MATH 198 Project Preproposal

Professor George Francis
Brian Campbell-Deem

October 16th, 2015

Abstract

Official programming has begun in VPython, beginning with the particles-in-a-box simulation. Current program is running two particles which bounce correctly off the walls but not off of themselves, physical laws have not yet been taken into account. Current structure will need to be reformatted to a more efficient method if the simulation is the account for n particles.

1 Code Progress

I have begun writing the first code for the first piece of my project, the simulation of particles in an enclosed space. Following a guide[1] I was able to find the general functions necessary, namely drawing spheres and boxes, as well as properties of these objects (they're mutable and adding extra properties is extremely straightforward through VPython). The checks to bounce off of the wall are straightforward checks for its position, however with six walls that means n particles would have $6n$ blocks of code solely for bounce checks, which may be inefficient later on.

The not-so-easy part is how the particles bounce off of each other; it is an obvious application of conservation of momentum and energy however I have not currently found a simple way to accurately depict it that isn't extremely messy. Currently, the program is running a simple method: cycle the velocities ($v_x \mapsto v_y$, $v_y \mapsto v_z$, etc.) which has the benefit of not violating either conservation law, but isn't exactly descriptive of the real physical case. In any case, to show bounces actually are happening, I have the program draw a blue sphere at the point when it occurs. Due to the inaccuracies of current bounce methods, sometimes the blue spheres smear.

2 Plans For Next Week

I hope to be able to get an accurate representation of the bounce for two implemented. I have a current idea about using a vector tangent to both particles,

[1] http://vpython.org/contents/docs/VPython_Intro.pdf

which they will bounce away from at equal angles (with checks to make sure the correct momenta are distributed). I also want to implement visible velocity vectors for the particles to show how they change upon bouncing. Additionally, I want to start working on another piece of the project, since it's to be a collection of many programs, and I believe that will be either the simple kinematics program, which I could knock out fairly quickly, or an adaptation of the doublependulum.py example.