Using the Open Source Physics Java Library
to do Computer Simulations

Harvey Gould
Clark University

Jan Tobochnik
Kalamazoo College

Wolfgang Christian
Davidson College

Supported in part by NSF grants DUE-0127363 & DUE-0126439.
14 March 2006
Themes

- Java is an excellent choice of programming language for writing software for educational applications.
- The use of the Open Source Physics Library makes writing programs with a GUI much easier.
- Java is very useful for writing simulations of research interest in which visualization is important.
- Associated curricular materials and a user’s guide are available.

Web sites: <www.opensourcephysics.org/> <sip.clarku.edu/> <stp.clarku.edu>
Why choose Java?

- Java is object-oriented – functions and data are grouped together in an object. We do students a disfavor if we first teach them Fortran or another procedural language.
- Platform independence, flexible graphics libraries, good performance, and zero cost.
- Java’s popularity ensures that the language will continue to evolve and expertise in Java is a marketable skill.
- Many third-party libraries, including those for numerical calculations and visualization.
- Availability of Open Source Physics library, including software for analyzing videos, 3D graphics, XML framework, and a software authoring tool (Ejs).
Features of the Open Source Physics Library

• Plots and animations can be implemented without knowing about threads and the more sophisticated aspects of Java.
• A Java application can be embedded into a html page without any change in the code.
• Many people and institutions are involved, including Wolfgang Christian, Mario Belloni, Anne Cox, Francisco Esquembre, Doug Brown, Jan Tobochnik, Kip Barros, and Joshua Gould.
• Programmers wishing to adopt the OSP library may do so provided that they release their source code under the GNU Open-Source GPL license.
Examples of programs for education

Figure 1: Simulation of a pendulum.
Figure 2: Example of the use of the self-contained OSP Spins Launcher package program by Mario Belloni and Wolfgang Christian.

The OSP Spins program is based on the Spins program by David McIntyre, which was based on a program by Dan Schroeder. Because their code is open source, Belloni and Christian were able to use it and improve it – an argument for open source. See Schroeder and Moore, “A computer-simulated Stern-Gerlach laboratory,” Am. J. Phys. 61, 798–805 (1993).
package org.opensourcephysics.sip.ch04;
import java.awt.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.numerics.*;

public class Pendulum implements Drawable, ODE {
    double omega0Squared = 3; // g/L
    double[] state = new double[] {0, 0, 0}; // {theta, dtheta/dt, t}
    Color color = Color.RED;
    int pixRadius = 6;
    EulerRichardson odeSolver = new EulerRichardson(this);

    public void setStepSize(double dt) {
        odeSolver.setStepSize(dt);
    }

    public void step() {
        odeSolver.step(); // execute one time step
    }

    public void setState(double theta, double thetaDot) {
        state[0] = theta;
        state[1] = thetaDot; // time rate of change of theta
    }

    public double[] getState() {
        return state;
    }

    public void getRate(double[] state, double[] rate) {
        rate[0] = state[1]; // rate of change of angle
        rate[1] = -omega0Squared*Math.sin(state[0]); // rate of change of dtheta/dt
        rate[2] = 1; // rate of change of time dt/dt = 1
    }

    public void draw(DrawingPanel drawingPanel, Graphics g) {
        int xpivot = drawingPanel.xToPix(0);
        int ypivot = drawingPanel.yToPix(0);
        int xpix = drawingPanel.xToPix(Math.sin(state[0]));
        int ypix = drawingPanel.yToPix(-Math.cos(state[0]));
        g.setColor(Color.black);
        g.drawLine(xpivot, ypivot, xpix, ypix); // the string
        g.setLine(xpivot, ypivot, xpix, ypix);
        g.setColor(color);
        g.fillOval(xpix-pixRadius, ypix-pixRadius, 2*pixRadius, 2*pixRadius); // bob
    }
}
Model-View-Controller

```java
package org.opensourcephysics.sip.ch04;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.*;

public class PendulumApp extends AbstractSimulation {
    PlotFrame plotFrame = new PlotFrame("Time", "Theta", "Theta versus time");
    Pendulum pendulum = new Pendulum();
    DisplayFrame displayFrame = new DisplayFrame("Pendulum");

    public PendulumApp() {
        displayFrame.addDrawable(pendulum);
        displayFrame.setPreferredMinMax(-1.2, 1.2, -1.2, 1.2);
    }

    public void initialize() {
        double dt = control.getDouble("dt");
        double theta = control.getDouble("initial theta");
        double thetaDot = control.getDouble("initial dtheta/dt");
        pendulum.setState(theta, thetaDot);
        pendulum.setStepSize(dt);
    }

    public void doStep() {
        plotFrame.append(0, pendulum.state[2], pendulum.state[0]); // angle vs time data added
        pendulum.step(); // advances state by one time step
    }

    public void reset() {
        pendulum.state[2] = 0; // set time = 0
        control.setValue("initial theta", 0.2);
        control.setValue("initial dtheta/dt", 0);
        control.setValue("dt", 0.1);
    }

    public static void main(String[] args) {
        SimulationControl.createApp(new PendulumApp());
    }
}
```
Figure 3: Modified Wang-Landau algorithm for hard disks (Hui Wang).
Resources


Information on other textbooks at <sip.clarku.edu/books/>.