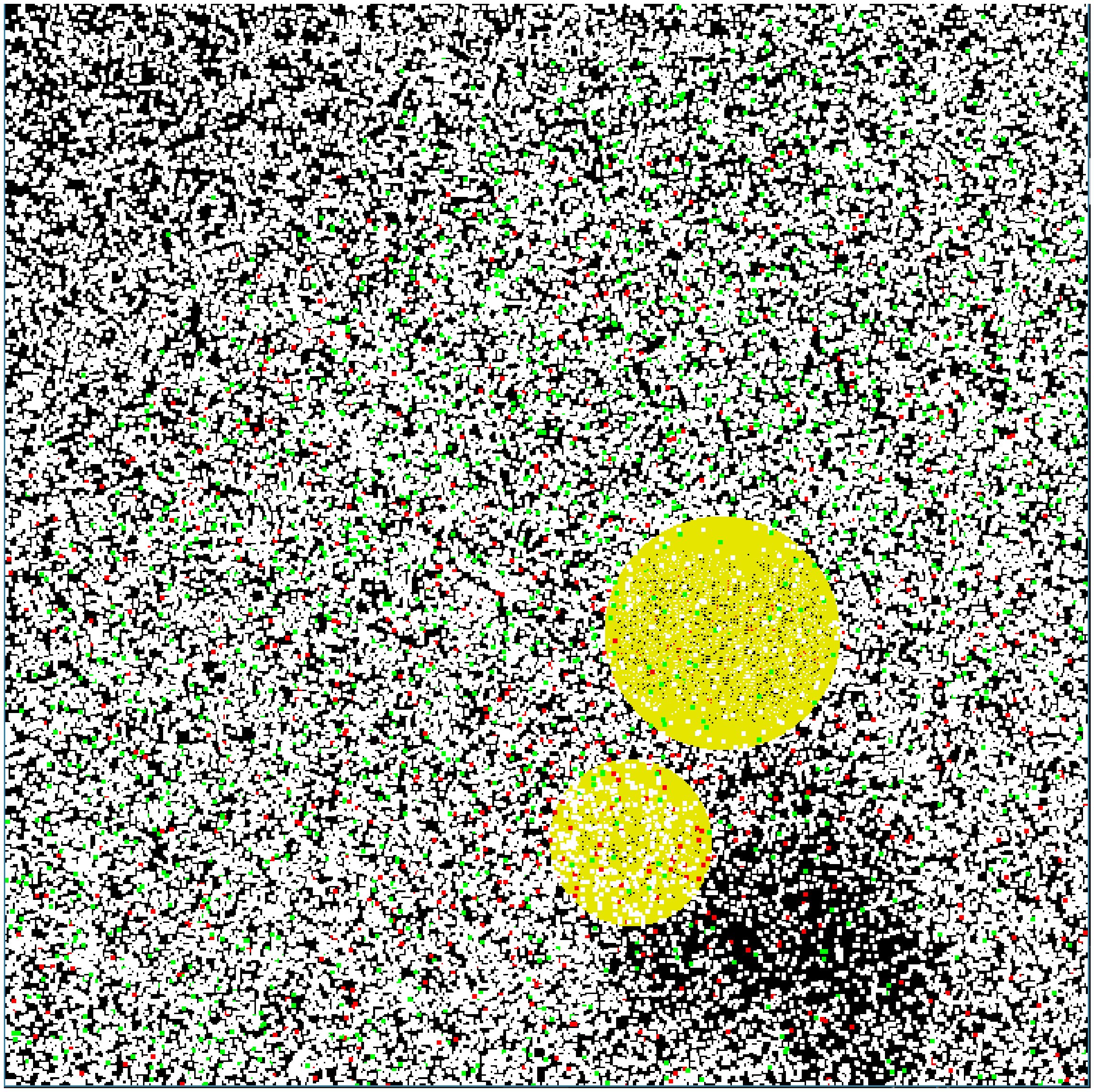
Taylor Kirkpatrick

Project 7A

1. I ran this code in Visual Studio 2017 on my personal desktop. My desktop is a Windows 10 Home edition box with an i7-6700 CPU with four cores. I have 16 gigs of DDR4 RAM. My GPU is a GeForce GTX 960 with 1024 CUDA cores and 4 gigs of VRAM. Load on my system during testing was reasonably low with only Chrome and Visual Studio having a notable resource usage, and not enough that it should have caused any issues during testing.
2. I caused all particles to change color if they should bounce off of a sphere. As I added a second sphere and changed them around a bit, particles bouncing off of the higher sphere were changed to be green, and particles bouncing off of the lower sphere changed to be red. All particles started as white.

