A Ball and a Trampoline – Daniel Sinkovits

Purpose
To be able to fully master the mathematical formulas used in physics, students need to develop an intuition for what they physically imply. This lesson makes those connections through the use of a simulation of a ball falling and bouncing on a trampoline. The students will use the Java source code and make small changes to produce different behavior.

Overview
The teacher will introduce the concept of a molecular dynamics simulation using a Java applet as a visual aid. The teacher will guide everyone to open the source code and identify the main features of the program. He or she will also assess student understanding of understanding of translating mathematical formulas to and from computer code by eliciting student feedback, and will be ready to explain. Then, the students will work independently on the computers using the worksheet with periodic help from the teacher. At five minutes before the end, the teacher will pass out the short quiz.

Student Outcomes
1. SWBAT translate a mathematical formula into computer code and vice versa.
2. SWBAT identify physical phenomena from the trajectory produced by the simulation. (free fall, terminal velocity, collisions, periodic motion, damping, equilibrium)
3. SWBAT identify physical forces from computer code. (gravity, air friction, trampoline)

Illinois State Science Standards identified
• 12.D.5a Analyze factors that influence the relative motion of an object (e.g., friction, wind shear, cross currents, potential differences).

College Board Learning Objectives
• II.A.b.3 Students should be able to analyze situations in which a particle is in static equilibrium under the influence of several forces.
• II.A.c.6 Students should understand the effect of fluid friction on the motion of a body so that they can:
  o Find the terminal velocity of a body moving vertically through a fluid that exerts a retarding force proportional to velocity, \( F_{\text{fluid}} = -kv \).
  o Describe qualitatively, with the aid of graphs, the acceleration, velocity, and displacement of such a particle when it is released from rest or projected vertically with specified initial velocity.

Time
90 minutes

Level
AP Physics
Materials and Tools

- One computer per student or pair of students
- A Java programming environment, such as JCreator, Netbeans, or Eclipse, installed on each
- The Java source code
- One worksheet per student (attached)
- One quiz per student (attached)

Preparation

- Test and create a list of steps to access and open the Java source code with your programming environment.
- Test and create a list of steps to import the simulation data into Excel and create a graph.
- Identify the common programming mistakes that students will make (missing semicolon, unmatched braces, etc.) and how the programming environment reports them. These can be distributed to the students or used by the teacher and any assistants to quickly fix the most common problems so students can continue working.

Prerequisites

Students need to have learned some aspects of Java programming before this lesson. They should have used the programming environment before to compile a simple program. They also must know how to use variables and mathematical operators to do mathematical operations. Finally, they must know the operation of the Java statements while and if. A short primer may suffice.

Background

Students need to have a good working knowledge of the formulas for constant acceleration motion. They also need familiarity Hooke’s law. Exposure to the concepts of air resistance and terminal velocity is also recommended. Finally, students should be able to do basic operations in Excel, including making a plot.

Teaching Notes

2. Demonstrate a simulation using this Java applet: http://physics.weber.edu/schroeder/software/mdapplet.html
   Explain the division of the simulation into timesteps of constant acceleration.
3. Then direct their attention to the handout. Explain the different sections of the code: the constants, the initial conditions, the simulation loop, and the force calculation.
4. Check for understanding of how to read and write equations in computer code.
5. Once this is clear, guide the students to copy the source code file to their home directory and open it in their Java editor.
6. Let them work on the worksheet while you assist the students who need help getting started. Announce that the worksheet will be collected. Also announce that there will be a short quiz about the physical interpretation of the cod at the end.
7. Demonstrate how to import the data file generated by the simulation into Excel and plot it.
8. Return to circulating the classroom and assisting students.
9. Check for understanding and last-minutes questions. Deliver the quiz.

Assessment

It is imperative that all students master objective 1 early in the lesson; therefore the teacher should elicit immediate feedback and address any needs. The attached worksheet will assess mastery of objective 2, and the attached short quiz will test objective 3.
A Ball and a Trampoline
Today we’ll simulate a more realistic simulation: a falling ball that bounces on a trampoline. We’ll start with just a falling ball.

Here is the source code that you can write notes on. You will also be provided with the source code on a file that you can load.

```java
import java.io.*;
public class Simulation {
    // These are constants used by both the main function and the force function
    static double g = 9.8;  // acceleration of gravity
    static double m = 6.0;  // mass of ball

    public static void main(String[] args) throws IOException {
        PrintStream p = new PrintStream(new FileOutputStream("myfile.txt"));
        double y = 0;          // initial position
        double v = 0;          // initial velocity
        double t = 0;          // initial time
        double endTime = 10;   // simulation time
        double dt = 0.01;      // timestep
        double a = force(y,v)/m;  // Calculates initial acceleration

        while(t <= endTime) {
            String output = String.format("%8.3f % 8.3f % 8.3f ", t, y, v);
            System.out.println(output);
            p.println(output);
            t = t + dt;          // Increments timestep
            v = v + 0.5*a*dt;    // Integrates velocity a half step
            y = y + v*dt;        // Integrates position a full step
            a = force(y,v)/m;    // Calculates new acceleration
            v = v + 0.5*a*dt;    // Integrates velocity the rest of the way
        }
    }

    static double force(double y, double v){     // this is a function to output the force
        double force = -m*g;   // this includes only the acceleration due to gravity
        return force;
    }
}
```
1. Run the simulation.
   a. Import myfile.txt into Excel.
   b. Make plots of y vs t and v vs t.
   c. What is the shape of each of those plots and what do they mean physically?

2. Add air friction to the simulation.
   a. Add a term to the force of the form \( F_f = -fv \), where \( f \) is a constant.
   b. Simulate with \( f = 1.0 \).
   c. Plot the results of both y vs t and v vs t.
   d. Based on the plots, in what ways does air friction change the results? (What happens after a long time?)
   e. Turn off air friction for the next part (set \( f = 0 \)).

3. Add a trampoline to the system.
   a. Add a term to the force of the form \( F_{\text{trampoline}} = -ky \) that only activates when \( y < 0 \). (Use an if statement for this.)
   b. Set the initial y to 10m. Set the constant \( k = 1000 \).
   c. Simulate and plot the results. Increase endTime, if necessary.
   d. Now, increase the mass to 12kg. Simulate again and plot on the same plot.
   e. How are the simulations with different mass similar and how are they different?

4. Now add friction to the trampoline.
   a. Devise a term to add to the force that would provide friction only when the ball is in contact with the trampoline.
   b. Hint: you can look at the example of air friction for inspiration.
   c. Plot the result over a long time interval.
   d. What happens to the ball with each successive bounce?
   e. What eventually happens?

f. A system that doesn't change is said to have reached equilibrium or steady-state.

g. What is the equilibrium y value for this system (after a very long time)?

h. Use the equation for force to calculate the equilibrium y value. (Hint: Draw the force diagram for the ball at equilibrium.)
1. In the following short snippet of Java code, circle and identify the different forces that are present:

   ```java
   static double force(double y, double v){
       double force = -m*g - f*v;
       if(y<0){
           force = force - k*y;
       }
       return force;
   }
   ```

2. The following code contains an additional force. Explain what it does.

   ```java
   static double force(double y, double v){
       double force = -m*g - f*v;
       if(y<0){
           force = force - k*y;
       }
       double h = 5.0;
       if(y > h){
           force = force - k*(y-h);
       }
       return force;
   }
   ```