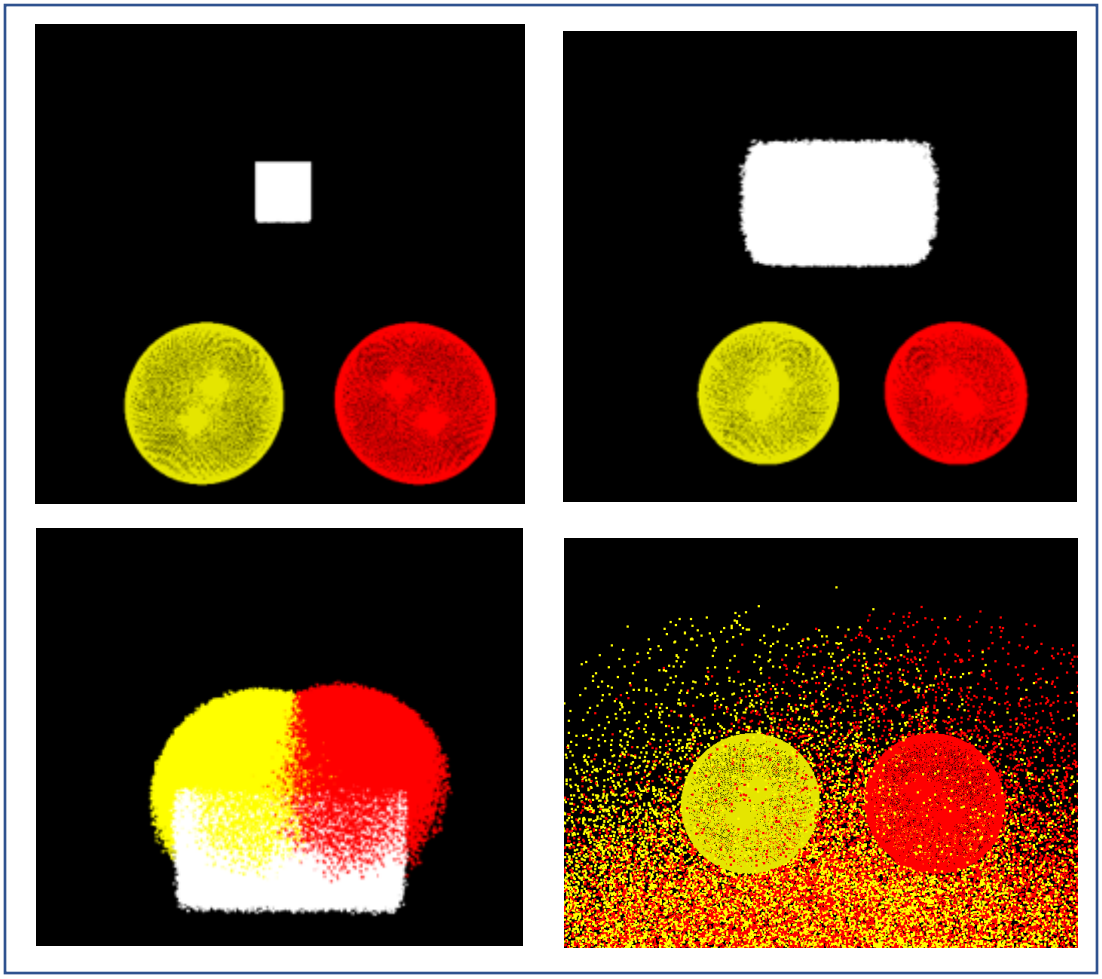

Figure 1: Representation of the progression of the particle simulation and how the colors are changing as they collide with each sphere. The time progression is clockwise starting in the top left image.

Benjamin Fields
CS 575 Parallel Programming
Project 7a - OpenCL / OpenGL Particle System
June 13, 2017

III. Performance Table and Graph

Local size = 32

Frames Run = 500

Number of Particles	Performance(GigiParticles/Sec)
1024	0.013878
102400	0.626974
204800	0.811061
307200	0.881916
409600	0.929312
512000	0.964061
512000	0.968356
1024000	1.02764
1536000	1.05113
2048000	1.06932
2560000	1.06378
3072000	1.0551
3584000	1.06434
4096000	1.07611
4608000	1.07126
5120000	1.06763
5632000	1.07106
6144000	1.02881
6656000	1.07485
7168000	1.07165
7680000	1.06989
8192000	1.08103
8704000	1.06978
9216000	1.07493
9728000	1.07364
10240000	1.08512

Table 1: Performance data for each number of particles.

All the trials were performed with a Local size of 32 and a duration of 500 frames. A termination condition was added to check the number of the current frame and end the simulation once the 500^{th} frame was reached. The number of particles were continuously increased with each trial.

