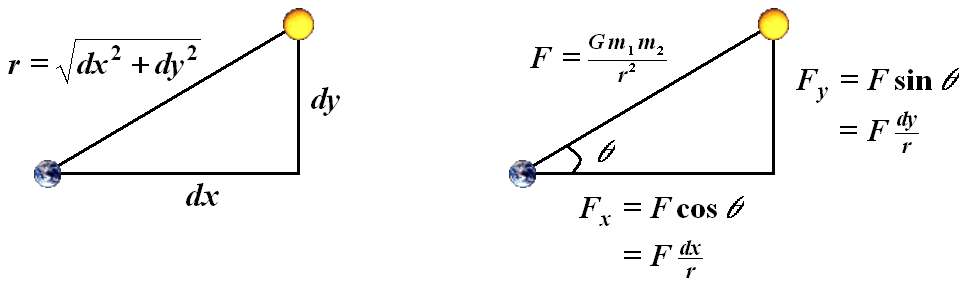
N-body

Given the starting positions, velocities, and masses of n celestial bodies (planets, stars, asteroids, etc.), write a program that simulates their motion based on gravitational forces.

# Physics

*Newton's law of gravitation* says that the strength of the gravitational force between two bodies is given by the product of their masses divided by the square of the distance between them, scaled by the gravitational constant G. Note that this is a *vector* sum: the pull of one body towards another acts on the line between them. Since Java uses Cartesian coordinates to display graphics, it is convenient to break up the gravitational force between two bodies into their x and y components (by projecting the original force onto the x- and y-axes).



### Total Forces

To calculate the total force acting on one body, you need to accumulate the gravitational force between this body and every other body in the system. The *principle of superposition* says that the net force acting on a body is the sum of the individual pairwise forces. Thus, by adding up the forces in the x and y directions, we obtain the total forces in the x and y directions for a body.

### Acceleration

Next, use *Newton's second law of motion* Ft[ ] = mat[ ] to calculate the current acceleration for each body. The square brackets indicate that this is a vector formula, e.g., at[ ] represents both the x and y components of the acceleration at time t. So use the formula twice, once to find the Force in the x direction and once for the y direction.

### Velocity

Use the acceleration to compute the current velocity. vt[ ] = v0[ ] + at[ ] dt. The time difference, dt, is a small (relative to G) constant number that will affect the smoothness of the simulation. Start by using 10,000 and adjust until the standard planets simulation runs correctly.

### Position

Finally, use the computed velocities to update the position at time t + dt. Assume the velocity is held constant in the interval from t to t + dt, and use the formula pt[ ] = p0[ ] + vt[ ] dt. Now that you know the position of each of the bodies you can update them on the screen.

# Drawing

You have been given a library, StdDraw.class that you may use to draw to the screen. BouncingBall.java is an example. You also have access to the api at <http://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html>

# File Input

The starting conditions for a simulation are defined in a text file with a specific structure, which will allow you to write code to read it in to start your program. Here is planets.txt, the first simulation you should get working:

5

2.50e11

1.496e11 0.000e00 0.000e00 2.980e04 5.974e24 earth.gif

2.279e11 0.000e00 0.000e00 2.410e04 6.419e23 mars.gif

5.790e10 0.000e00 0.000e00 4.790e04 3.302e23 mercury.gif

0.000e00 0.000e00 0.000e00 0.000e00 1.989e30 sun.gif

1.082e11 0.000e00 0.000e00 3.500e04 4.869e24 venus.gif

* The first line represents the number of bodies
* The second line is the radius of the universe
* Each of the next n lines represents a body, with its starting conditions (in order left to right):
  + x position
  + y position
  + x velocity
  + y velocity
  + mass
  + name of the image file representing the planet, located in the images folder