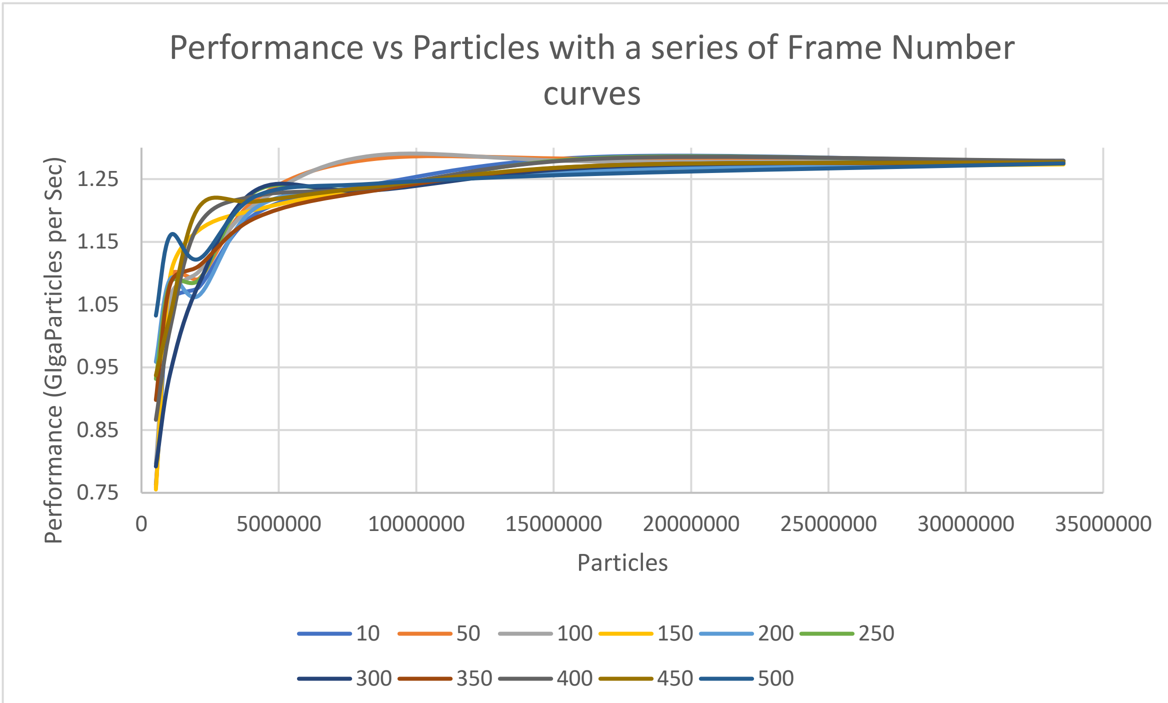
3.

