PHY307 HWK ASSIGNMENT #11, due by the start of class Nov. 5, 2002:

This homework will give you some practice in running and interpreting Monte Carlo simulations.

Problems:

1. ESTIMATING VOLUME. You wish to find the volume of a sphere minus a cylindrical hole. The sphere has radius 1 and the hole has radius 0.5. The hole goes through the center of the sphere.
    a. Sketch out what this volume looks like, including a box that contains the sphere.
    b. Download the program **MCsphereshow.py** from the class code page.
    c. Save the program as **MCspherecylindershow.py**. You will be changing only one line of code.
    d. Modify this program. Change the condition for deciding whether the point is within the volume of interest:
        i. The point needs to be in the sphere *and*
        ii. it needs to be outside of the cylinder (the square of the x-component of the vector plus the square of the y-component must be greater than the square of the radius. The x-component of a vector **vec** is found through the attribute **x**, i.e., **vec.x** is the x-component of **vec**.)
    e. Run the program for about 5000 points, taking a snapshot to include in your homework, to verify that you apparently have the correct shape.
    f. Run the program 10 times using 50 points and 10 times using 5000 points. Record the volume estimate each time. Include this data in your report.
    g. Estimate the size of the fluctuations among the 50 point measurements. Do the same for the set of 5000 point measurements. How much do the fluctuations differ between the two sets of data, approximately?
2. PLAYING GAMES. Modify **MCpingpong.py**, available on the web, to repeat the match a number of times – all you need to do is add a single **for** line and indent the appropriate set of lines of code, so that you repeat that set of lines 100 times. Run 100 matches for **Askill** values of 0.95, with **Bskill** fixed at 0.9. What is the approximate chance of A winning a match? Repeat this, keeping **Bskill** fixed, for **Askill = 0.9** and **0.85**. Include your modified code, show sample output, and summarize your results**.**
3. [FOR PHY607 STUDENTS] Computing a cross-section. Find the probability that an asteroid fired at the Earth will strike the Earth. Start an asteroid about 10 Earth radii from the Earth, with a velocity in the positive x-direction; the Earth is centered at the origin. The initial y and z offsets are to be uniformly chosen between –5 and +5 Earth radii. Use 3 different values of initial velocity for the asteroid, with a minimum velocity above the escape velocity from that distance. For each value of the velocity, what is the probability that the asteroid will impact the Earth (check if it goes off to infinity instead by setting some criterion for a "miss", e.g., if the distance to the center of the Earth exceeds 12 Earth radii.) Take the Earth to be spherical, use proper units, compare the apparent cross-section of the Earth with its actual cross-section, include your code and output in the submission.

```
## Monte Carlo simulation to estimate the volume of a sphere.
## Points are generated in a box of side 2, centered at the
## origin. The box has volume 8. The sphere is centered at the
## origin. It has radius 1. If the random point lies within the
## sphere, count as "in". Ratio of ins to total number of points
## gives the ratio of the volume of the sphere to the box. So
## the volume of the sphere is the in/total ratio times 8.

from random import *
from visual import *    ## we are not drawing, but will use mag()

## define a function that returns a random number between -1 and 1
def MCcoord():
    x = random()      ## make a random number between 0 and 1
    x *= 2.0          ## double puts into range 0 to 2
    x -= 1.0          ## subtract 1 to put in range -1 to 1
    return x

print "This program uses a Monte Carlo method to estimate the"
print "volume of an object (initially, a sphere, but will change.)"

num_points = input("How many points do you want to throw? ")

num_in = 0.0
for i in range(num_points):
    MCpoint = vector(MCcoord(),MCcoord(),MCcoord())
    ## here is the MC test - the decision/score
    pointsize=0.04
    if mag(MCpoint) < 1:
        s = sphere(radius=pointsize,pos=MCpoint,color=color.red)
        num_in += 1.0

print "Number within the volume is ", num_in
print "Estimated volume is", 8.0 * num_in / num_points
print "Volume of sphere of radius 1 is", 4.0 * 3.1415926535 / 3.0
```

```
    ### Monte Carlo table tennis

    ### Assume player A wins fraction Askill of the serves,
    ### similarly for B - skill values are in range [0,1]
    ### Scoring: players serve twice in a row, switch initial
    ### server at the end of a game. Must win by 2. When reach
    ### deuce (10-10), alternate serves one at a time. 5 games
    ### in a games. Here, we will have a rule change: simply
    ### alternate serves.

    from random import *

    Askill = 0.9
    Bskill = 0.9
    Amatches = 0
    Bmatches = 0

    #start a match - A always serves first, each match
    Agames = 0
    Bgames = 0
    gameserver = "A"

    # loop over games in match
    for game in range(5):
        Apoints = 0
        Bpoints = 0
        gamerecord = ""
        server = gameserver

        while (abs(Apoints - Bpoints) < 2 or (Apoints < 11 and Bpoints < 11)):
            if server == "A":
                if random() < Askill:
                    Apoints += 1
                    gamerecord += "A"
                else:
                    Bpoints += 1
                    gamerecord += "B"
                server = "B"
            else:
                if random() < Bskill:
                    Bpoints += 1
                    gamerecord += "B"
                else:
                    Apoints += 1
                    gamerecord += "A"
                server = "A"

        print gamerecord, "Game score: A=", Apoints, "B=", Bpoints
        if Apoints > Bpoints:
            Agames += 1
        else:
            Bgames += 1
        if gameserver == "A":
            gameserver = "B"
        else:
            gameserver = "A"
        print "Games: A=", Agames, "B=", Bgames
        # end of loop over games in a match

    # summarize results of match
    if Agames > Bgames:
        print "A wins match"
        Amatches += 1
    if Bgames > Agames:
        print "B wins match"
        Bmatches += 1

print "After all matches, A has won", Amatches, "and B has won", Bmatches
```