Benjamin Fields
CS 575 Parallel Programming
Project 7a - OpenCL / OpenGL Particle System
June 13, 2017

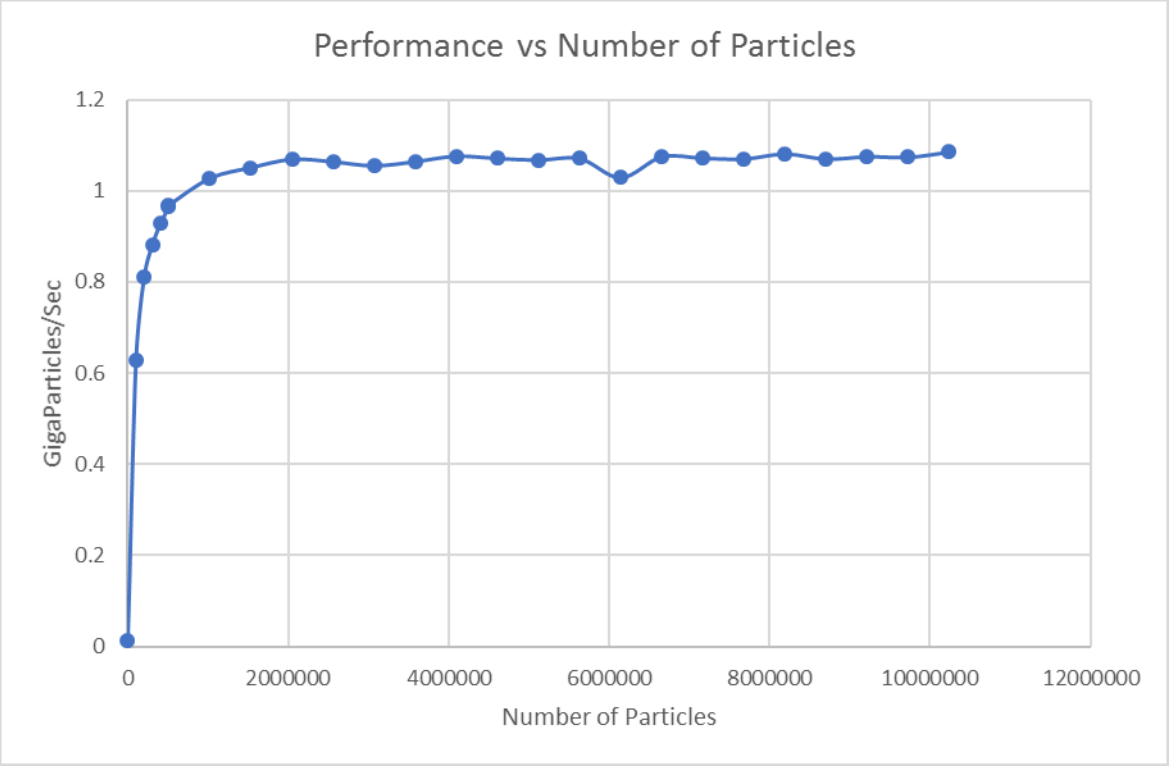

Figure 2: Performance plot of the trials of each number of particles.

IV. Performance Patterns

When observing figure 2 as seen above there are some very distinct patterns. The performance steadily increases with the smaller number of particles and then quickly plateaus and does not change with all. Aside from this there are no other distinguishing patterns providing insight into the performance of the program.

V. Pattern Analysis

With knowledge of the OpenCL pipeline the patterns seen in figure 2 can be explained. As mentioned previously, the performance quickly plateaus and does not change. OpenCL functions by assigning commands to a command buffer and then continuously processing those commands as each event is processed. This naturally means that the system can only process the commands at a discrete rate. This rate or processing speed is represented by the plateau seen in figure 2. Each command is the same event and so every trial achieves the same performance because the OpenCL model is processing commands at the same speed in each trial. In the instances lower than 2 million the system is not hitting its limit and thus we see the lower performance. The plateau is also going to be a function of the tasks performed in each event. The performance can change with the size of each event passed into the command buffer.

Benjamin Fields
CS 575 Parallel Programming
Project 7a - OpenCL / OpenGL Particle System
June 13, 2017

VI. Meaning in the Context of GPU parallel computing

When considering this kind of program on a GPU system with OpenCL we must take into consideration a few things. First, we need to think about the complexity of the tasks being performed and whether or not they lend themselves to a parallel execution model. Also, if the tasks are too complicated then the system may not be able to achieve very promising performance results. The specifications of the GPU that is being used must also be considered. If the program requires more memory than is available on the GPU then the system will not be able to run the program. The number of cores and processing power of the GPU must also be considered as the OpenCL programming model is built on the idea of command buffers that continuously feed the system with work. If these command buffers are not processed effectively due to limitation in the processing on the GPU then the performance will take a hit. One must also consider the number of command buffers to be used. If the system will be blocked with one command buffer then we can partition up the events to different command buffers and then the GPU will be able to take advantage of all the tasks even if one event is taking an extended time frame.