COS 126

N-body Simulation

Preparation

- Copy the following files from /u/cs126/files/Nbody/ into a directory:
  `Nbody.java, Nbody.html, NbodyDeluxe.java, NbodyDeluxe.html, earth.gif, sun.gif, venus.gif, mars.gif, mercury.gif, 2001.au, starfield.jpg`
- You can copy these files into your current working directory with the following command:
  `cp /u/cs126/files/Nbody/*`
- Use the code in MovingBall.java as a general template for the code you need to write in Body.java
- Read the code in Nbody.java to help you understand the what each method you will write needs to do.

Overview

- Create instance variables for the x & y coordinates, the x & y components of the velocity, the x & y components of the force, the mass, the image, and the applet associated with the body.
- Create a class variable for the gravitational constant G: 6.67e-11
- Implement the following methods:
  1. constructor
  2. void show(Graphics g)
  3. void move(int dt)
  4. void resetForce()
  5. void computeForce(Body b)

constructor

- Initializes all instance variables

show method

- rescale the actual coordinates of the body to fit the applet coordinates
- look at the init() function in Nbody.java to get an idea of the actual coordinates that each body may have (note: 2.279e11 = 2.279×10^{11})
- the sun should sit roughly in the center of the screen & you should be able to see the other 4 planets
- debug the code for the constructor and this function before you add any code for the other functions
- compile your code with the following command:
  `javac Body.java Nbody.java`
- run your applet with the following command:
  `appletviewer Nbody.html`

move method

- Use the following physics equations to write this code:

```
F[] = ma[]
new p[] = (current p[]) + v[]dt + a[]*dt*dt/2
new v[] = (current v[]) + a[]dt
```

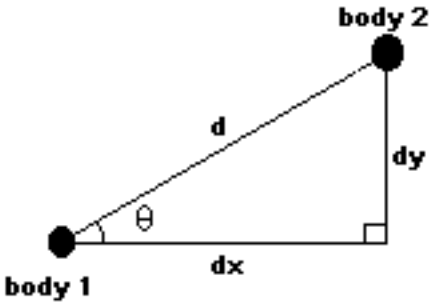
  - where F is force, m is mass, a is acceleration, p is position, and dt is the time interval passed into the function as a parameter
  - The [] brackets indicate that you should perform the calculation for both the x & y components of the force, acceleration, velocity, and position.
  - For example, calculate: fx = mass*ax AND fy = mass*ay
- This method should update the position and velocity of the body, given the current position, velocity, and force.
- When you write this code, the planets in the applet will all 'fall' straight down the screen (since there is no force causing them to orbit.)
- If you don't see this behavior, make sure you have the calculations in the right order. (The position and velocity need to be updated 'simultaneously' - recall the technique you used in the mandel() function for that assignment.)

resetForce method

- This one is pretty simple: you just need to reset the force acting on each particular body to zero.

computeForce method

- This method computes the force between this body and a second body (passed in as a parameter.)



- Use the following physics equations to write this code:

$$\text{Force} = \frac{(\text{mass of body 1} * \text{mass of body 2})}{(\text{distance between these bodies})^2} * G$$

```
Force in the x direction = Force * cos(Θ)
Force in the y direction = Force * sin(Θ)
(Recall cos(Θ) = dx/d and sin(Θ) = dy/d, as illustrated above)
(G is the gravitational constant)

So, new F[] = old F[] + new Force[]
```

- A useful method is double `Math.sqrt(double x)`
- Note that in Nbody.java, the method resetForce() will be called for all bodies, resetting their forces to zero. So, the 'old F[]' above is really just the sum of all the forces acting on a body computed already.
- If your planets are behaving strangely, you may want to check that your calculations for dx & dy.

NbodyDeluxe.java

- To try running your code with NbodyDeluxe.java, type the following commands:

```
javac Body.java NbodyDeluxe.java
appletviewer NbodyDeluxe.html
```

extra credit

- add a *slow-moving* comet that hits one of the other planets & write code that does something interesting when the collision occurs (merging planets, sound effects, etc.)
- or, make the simulation 3-dimensional by doing calculations for x, y, and z coordinates. use the z coordinate to vary the sizes of the planets
- or, design a planetary system with interesting behavior

what to submit

- submit Body.java (with comments, name, precept #) and a readme file.

```
/u/cs126/bin/submit 9 Body.java readme
```

- your readme should contain a high level description of the methods you wrote (don't explain the physics), description of problems encountered, and whatever help you received. Also include a description of any extra credit features you may have implemented.

Please contact Lisa Worthington <lworthin@cs.princeton.edu> if you find any typos/bugs in these instructions.