# MATH198 Weekly Update

Professor George Francis
Brian Campbell-Deem

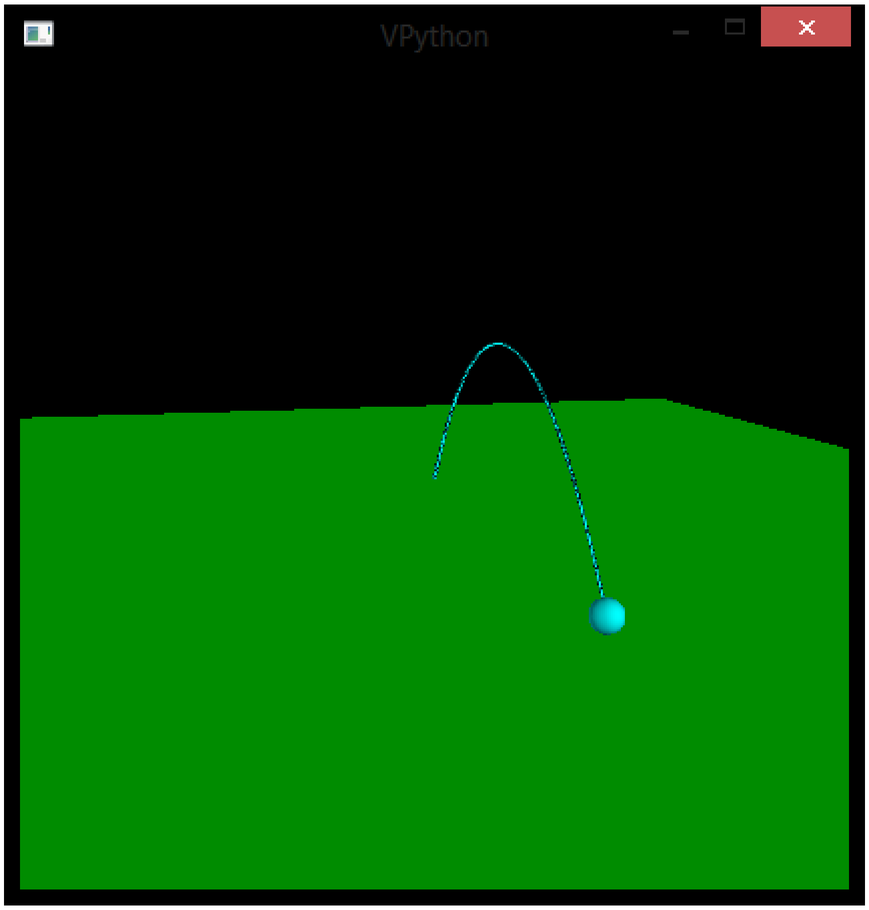
October 23$^{\text{rd}}$, 2015
Week 8

**Abstract**

A simple 3-D kinematics simulator has been built which displays a "thrown" object traveling at a randomized initial velocity, and a simple ground is drawn which scales to how far the object is expected to travel. Animation speed has been attempted to reflect real-time, and displays accuracy to within 0.5%.

## 1 Code Progress

I began writing the code to simulate a simple 3-D Kinematics system, i.e. a the trajectory of a thrown ball. Following the previous particle example, drawing the ball and ground as well as animating the ball's position were easy. I currently have the velocity values $v_x$ and $v_z$ (which lie in the plane of the ground, as drawn by VPython) to be random values in the range $[-40, 40]$; to have the ball thrown up (and remain up for a decent time) $v_y$ is limited to the range $[20, 40]$. The check for the size of the ground is drawn based on the absolute magnitude of the velocity, making sure the square is always big enough but never overly large. I discovered an attribute of the sphere which allows me to easily display a trail, which enhances the kinematics simulation by showing its path of travel.
In order to get the timing correct, I have a variable to track how long the program runs and compares it (percent error calculation) to the expected time the ball should last, as determined by real physics. To aid in the correspondence of timing, I have the infinitesimal time step $dt = 0.001$ and the animation rate $r = 1000$ as inverses of each other, so that the animation will update approximately in real time, ignoring external factors. Problematically, drawing extra objects causes the run-time to slow down meaning the animation rate needs to vary based on what is being drawn. A stationary velocity vector, for example, which displays the ball's velocity but remains locationally fixed causes the timing error to triple to around 1.5%.



## 2 Plans For Next Week

I want to work on animating multiple objects for the kinematics.py example while also retaining a real-time display. With multiple objects, the example could be extrapolated to a more interesting application, such as explosion simulation or fireworks display. I also want to work on the realism of the particle bounce in bs.py, which I didn't get around to this week.
For another new program to put together for my project, I want to start working on the double pendulum next, whose physics are easily described with the Lagrangian approach. Thereafter, adding at least a third pendulum should not be drastically difficult, although run-time is again a problem.