The n-bodies problem, which is a problem bodies are created that resemble planets, had two major components then start the parallel and sequential versions of the program. The two versions were so that they could be compared to each other, and found out which one is faster and by how much. The data showed that the parallel version was faster than the sequential. This should be the results because the work is divided. This puts an emphasize on comparing the number of threads to each other to find out which is more effective.

The timing was done was tested on Cambridge with the number of threads being the only number differing; for two tests one being 500 bodies (figure 1) and another 1000 bodies (figure 2). The timing is starts after all the bodies are populated, and ends when the number of iterations desired is over. This means that each time is was ran that it was given a similar start time. The data showed that running more than one thread is faster than just one. This is because the when there were multiple threads number of bodies that were calculated was sliced up, meaning that some bodies were calculated at a similar time as others. Each test was done three times to find a consistency (see Figure 3). The timing would have met nothing if the bodies were not the same or react in a similar motion, so test had to be done to make sure that bodies were acting the way they should.

Figure 1

Figure 2

The GUI was made to give a visual of what is happening, and uses the same calls as the console version but displays it visually. This, GUI, was used to tell that the collision was correct. Another way is that the collision was tested was using the console prints outs. In the print, out it prints out positions of the bodies that collided. This data can be used to solve if the all radiuses are the same to make sure that a collision did happens manually. Then, the velocity and the force could be solved by using the position points for each body. The force depends on how close a body is to another. Bodies that are getting closer change from their current velocity, thus showing a force by another body. The GUI shows a body with a curve path, and this gets worse the lighter the body is compared to the other body. Then the velocity can be proven correct by using the points that are printed to the console, because a slope can be determined by two points from the same body. To emphasize this using one body and the velocity is consistent throughout the iterations. The GUI also had a timing too because how close it is to the console version of the program.

The console version’s timings were done to test how effective the parallel threading is, so the GUI was also done because it uses the same simulation as the console version. The GUI’s timing was slower than the console version, but in the GUI there is a thread.sleep(50) which for 500 bodies and 100 iterations (see figure 4) and gave similar results to console version. This can because it does not have to print the position of all the bodies while the console version does. Meaning that it can be possible that the sleep is less than the system printing all the bodies locations. The versions of the tests ran at different times.

The expected results for the console happened. This is because more threads the faster, like slicing up the work. The results differed in the GUI though, could mean that there needed to be larger timing to see the difference because they were close. That is each test was ran at least three times, so that it can limit those factors from being effective..

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iterations | Size (radius) | Mass | Threads | DeltaT | TimeEnd (Total Time) milliseconds | Bodies |
| 100 | 100 | 100 | 1 | 0.5 | 15703 | 1000 |
| 100 | 100 | 100 | 1 | 0.5 | 11080 | 1000 |
| 100 | 100 | 100 | 1 | 0.5 | 13351 | 1000 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **13378** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 1 | 0.5 | 5896 | 500 |
| 100 | 100 | 100 | 1 | 0.5 | 5985 | 500 |
| 100 | 100 | 100 | 1 | 0.5 | 6420 | 500 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **6100.333333** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 2 | 0.5 | 11971 | 1000 |
| 100 | 100 | 100 | 2 | 0.5 | 10727 | 1000 |
| 100 | 100 | 100 | 2 | 0.5 | 10582 | 1000 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **11093.33333** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 2 | 0.5 | 5623 | 500 |
| 100 | 100 | 100 | 2 | 0.5 | 4586 | 500 |
| 100 | 100 | 100 | 2 | 0.5 | 5516 | 500 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **5241.666667** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 3 | 0.5 | 12319 | 1000 |
| 100 | 100 | 100 | 3 | 0.5 | 13559 | 1000 |
| 100 | 100 | 100 | 3 | 0.5 | 12105 | 1000 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **12661** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 3 | 0.5 | 6157 | 500 |
| 100 | 100 | 100 | 3 | 0.5 | 5223 | 500 |
| 100 | 100 | 100 | 3 | 0.5 | 5629 | 500 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **5669.666667** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 4 | 0.5 | 12999 | 1000 |
| 100 | 100 | 100 | 4 | 0.5 | 12128 | 1000 |
| 100 | 100 | 100 | 4 | 0.5 | 13174 | 1000 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **12767** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 4 | 0.5 | 4836 | 500 |
| 100 | 100 | 100 | 4 | 0.5 | 5222 | 500 |
| 100 | 100 | 100 | 4 | 0.5 | 5130 | 500 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **5062.666667** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 8 | 0.5 | 12744 | 1000 |
| 100 | 100 | 100 | 8 | 0.5 | 11560 | 1000 |
| 100 | 100 | 100 | 8 | 0.5 | 12498 | 1000 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **12267.33333** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 8 | 0.5 | 5365 | 500 |
| 100 | 100 | 100 | 8 | 0.5 | 5584 | 500 |
| 100 | 100 | 100 | 8 | 0.5 | 4812 | 500 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **5253.666667** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 16 | 0.5 | 11950 | 1000 |
| 100 | 100 | 100 | 16 | 0.5 | 13477 | 1000 |
| 100 | 100 | 100 | 16 | 0.5 | 12129 | 1000 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **12518.66667** |  |
|  |  |  |  |  |  |  |
| 100 | 100 | 100 | 16 | 0.5 | 6299 | 500 |
| 100 | 100 | 100 | 16 | 0.5 | 5351 | 500 |
| 100 | 100 | 100 | 16 | 0.5 | 4819 | 500 |
|  |  |  |  |  |  |  |
|  |  |  |  | Average | **5489.666667** |  |
|  |  |  |  |  |  |  |

Figure 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Average |  |
| 1 | 500 | 73755 | **75599.7** |  |
| 1 | 500 | 74293 |  |  |
| 1 | 500 | 78751 |  |  |
|  |  |  |  |  |
| 2 | 500 | 91827 | **91728** |  |
| 2 | 500 | 92994 |  |  |
| 2 | 500 | 90363 |  |  |
|  |  |  |  |  |
| 3 | 500 | 93279 | **94174** |  |
| 3 | 500 | 95361 |  |  |
| 3 | 500 | 93882 |  |  |

Figure 4