

PHY307 HWK ASSIGNMENT #5, due Sept. 24, 2002:

This homework will give you some practice in using lists for animation and in making plots.

Problems:

1. FIREWORKS, CHAPTER 1. Over a few homeworks, you'll put together the elements of a fireworks show using Python. This simulation will have some physics in it (gravity and friction) and will rely on your programming knowledge. The major new element you will use in this first step is the **random** library, which generates "random" numbers.

- a. Try out the following program:

```
from random import *
for j in range(12):
    print random(), random(), random()
```

The function **random()** generates a random number. This program will print out 12 lines, each line with 3 random numbers. What appears to be the upper and lower bound for these random numbers, that is, what range are they chosen from?

The program you will now write has much of the structure of the "What does this do?" program from Tuesday's lecture.

- b. Write part 1 of the program – the generation of 100 randomly located spheres. Here is an outline:
 - i. Save your empty program as **fireworks1.py**.
 - ii. Import everything from **visual**.
 - iii. Import everything from **random**.
 - iv. Turn off autoscaling.
 - v. Then start an empty list, named **sparks**.
 - vi. Use a for loop to append spheres to the end of the list. It should look very similar to the first **for** loop (two lines long) in "What does this do?", except
 1. You are making spheres, not rings, and the name of your list is **sparks**, not **objlist**.
 2. ALSO, give your sphere a random (x,y,z) position and a radius of 0.1. For example, to give a box a random size, you might call **box(width=random(), height=random(), length=random())**. Do something similar with the **x**, **y**, and **z** coordinates of your sphere.
- c. Record what you get when you run this program. Include a snapshot.

- d. Write part 2 of the program. This next loop will animate the sparks. At the end of **fireworks1.py** add the following lines:

```
for k in range(500):
    rate(20)
    for s in sparks:
        s.pos = s.pos * 1.005
```

- e. Run this program, record what you see in words, include a snapshot in your report. **EXPLAIN in words what the 4 lines from part (d) DO.**
- f. [OPTIONAL] If you like, you can fancy up the program. Include code changes, snapshots, and discussion. Can you make the expansion more symmetric (try **random()-0.5** in place of **random()**)? Can you color the spheres uniformly or randomly? Later, we will add trails, more realistic dynamics, and a fireworks container.
2. Make a bifurcation diagram for the quartic map $x = a * (1-x)^2 * (1+x)^2$, for a in the range $0.6 < a < 1.4$, in steps of no more than 0.01 (unless you freeze or crash, in which case note that and try a larger step size for the parameter a .) This will require very little change from the “Curves of Chaos” lab’s **bifurcatelogistic.py** – save your program as **bifurcatequartic.py**. Descriptively, compare the diagram with the bifurcation diagram for the logistic map. If you want to see more detail, narrow down the range of a and take finer steps.