PHY607 HWK ASSIGNMENT #5, due Sept. 24, 2002:

In the logistic map, there is a single variable, *x*, and a single control parameter, *a*. So the orbits in *x* can be plotted vs. *a* to give a two-dimensional bifurcation diagram that shows periodic islands and the cascades to chaos. However, if you have more variables and parameters, this doesn't work. Consider the Henon map, which has two variables, *x* and *y*, and two parameters, *a* and *b*:

$$(x,y) \text{ ? } (1 + by - ax^2, x) \qquad \text{[Henon map]}$$

The Henon map, unlike the logistic map, is invertible for *b* ? 0, and so is of special interest for modeling dynamical systems, like the orbits of planets. But now the bifurcation diagram is a 4D object!

How do we visualize this? There are several possible methods, involving color, animation, etc. The most straightforward is to step through 3D slices of the 4D diagram.

In general, one would want to make a point-set generator and a separate viewer. But since the Henon map is fast to execute and I don't want to start Python input and output, yet, why don't you make an integrated generator/viewer?

So, you will be mapping the 4D set of (*a*,*b*,*x*,*y*). I suggest you make points at locations (*b*,*x*,*y*) while varying *a*, so that *a* will be like the 4^{th} dimension, time.

Here is an outline on how to approach this:

1. Make a list of, say, 2000 small cubes of edge length 0.005. This set of cubes will be manipulated to draw each slice.
2. Loop on *a*, in some range (I suggest you explore values of *a* roughly from 1 to 1.7, but you might want to narrow in on some ranges.)
    a. In this loop, have a "pause" that allows the viewer to look at the 3D slice, before going to the next slice. A handy pause feature is **scene.mouse.getclick()** that waits for the left mouse button to be clicked.
    b. Set a cube counter to 0.
    c. Loop over 20 values of *b* (I suggest values ranging from about –0.1 to 0.3.)
        i. Set a value of *x*,*y* and iterate the Henon map a couple of hundred times. Remember that you must update *x* and *y* simultaneously. You can do this using temporary variables or the convenient assignment format:
        x,y = <expression 1>, <expression 2>
        ii. Now that you have warmed up *x*,*y*, repeat the map 100 more times,
            1. - each time setting the coordinates of a cube in your cube list to be (*b*,*x*,*y*). E.g., to set the **radius** of cube number 43 in a list of cube called **sparks**, you would write **sparks[43].radius = newradius.**
            2. - and each time incrementing the cube counter.

So try this out, varying your range of *a* and *b*, seeing how many cubes you can conveniently draw at once (can you do 4000 or more?), and scaling *b* to make the picture more accessible (for narrow ranges of **b**, you might want to set the cube position to be **scaleb * b**, to spread things out. What interesting slices of the 4D set can you find? Can you find a period-5 attractor, for example?