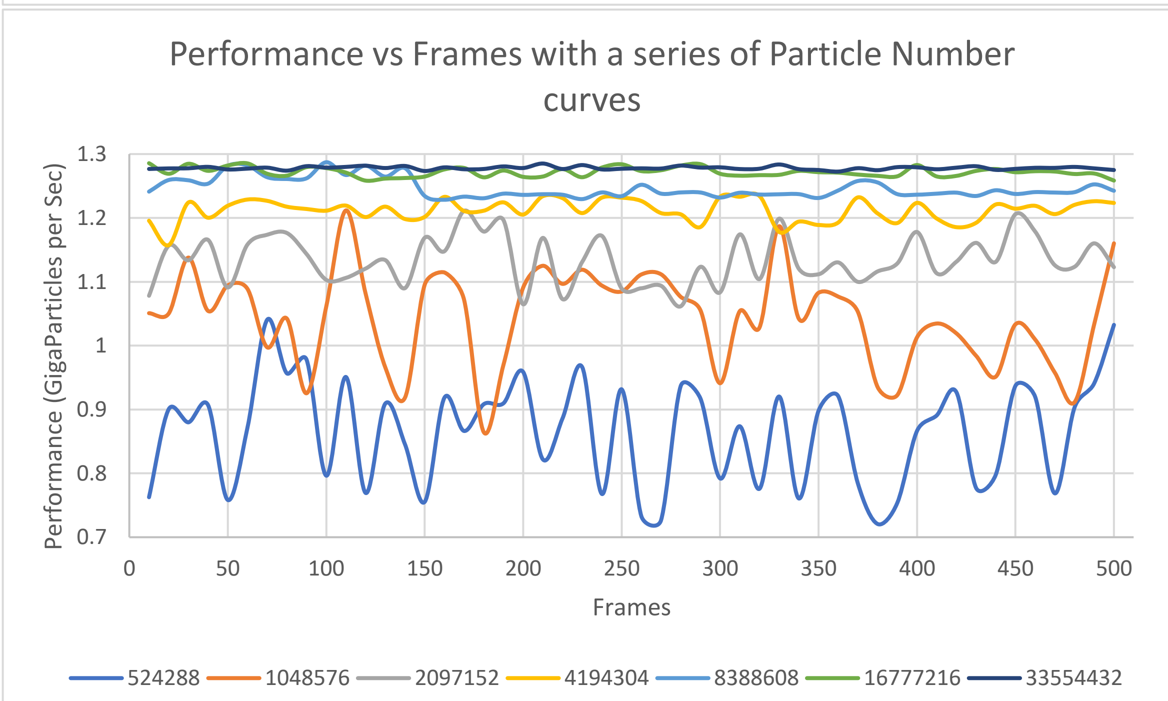


4. I elected to make a supplemental graph as the requested “Performance vs Particles” graph was difficult to see.

   A note about the data: I elected to only record timing information every 10 frames, and only up to 500 frames. I wanted to avoid having data with varying numbers of values like I had in Project 6 which created messy and hard to understand graphs and tables, so for this project I only recorded up to 500 frames. I only recorded once every 10 frames as I figured that recording for every single frame may effect the running speed of the project itself. In addition, any slowdown caused by a file write should be resolved by 10 frames, so I assumed that that spacing would create a better representative curve with minimal effect from the recording operations.

Also note that the Performance vs Particles graph only has curves of each 50 frames, and the initial series of 10 frames. Had this graph been made with all data collected, there would be 50 individual series which would have been even less readable than the graph is here with 11 of them.





I've also included a graph that reports average performance of each particle set versus performance, which shows the same general curve as the first graph but a tad clearer.

