INF1A FP Programming Competition 2024

Abdulrahman Hesham Al Bin Ali

November 2024

Project Description:

This program is a simulation of an n-body physics system with customizable gravity functions (using temperature, size, or even velocity!) as well as elastic and simultaneous collision behavior, all wrapped in a little toroidal world. There also exists an 'analysis' mode to evaluate the accuracy of given numerical integration methods.

The simulation can be run using any of the following numerical integration methods, and the analysis can be run using any of the following as a control:

- Euler's Method
- The Runge-Kutta 4th Order Method
- Velocity Verlet Integration
- Leapfrog Integration
- The 4th Order Yoshida Integrator

As you go down the list, the accuracy increases while the performance roughly decreases.

The simulation features several preset scenarios that demonstrate different gravity and collision effects, such as:

- Basic gravity between two bodies
- Basic gravity between three bodies
- A large planet with a ring of non-gravitational particles
- Gravity between three bodies that applies a translational as well as an angular force
- Attraction and repulsion based on temperature differences

- Extremely strong gravity between four bodies, as well as a counterclockwise angular force
- and more!

How to Run:

The program can be run from the command line by the following commands:

```
cabal update
cabal build
cabal run
```

directly. When prompted, choose the mode (simulation or analysis), a demo (or make your own!), and then choose the numerical method and any other required variables. The simulation will now run, and you can observe it in real-time.

Dependencies

The program uses the following external Haskell libraries:

- gloss for the graphics rendering
- vector for efficient data structures
- parallel and deepseq for parallelization of computations

These dependencies are managed using Cabal. The .cabal file is included in the project directory.

Acknowledgments

The implementation and design choices are my own, but I've built upon the concepts and ideas presented in the following resources during the development of this project:

- Gloss library documentation[1]
- Vector library documentation[2]
- Euler's Method[3]
- The Runge-Kutta method[4]
- Verlet Integration[5]
- Leapfrog Integration and Yoshida Integrators[6]

[1] http://hackage.haskell.org/package/gloss
[2] http://hackage.haskell.org/package/vector
[3] https://en.wikipedia.org/wiki/Euler_method
[4] https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods
[5] https://en.wikipedia.org/wiki/Verlet_integration
[6] https://en.wikipedia.org/wiki/Leapfrog_integration

Graphics

I have tried for at least half a dozen hours to record some demo videos of the simulation in action. However, my device has been unable and unwilling to keep the frame-rate of the simulation stable enough to be smooth and accurate when under the stress of recording a window.

I have attempted to:

- Manually record using QuickTime
- Automate appleScript to record QuickTime
- Manually record the screen using OBS to crop the simulation window out later
- Automatically record the simulation window using OBS
- Export each frame of the simulation onto the disk to read later

All my attempts have either completely failed to succeed or were too laggy to be salvageable.

However, I have included some still screenshots of the simulation in action, as well as a few graphs in analysis mode.
I would be extremely grateful if this is taken into consideration, and that the actual visuals of the program are assessed based on the demos ran on the judges' devices, and not on the lack of demos from mine.