

HOMEWORK #6, PHY307/607, Fall 2002 – Due by start of class Oct. 1

1. More fireworks. Improve upon last week's fireworks program. Instead of random initial positions, give each spark a position at (0,0,0) and a **radius** of 0.2. After making the sparks, add a loop to give each spark a random *velocity vector*. Then update the locations using **mymodule.py** from Lab #6.

```
from mymodule import *

scene.autoscale = 0

## Build up a list of sparks by appending to the
## end of an initially empty list.
sparks = []
for i in range(100):
    sparks.append(sphere(radius=0.2))

## Initialize the velocity of each spark
for s in sparks:
    s.vel = vector(random()-0.5,
                   random()-0.5,random()-0.5)
    s.vel = s.vel/mag(s.vel)

## Set the strength of gravity and the time step.
grav = vector(0,-0.1,0)
dt = 0.05

## Main loop - repeat forever
while 1:
    rate(20)
    ### PUT IN BELOW WHAT YOU NEED TO STEP
    ### THE LIST OF SPARKS, UPDATING THEIR
    ### POSITIONS - SHOULD TAKE TWO LINES: A LOOP
    ### OVER sparks AND THE FUNCTION TO MAKE A
    ### PARTICULAR SPARK FALL
```

Submit the following:

- ?? Completed code.
- ?? An explanation or guess what the line `s.vel=s.vel/mag(s.vel)` does. Can you find the mag function under the VPython documentation? What do you see if instead you leave that line out?
- ?? An explanation of what the line `s.vel = vector(...)`, does.

HOMEWORK #6, PHY607, Fall 2002 – Due by start of class Oct. 1

It is useful to be able to draw images directly. See if you can install the Python Imaging Library (www.pythonware.com/products/pil) (put the DLL's etc. into your Python22 folder on your own computer or make it accessible by modifying the classpath under IDLE) and if you can get the following (**logisticpic.py**) to work:

```
import Image, ImageDraw

mx = 600
my = 400
## 1-bit, Black/White image
im = Image.new(mode="1",size=(mx,my))

amin = 3.8
amax = 3.9
xmin = 0.1
xmax = 0.5
awidth = amax-amin
xstep = (xmax-xmin+0.)/(my-1.)
for ia in range(mx):
    xA=0.25
    a = amin+ awidth * ia/float(mx)
    for k in range(4000):
        xA = a * xA * (1.-xA)
    for k in range(2048):
        xA = a * xA * (1.-xA)
        if (xA >xmin and xA < xmax):
            ypos = (my-1) - int((xA-xmin)/xstep)
            im.putpixel((ia,ypos),1)
    if (ia % 10 ==0):
        print ia

im.save("logisticpic.jpg")
```

Once you have this working, indicate how this program works by marking it up with comments. For example, why is there a "+0." in the **xstep** initialization? Submit your marked up version.

Colorize the diagram:

- ?? Set the mode to "RGB" rather than "1"
- ?? Make a list **colors** of 7 colors: tuples with 3 integer entries from 0 to 255 in each entry (RGB strengths out of 256.)
- ?? Replace '1' as the color in put.pixel with a selection from your color palette. Say by the (k%7)th color? View a larger portion of the bifurcation diagram (larger a and x range.)