
In this problem you will study the motion of a planet around a star. To start with a somewhat familiar situation, you will begin by modeling the motion of our Earth around the Sun. Write your answers to the questions on this sheet.

Planning
(a) What initial momentum should you use for the Earth? Note that the Earth orbits the Sun once per year and that there are approximately $\pi \times 10^7$ seconds in a year.
(b) What is a reasonable value for the time step to use? If it is too large, you will have accuracy problems, while if it is too small, the program may not run fast enough.

Circular Orbits
(c) Run your program and check to see that the Earth returns to its starting point, and closes the orbit. Do this by displaying the trail of the Earth as it moves.

Computational Accuracy:
(d) As a check on your computation, is the orbit a circle as expected? Run the calculation until the accumulated time, $t$, is the length of a year: does this take the Earth around the orbit once, as expected? What is the largest step size, $dt$, you can use that still gives a circular orbit and the correct length of a year? What happens if you use a much larger step size?

Non-circular Trajectories:
(e) Set the initial speed to 1.2 times the Earth’s actual speed. Make sure the step size is small enough that the orbit is nearly unaffected if you make the step size smaller. What kind of orbit do you get? What step size do you need to use to get results that you can trust? What happens if you use a much larger step size? Produce at least one other qualitatively different non-circular trajectory for the Earth. What difficulties would humans have surviving on the Earth if it had such a non-circular orbit?

Force and Momentum:
(f) Choose initial conditions that give a non-circular orbit. Continuously display a vector showing the momentum of the Earth (with its tail at the Earth’s position), and a different colored vector showing the force on the Earth by the Sun, (with its tail also at the Earth’s position). You must scale these vectors appropriately to fit into the scene. A method to
figure out what scale factor to use is to print out numerical values of the momentum and the force and compare the magnitudes with the scale (distance) of the plots. For example, if the width of the scene is $W$ meters, and the force vector has a magnitude of $F$ (in Newton's), you might scale the force by a factor of $(0.1 \times W/F)$, which would make the length of the vector about 10% of the size of the picture. Is the force in the same direction as the momentum? How does the momentum depend on the distance from the Sun?

(SHOW THIS PROGRAM TO AN INSTRUCTOR FOR THE SIGNATURE.)

Additional Problems:

Change the form of the gravitational force from $Gm_1m_2/r^2$ to $Gm_1m_2/r^{1.05}$. Start with the same initial conditions that you had for the earth in a circular orbit around the Sun. What sort of orbit do you observe? Go back to $1/r^2$ and now increase the speed of the Earth by a small factor (1.05), what sort of orbit do you observe, (let the Earth go through several orbits around the Sun). Decrease the initial speed by a factor of 2. What sort of orbits do you observe (again, let the Earth make several orbits for each test).

Useful VPython Constructs:

The following three VPython constructs will be quite useful in this exercise.

```
#
earth.pos = vector(A,B,0)
#
# rhat is a unit vector in the direction of the vector.
#
rhat = norm(earth.pos)
#
# rsqrd is the square of the magnitude of the vector.
# rleng is the magnitude of the vector.
#
rsqrd = mag2(earth.pos)
rleng = mag(earth.pos)
#
```
Program Shell:

The following is a shell to help those with limited programming experience get started with the code. You are not required to use this shell, it is only provided as a template to help people get going on the code. Much of the preparation work above will allow one to initialize things to the correct values (the zeroes in the code). You will also need to code up the physics inside the main loop to cause the earth to move.

```python
from visual import *

# General setup functions for display. The first opens a display
# of a predefined size, while the second prevents autoscaling.
#
scene = display(title="Orbits",width=500,height=500,range=3.e11)
scene.autoscale=0 # supress scene jitter
#
# Define the sun and the earth:
#
sun    = sphere(color=color.yellow)
earth  = sphere(color=color.blue)
#
# Set up constants and initial conditions that will be needed by
# the program. Work these out in advance and enter them here.
#
G      = 0.0 # Gravitational Constant
sun.pos = vector(0,0,0) # Initial Sun Position
earth.pos = vector(0,0,0) # Initial Earth Position
sun.mass = 0.0 # Mass of the sun (kg)
earth.mass = 0.0 # Mass of the earth (kg)

dt      = 1.e6 # time increment in seconds (choose a sensible value?)
total   = 0 # total elapsed time

earth.velocity = vector(0,0,0) # Initial speed of the Earth (m/s)

# Some scale factors to control how big the earth and sun are drawn
# on the plot.

sun.scale = 1e1
earth.scale = 5e2
```
sun.radius = 7.e8*sun.scale
earth.radius = 6.4e6*earth.scale
#
# Initialize the momentum of the earth and sun.
#
earth.momentum = earth.mass*earth.velocity  # momentum of the earth
earth.trail = curve(color=earth.color)
earth.trail.append(pos=earth.pos)
#
# Define an arrow that points from the origin to the earth
#
rearrow = arrow(pos=(0,0,0), axis=earth.pos, color=earth.color, shaftwidth=1e6)
#
# We will loop for one year, and then stop.
#
# Useful commands:
# If vec is a vector, then mag(vec) is the magnitude of the vector
# and norm(vec) is a unit vector along the vector.
#
# Top of loop
while (total/3600/24 < 365):
    rate(100)  # limit the loop to a maximum of 100 times per second.
    earth.force =  # compute the force that the sun exerts on the earth.
    earth.momentum =  # update the earth’s momentum
    earth.pos =  # update the earth’s position
    
    earth.trail.append(pos=earth.pos)  # draw the trail
    
    total = total + dt # increment the time
#
# print this stuff when the loop is done!
#
print earth.pos
print 'All done.'