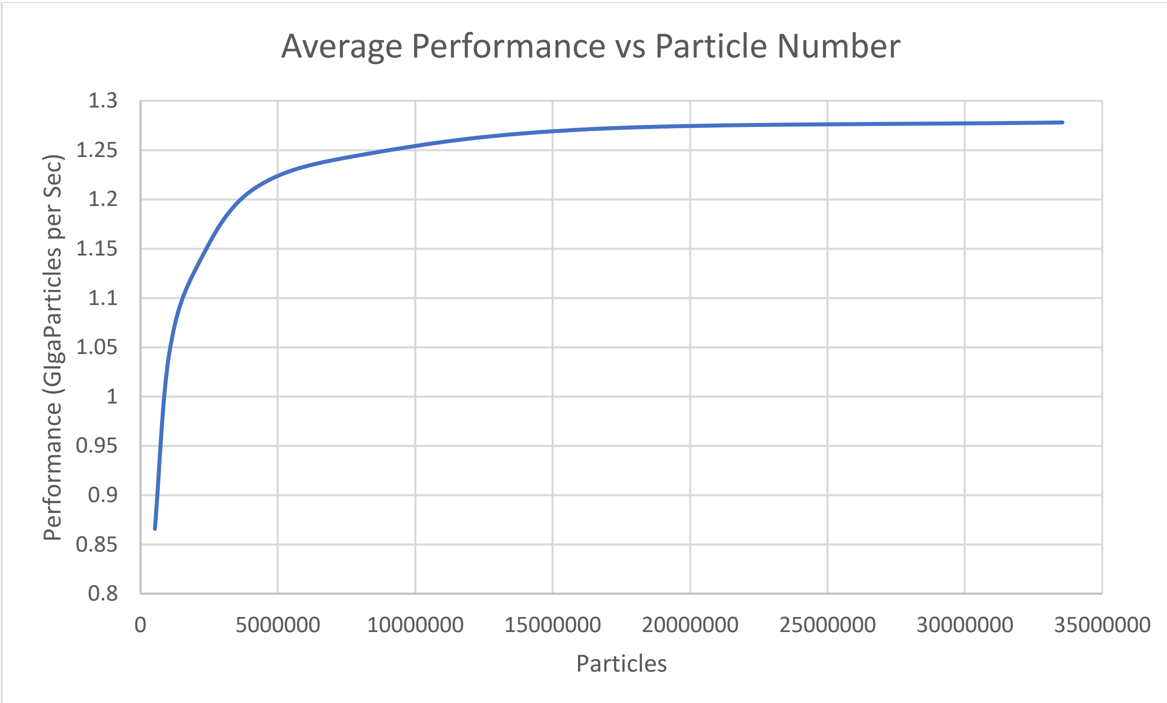


Table:

Number of Particles	Frames	GigaParticles per Second
524288	10	0.76263
524288	20	0.90036
524288	30	0.87993
524288	40	0.90691
524288	50	0.75832
524288	60	0.87158
524288	70	1.04049
524288	80	0.95663
524288	90	0.97808
524288	100	0.79639
524288	110	0.9509
524288	120	0.76936
524288	130	0.90927
524288	140	0.8454
524288	150	0.75537
524288	160	0.91837

524288	170	0.86638
524288	180	0.90833
524288	190	0.90975
524288	200	0.95874
524288	210	0.8219
524288	220	0.88619
524288	230	0.96617
524288	240	0.76766
524288	250	0.93161
524288	260	0.73315
524288	270	0.72644
524288	280	0.93661
524288	290	0.9174
524288	300	0.79206
524288	310	0.87376
524288	320	0.77586
524288	330	0.92031
524288	340	0.76064
524288	350	0.89804
524288	360	0.92079
524288	370	0.78387
524288	380	0.72073
524288	390	0.75374
524288	400	0.86638
524288	410	0.89071
524288	420	0.92715
524288	430	0.77724
524288	440	0.79785
524288	450	0.93711
524288	460	0.91982
524288	470	0.76868
524288	480	0.90316
524288	490	0.94115
524288	500	1.03248
1048576	10	1.05083
1048576	20	1.05019
1048576	30	1.13785
1048576	40	1.05432
1048576	50	1.09432
1048576	60	1.08886

1048576	70	0.99737
1048576	80	1.04235
1048576	90	0.92543
1048576	100	1.06105
1048576	110	1.21174
1048576	120	1.08144
1048576	130	0.9659
1048576	140	0.91861
1048576	150	1.09535
1048576	160	1.1146
1048576	170	1.07181
1048576	180	0.86445
1048576	190	0.96993
1048576	200	1.0909
1048576	210	1.12502
1048576	220	1.09673
1048576	230	1.11888
1048576	240	1.09398
1048576	250	1.0848
1048576	260	1.1114
1048576	270	1.11175
1048576	280	1.07643
1048576	290	1.0556
1048576	300	0.94115
1048576	310	1.054
1048576	320	1.02823
1048576	330	1.18701
1048576	340	1.04204
1048576	350	1.08278
1048576	360	1.07677
1048576	370	1.05273
1048576	380	0.9351
1048576	390	0.92249
1048576	400	1.013
1048576	410	1.03463
1048576	420	1.01922
1048576	430	0.98415
1048576	440	0.95142
1048576	450	1.0334
1048576	460	1.00949

