Computer Homework 16 Drift Veolcity

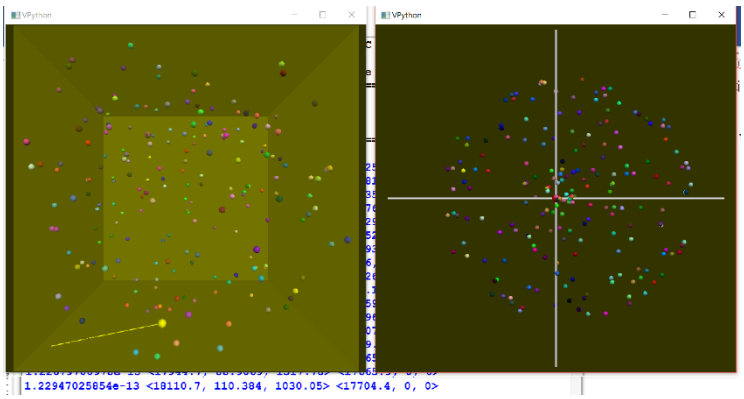
video: https://goo.gl/pgIu5K

In the program, *N*=400 (if your computer is slow, use smaller number) charged particles, each with mass *m* and charge *q*, are in a cubic box of length 2*L*. The velocity of each particle have the same magnitude *vrms*, but with a random direction. With periodic boundary conditions, each particle that hits a wall of the container will appear on the opposite wall with the same velocity. In two different displays, the position and velocity of every particle are shown. In the program, the average velocity *vd* over all particles over all time is found and is also shown in the velocity display.

Homework:

(1) Within each simulation time step *dt*, each particle has a probability (*prob*) to encounter a collision. After the collision, the velocity of the particle has the same magnitude *vrms* (due to thermal equilibrium, the speed does not change) but with a new random direction. Find the collision time:
tau = (total time *t***N*) / (total collision number of all particles).

(2) Follow (1). If there is an electric field *E* to accelerate all particles, find the averaged velocity *vd*, now is the drift velocity. Is it close to the theoretical value *q***E***tau*/*m*?




```
from visual import *

prob = 0.0008
N, L = 400, 10E-9/2.0
E = vector(1500000,0, 0)
q, m, size = 1.6E-19, 1E-6/6E23, 0.1E-9
t, dt, vrms = 0, 1E-16, 100000.0
atoms, atoms_v = [],[]

scenev = display(width=600, height=600, x = 600, fov = 0.01, range = (1.5*vrms,1.5*vrms, 1.5*vrms), background=(0.2, 0.2,0))
scene = display(width=600, height=600,background=(0.2,0.2,0))
container = box(display=scene, length = 2*L, height = 2*L, width = 2*L,    opacity=0.2, color = color.yellow )

pos_array = -L + 2*L*random.rand(N,3)
theta = pi*random.rand(N,1).flatten()
phi = 2*pi*random.rand(N,1).flatten()
v_array = transpose(vrms*array([sin(theta)*cos(phi),sin(theta)*sin(phi), cos(theta)]))
for i in range(N):
    atoms.append(sphere(display=scene, pos=pos_array[i], radius = size, color=random.rand(3,1)))
    atoms_v.append(sphere(display=scenev,pos=v_array[i],radius = vrms/50, color=random.rand(3,1)))

vd_ball = sphere(display=scenev,pos=(0,0,0),radius = vrms/30, color=color.red)
x_axis = curve(display=scenev, pos=[(-1.4*vrms,0,0), (1.4*vrms,0,0)], radius=vrms/100)
y_axis = curve(display=scenev, pos=[(0,-1.4*vrms,0), (0,1.4*vrms,0)], radius=vrms/100)

vv = vector(0, 0, 0)
while True:
    t += dt
    rate(10000)

    pos_array += v_array*dt
    outside = abs(pos_array) >= L
    if outside[N-1, 0] or outside[N-1, 1] or outside[N-1, 2]: atoms[N-1].retain = 0
    pos_array[outside] = - pos_array[outside]
    vv += vector(sum(v_array,0)/N)
    if int(t/dt)%2000 == 0:
        tau = 0                          #to be found
        print tau, vv/(t/dt), q*E*tau/m
        vd_ball.pos = vv/(t/dt)

    for i in range(N): atoms_v[i].pos, atoms[i].pos = v_array[i], pos_array[i]
    atoms[N-1].retain = 2000
```