/\*---------------------------------------------------

\* Tanner Bernth

\* Robert Walters

\* Project 1 Report

\*---------------------------------------------------\*/

/\*---------------------------------------------------

\* **Introduction**

\*---------------------------------------------------\*/

These programs involve solving the n-body problem, which simulates the evolution of the galaxy. Each body in the galaxy has a mass and an initial position and velocity. Gravity causes acceleration and movement between these bodies. The n-body system simulates the evolution by calculating the forces on every body at each time step. To make the programs more interesting, elastic collisions were included, meaning the total energy remains the same after the two objects collide.

To simulate the n-body problem, we created both a sequential and parallel version in order to see the differences in performance for each. Additionally, we created a graphical representation of the programs in order to see the results of the calculations and the collisions.

/\*---------------------------------------------------

\* **Programs**

\*---------------------------------------------------\*/

For our n-body problem solutions, both the sequential and parallel versions use the same GUI, planet, and point classes, while the parallel version has additional classes, which make it parallel. For the initial conditions, we just used random positions, velocities, forces, and masses. Using random initial conditions allows for many varying types of interactions between the bodies. The biggest enhancement to the solution is the GUI, using the java.swing class, that makes it very simple to visualize what is happening in the code. Since the GUI is in use during the calculations, it slows the execution time somewhat, but the option to hide the GUI is available as well. The input for the programs includes the number of threads (which does nothing for the sequential version), the number of bodies, the size of the bodies (although we simply made the sizes of all bodies randomly), the number of time steps, and either 0 or 1 for whether or not to use the GUI.

The programs time the duration of the calculations as soon as all of the bodies and threads have been initialized. After the calculations are finished, the computation time and the number of collisions is output and the final positions and velocities of all of the bodies is input into a file named results.txt.

/\*---------------------------------------------------

\* **Verification**

\*---------------------------------------------------\*/

The programs were much easier to verify they were correct with the use of the GUI. The GUI allowed us to visualize the changes that were happening as a result of the code. When two bodies collide, they are supposed to be launched in a different direction based on the forces and velocities of the two bodies, which is visible from our trial runs. Two bodies colliding together also maintains that they are supposed to elastic collisions because a faster body may slow down after colliding, but the other body will speed up accordingly. Using the GUI, we see that the collisions are resolved correctly because the bodies only have changes in their velocities when they directly touch another body. They can come very close to another body and have no effect as a result of a collision.

/\*---------------------------------------------------

\* **Timing Experiments**

\*---------------------------------------------------\*/

/\*---------------------------------------------------

\* **Other Experiments**

\*---------------------------------------------------\*/

/\*---------------------------------------------------

\* **Conclusion**

\*---------------------------------------------------\*/