ROBERT CASSIDY – HW1 – Part 4 Performance Analysis

Once I got my N body simulation up and running I tested it with 5000 bodies. (Note: I turned off the additional call to accelerate() that was not needed for the homework). I then tested it by adjusting the number of Threads per block and setting block size to (numberOfBodies / threads) + 1. I got the following results:

Threads per Block	Frames per Second
1	.89
2	1.7
4	3.5
8	6.9
16	13.5
32	26.2
64	29.7
128	30.4

As you can see doubling the number of threads per block basically doubled my performance until it leveled off around 32-64. The reason for this is pretty obvious. If each block can handle up to 32 (or 64 I'm not sure) threads then scheduling fewer will simply be underutilizing the GPU. Without using all the threads available there are tons of ALUs sitting unused because I didn't put enough parallel threads in the block even though I had more threads to schedule.

I also tested adjusting the number of planets as well, however my system crashed after 6000 so I wasn't able to add a lot more Obviously fewer bodies ran faster as well. Having fewer bodies helped us in two ways, one there are fewer threads to compute overall so if we have more threads to schedule than the maximum we can get in flight, this will help us cut down on time. The other place we save is because each body has to compare with each other body we get a linear savings here as well.

As far as the cuda math goes, we could definitely expect the GPU to run faster, particularly on large matrices. The reasons are obvious, matrix operations are incredibly parallel so we're able to perform operations on large portions or the entire matrix concurrently on the GPU while on the CPU we have to perform operations for each index in sequence.