Proposal: Newtonian Particles

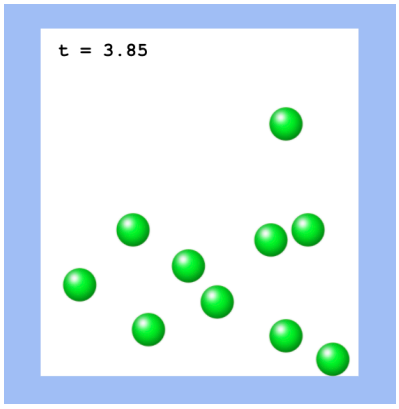
Parallel Functional Programming (Fall 2021) | Stephen Edwards | Columbia University

Written By: Nathan Cuevas (UNI: njc2150)

Overview

I plan on doing a project that uses a graphics library to animate the transient movement and collision of particles within some rectangular closed container. The container will contain n particles, each particle will have some initial position and velocity, the motion of the particles will be governed by basic newtonian mechanics. The particles can bounce off of each other and the walls of the container. The particles will also be subject to gravity and friction/loss of kinetic energy with each collision, so eventually the system will reach a steady state. This problem benefits from parallization because the computation for each particle can be divided in threads.

Below is an example of what the graphics window might display at a given time:



Note that I have implemented something similar (using MATLAB) before with a single particle. This time around I will be implementing it with the additional challenge of multiple particles, using functional, and parallelization.

General Mathematical Model

The position and velocity of a particle at any time t will be represented by a vector $\vec{s^{(t)}} = (x^{(t)}, y^{(t)})$. and $\vec{v^{(t)}} = (v_x^{(t)}, v_y^{(t)})$. We can calculate these values using:

$$x^{(t)} = x^{(t-1)} + v_x^{(t-1)}\Delta t$$

$$y^{(t)} = y^{(t-1)} + v_y^{(t-1)}\Delta t$$

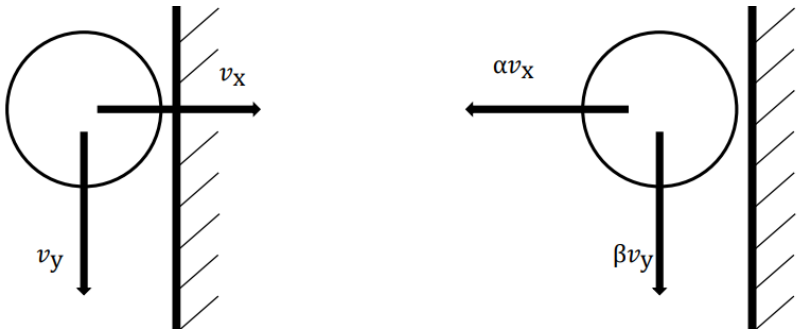
$$v_x^{(t)} = v_x^{(t-1)}$$

$$v_y^{(t)} = v_y^{(t-1)} - g\Delta t$$

Where Δt is the time step. If during Δt there is a collision with the wall, the following equations will be used to calculate the new values for position and velocity. See figure and equation below for the velocity calculation after colliding with the right-hand wall.

$$v_x^{(t)} = -\alpha v_x^{(t-1)}$$

$$v_y^{(t)} = \beta v_y^{(t-1)}$$



Where $0 < \alpha < 1$ is the the loss coefficient for kinetic energy and $0 < \beta < 1$ is the loss coefficient for friction.

In the case that there is collision between multiple particles, the new velocity vectors can be determined using conservation of linear momentum and conservation of energy:

$$\sum m\vec{v_{\text{before}}} = \sum m\vec{v_{\text{after}}}$$

$$\sum \tfrac{1}{2} m\vec{v_{\text{before}}}^{\,2} = \alpha^2 \sum \tfrac{1}{2} m\vec{v_{\text{after}}}^{\,2}$$

Potential Challenges

Here are the challenges that I forsee, please comment if you have suggestions:

- Using a graphics library might be annoying in Haskell
- Numerical instability could cause some odd physics. I could probably fix this by tweaking the time step.
- Many particles colliding at the same time (say at the bottom of the container as $t \to \infty$) can lead to odd physics.