1048576	470	0.95768
1048576	480	0.91141
1048576	490	1.03371
1048576	500	1.16017
2097152	10	1.0781
2097152	20	1.15652
2097152	30	1.1336
2097152	40	1.1656
2097152	50	1.09124
2097152	60	1.15825
2097152	70	1.17363
2097152	80	1.1764
2097152	90	1.14345
2097152	100	1.10297
2097152	110	1.10612
2097152	120	1.12068
2097152	130	1.13397
2097152	140	1.09005
2097152	150	1.16892
2097152	160	1.14777
2097152	170	1.2111
2097152	180	1.17859
2097152	190	1.19657
2097152	200	1.06477
2097152	210	1.16872
2097152	220	1.07279
2097152	230	1.13121
2097152	240	1.17186
2097152	250	1.08954
2097152	260	1.08988
2097152	270	1.09381
2097152	280	1.06154
2097152	290	1.12339
2097152	300	1.08345
2097152	310	1.17423
2097152	320	1.10419
2097152	330	1.19883
2097152	340	1.12032
2097152	350	1.11175
2097152	360	1.1303

2097152	370	1.09984
2097152	380	1.1162
2097152	390	1.12865
2097152	400	1.17799
2097152	410	1.11335
2097152	420	1.1314
2097152	430	1.16094
2097152	440	1.13121
2097152	450	1.20587
2097152	460	1.17859
2097152	470	1.12592
2097152	480	1.12339
2097152	490	1.16017
2097152	500	1.12284
4194304	10	1.19565
4194304	20	1.15738
4194304	30	1.22417
4194304	40	1.20027
4194304	50	1.21904
4194304	60	1.2288
4194304	70	1.22675
4194304	80	1.21755
4194304	90	1.21395
4194304	100	1.21131
4194304	110	1.21925
4194304	120	1.20131
4194304	130	1.21765
4194304	140	1.19821
4194304	150	1.20162
4194304	160	1.23293
4194304	170	1.21079
4194304	180	1.21131
4194304	190	1.22438
4194304	200	1.20525
4194304	210	1.23402
4194304	220	1.23097
4194304	230	1.20754
4194304	240	1.23184
4194304	250	1.23173
4194304	260	1.22653

4194304	270	1.20754
4194304	280	1.20566
4194304	290	1.1857
4194304	300	1.23314
4194304	310	1.23314
4194304	320	1.23434
4194304	330	1.17799
4194304	340	1.19422
4194304	350	1.18893
4194304	360	1.19279
4194304	370	1.23216
4194304	380	1.20702
4194304	390	1.19197
4194304	400	1.22352
4194304	410	1.19904
4194304	420	1.18549
4194304	430	1.19228
4194304	440	1.22106
4194304	450	1.21448
4194304	460	1.21893
4194304	470	1.20598
4194304	480	1.22042
4194304	490	1.2261
4194304	500	1.22342
8388608	10	1.24115
8388608	20	1.25934
8388608	30	1.25889
8388608	40	1.25363
8388608	50	1.28166
8388608	60	1.28225
8388608	70	1.26373
8388608	80	1.26094
8388608	90	1.26202
8388608	100	1.28715
8388608	110	1.26717
8388608	120	1.28184
8388608	130	1.26465
8388608	140	1.27785
8388608	150	1.23434
8388608	160	1.22842

8388608	170	1.23331
8388608	180	1.23081
8388608	190	1.23796
8388608	200	1.23604
8388608	210	1.23708
8388608	220	1.23626
8388608	230	1.22945
8388608	240	1.23994
8388608	250	1.23402
8388608	260	1.25189
8388608	270	1.23801
8388608	280	1.23988
8388608	290	1.23982
8388608	300	1.23189
8388608	310	1.23938
8388608	320	1.23658
8388608	330	1.23702
8388608	340	1.23724
8388608	350	1.23119
8388608	360	1.24297
8388608	370	1.25787
8388608	380	1.25549
8388608	390	1.23724
8388608	400	1.23637
8388608	410	1.23796
8388608	420	1.23955
8388608	430	1.23434
8388608	440	1.2433
8388608	450	1.23746
8388608	460	1.24043
8388608	470	1.23971
8388608	480	1.24032
8388608	490	1.25251
8388608	500	1.24247
16777216	10	1.28564
16777216	20	1.26913
16777216	30	1.28478
16777216	40	1.27371
16777216	50	1.28204
16777216	60	1.28541

