Information - Project 0

General Information

Name: Steven Hanna

NetID: sthanna

Academic Year: Sophomore

Github: steventhanna (http://github.com/steventhanna)

Programming Experience

- Java
- C++
- JavaScript / Node.js
- Python
- Haskell

While these arent real programming languages, I thought I would include them also

- HTML
- CSS
- Git
- Markdown

Unix

I currently run OSX, but in the past I have run GNOME. I do most of my work on the computer either in a text editor or in the command line.

Physics

I took your Physics I course last semeseter.

Math

- Calc I
- Calc II
- Linear Algebra (curent)

- Discrete Math (current)

Physcial Systems

I am interested in simulating a galaxy / solar system, or the Earth with all of its man-made satelites around it.

Note:

Most of the code is written in C++, with the exception of `MaxMathThread`, which is written in C do a threading error.

Documentation was written using the Argot (http://github.com/steventhanna/argot) documentation system.

Convert C to F

It took me a little bit to famillarize myself what is different between C and C++, but once I did some Googling I quickly found what I needed. Most of my issues stemmed from the different flags in `scanf` and `printf`, as I never had to use them in C++ as it uses `cin` and `cout`. Beyond that, it was just simple math to get the correct temperature.

Output to a File Columns

After checking the spec for outputting to a file, everything fell into place nicely. I knew that I had to use the Math header file, and since I just had some practice with using flags from Convert C to F, there were no compilation errors at all. Formatting the columns were a little difficult, but it was something I remembered from `printf` in Java. Geting a graph to work was more difficult, as I was using gnuplot on OSX. After changing my step size, domain, and range, I got the lines to smooth out.

Max Math Operations

The bottleneck on doing opeartions is the standard output, so I am going to avoid outputting until after the second is over. What I did is I got the current EPOCH time and continiously checked in a while loop until the time was 1 + the stored epoch time. I used a counter to store the amount of operations completed. Technically, incrementing a counter is an operation, so I just used `counter++` as my operation. The highest operation count that I got was `4318367.000000`, while the lowest was `255331.000000` out of 10 trials. Currently running on an Intel I7 at 2GHZ.

When using 8 threads, the operation count got to 9461140.000000, and never went below 2000000.

My results are not entirely accurate, as I had many other applications running in the background. Additionally, OSX can be quite greedy in terms of resources at times, so that can account for issues as well. In comparison, a similar I7 can run 100,000 millions of instructions per second. source (https://en.wikipedia.org/wiki/Instructions_per_second#Millions_of_instructions_p