16777216	70	1.26916
16777216	80	1.26614
16777216	90	1.27954
16777216	100	1.27802
16777216	110	1.27092
16777216	120	1.25841
16777216	130	1.26156
16777216	140	1.26248
16777216	150	1.26473
16777216	160	1.27592
16777216	170	1.27823
16777216	180	1.26356
16777216	190	1.27423
16777216	200	1.26428
16777216	210	1.26479
16777216	220	1.27647
16777216	230	1.26373
16777216	240	1.27884
16777216	250	1.28399
16777216	260	1.2733
16777216	270	1.27464
16777216	280	1.28198
16777216	290	1.28425
16777216	300	1.26913
16777216	310	1.26611
16777216	320	1.26694
16777216	330	1.26746
16777216	340	1.27365
16777216	350	1.27159
16777216	360	1.27087
16777216	370	1.26786
16777216	380	1.26588
16777216	390	1.26514
16777216	400	1.28278
16777216	410	1.26493
16777216	420	1.2658
16777216	430	1.27359
16777216	440	1.27662
16777216	450	1.27156
16777216	460	1.27315

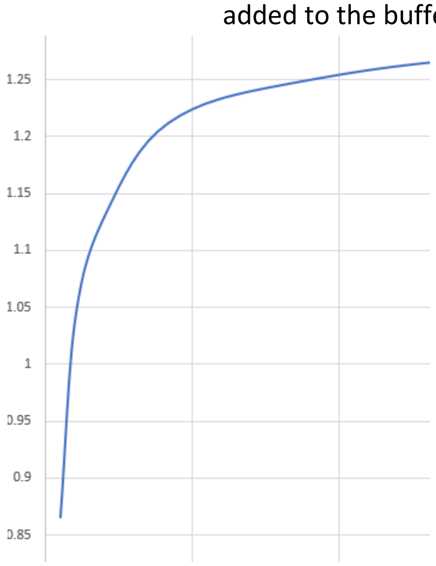
16777216	470	1.2726
16777216	480	1.26873
16777216	490	1.26951
16777216	500	1.25846
33554432	10	1.27665
33554432	20	1.27752
33554432	30	1.2777
33554432	40	1.27994
33554432	50	1.27586
33554432	60	1.27714
33554432	70	1.27868
33554432	80	1.274
33554432	90	1.28101
33554432	100	1.27878
33554432	110	1.27988
33554432	120	1.28172
33554432	130	1.27812
33554432	140	1.28136
33554432	150	1.27346
33554432	160	1.27913
33554432	170	1.2762
33554432	180	1.27663
33554432	190	1.28066
33554432	200	1.27812
33554432	210	1.28515
33554432	220	1.27655
33554432	230	1.28273
33554432	240	1.27589
33554432	250	1.27698
33554432	260	1.27764
33554432	270	1.27731
33554432	280	1.28195
33554432	290	1.27883
33554432	300	1.27921
33554432	310	1.27653
33554432	320	1.27701
33554432	330	1.28366
33554432	340	1.27636
33554432	350	1.27503
33554432	360	1.27257

33554432	370	1.27785
33554432	380	1.27472
33554432	390	1.27953
33554432	400	1.27921
33554432	410	1.2762
33554432	420	1.27868
33554432	430	1.28095
33554432	440	1.27519
33554432	450	1.27681
33554432	460	1.27846
33554432	470	1.27834
33554432	480	1.27987
33554432	490	1.27755
33554432	500	1.27496

5. One thing that the data and graphs show clearly is a performance cap at around 1.28 GigaParticles per Second. Outside of a few outliers, particles are never processed faster than this. Runs where the number of particles is low seem to be very slow, the lowest number of particles only manages to reach 1 GigaParticle per second twice when recorded. The lower particle numbers also seem to show far more variance in performance, while the larger sets of particles seem to be far more constant.

6. The easiest pattern to explain is probably how speeds are more or less constant through the animation. The same number of particles are moving at mostly the same speed the entire time, so the few outliers can probably be explained away as interference from other processes on my computer, load from the performance recording that I wasn't able to remove, or just some odd spikes in performance. I've already mentioned the first two parts in #1 and #3, so the few outliers visible here are likely out of my control. But, there are few, so it doesn't impact the data as badly as it did in Project 6. One pattern that isn't so obvious, however, is the odd variance displayed for the lower particle numbers that isn't present for the higher particle numbers. At first I assumed that all runs no matter how many particles were just that varied, and that the only reason for the larger particle counts being so constant was that a cap of around 1.27 GigaParticles per Second was just a global cap. That is, that were this cap not there, that all curves would be as varied as the smallest few (pictured to the right). However, this doesn't seem to be true, as the particle sizes that are larger than 524288, the smallest set, but still don't reach 1.27 GigaParticles per Second are present and seem to have less variance than the smaller particle sets. In addition, the variance isn't

completely random but rather the spikes and drops in the curve seem to be present in all lines, though with less magnitude the higher the line is. This leads me to believe that the cause of this oscillation is not due to the size of the particles but rather to the way the particles are computed and drawn. In addition, this data shows an oscillating pattern very similar to that of Project 6, the only other project to make use of the GPU. Thus, I believe this pattern to be a result of how the GPU is processing these particles. There are a number of possible causes within the GPU such as CUDA cores being freed up or used to capacity, a consistent pause to prevent race conditions, or something in the OpenCL or OpenGL code causing overhead in drawing the particles. Nothing in the OpenCL code for this project should be causing a pause in the particle drawing, so the culprit must be in either the GPU or in the OpenGL commands called during Display() but that I did not write (that is, part of a library). From what Professor Baily said in the Week 8 notes on Vertex Buffer Interoperability, he is also not sure why it oscillates like that, so why my performance looks like this is presumably not covered in this class. Of the possibilities I listed above, however, I would expect that this is a result of how the OpenCL code is added to the buffer, and how the buffer queue works.



As we expect with most parallel assignments, the earlier runs with smaller datasets actually have worse performance, and this assignment is no difference. The low performance of the smaller particle sets is visible in both the Performance vs Frame graph and the Averages graph (pictured to the left). Smaller datasets end up with a higher percentage of code that is not parallelizable, thus more code has static overhead, thus it is slower. This is normal, and as we can see from the data, around 4194304 particles is where the performance begins to level out as expected.

7. One extra thing that I think is worth mentioning that is not easily pictured with the graphs or even the tables is how long the particle animations lasted. The data gathering was stopped at 500 frames as that is around how long it took the earlier runs to finish, and to make data reporting more reasonable. Had the data collection continued too far past 500, it would skew the results for the lower particle numbers along with being very difficult data to present and understand. But while 500 frames is enough to observe most of the animations, some of the animations of the larger particle sets did not appear to be as finished in 500 frames as the earlier animations did. For the smallest data set of 524288 particles, 500 frames was enough for almost all of the particles to vanish from the screen, having bounced past the obstacles already. But for the larger runs with 33554432 particles, many were still rebounding and on screen when 500 frames were finished. While this information doesn't effect the processing power of the technologies or how efficiently things were written, it is worth noting that in a real world scenario, despite the 33554432 particle run being more efficient, it would take more total time to run than the least efficient 524288 particle run. With such small particles, the inherently non-precise nature of graphics, and the total elapsed time for the entire animation, I would argue that just

being the highest on the chart of performance does not mean that that animation is the most useful or viable, and that a real world graphics scenario would need to better define what is the most desired metric when choosing a particle size, assuming a static particle size is an option.

As for how this affects GPU parallelization in general, I think this project corroborates the week 10 notes that OpenGL is good for the more lightweight graphics-oriented projects as opposed to heavy parallelization tasks. While OpenGL is slightly more friendly and less cumbersome than OpenCL, it appears to have limits below that of what we could reach with OpenCL.