```
/* $Id: Tutorial_02_08.java,v 1.2 2006/05/18 21:36:08 mdanzig Exp $ */
/**
 * @author John Belcher - Department of Physics / MIT
 * @version $Revision: 1.2 $
 */

package tealsim.physics.tutorials;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.beans.PropertyChangeEvent;
import javax.media.j3d.*;
import javax.vecmath.*;

import teal.field.Field;
import teal.framework.TealAction;
import teal.math.RectangularPlane;
import teal.plot.PlotProperties;
import teal.plot.Graph;
import teal.render.Rendered;
import teal.sim.collision.SphereCollisionController;
import teal.sim.control.VisualizationControl;
import teal.sim.physical.EMEngine;
import teal.sim.physical.PhysicalObject;
import teal.sim.physical.Wall;
import teal.sim.physical.em.MagneticDipole;
import teal.sim.physical.em.RingOfCurrent;
import teal.render.j3d.*;
import teal.render.j3d.loaders.Loader3DS;
import teal.sim.simulation.SimWorld;
import teal.sim.spatial.FieldConvolution;
import teal.sim.spatial.FieldDirectionGrid;
import teal.sim.spatial.FieldLine;
import teal.sim.spatial.FieldLineManager;
import teal.sim.spatial.FluxFieldLine;
import teal.ui.control.*;
import teal.util.TDebug;
import teal.visualization.dlic.DLIC;

public class Tutorial_02_08 extends SimWorld {

    private static final long serialVersionUID
    = 3257008735204554035L;

    Rendered nativeObject = new Rendered();
    ShapeNode ShapeNodeNative = new ShapeNode();
    PropertyDouble currentSlider = new PropertyDouble();
    RingOfCurrent floatingCoil;
    Vector3d floatingCoilPos;
    MagneticDipole magDipole;
      Graph position_graph;
      PlotProperties position_plot;
      private FieldLineManager fmanager;
    private FieldLine fl = null;
    private FieldConvolution mDLIC;
    private FieldDirectionGrid fv;
    double ringRad = 0.43;
```

```
58      double torR = 0.08;
59      double ringMass = 3.5;
60      double current = -50.;
61      double magLen = 0.24;
62      double magR = 0.09;
63      double searchRad = magR;
64      double fLen = 0.033;
65      double minD = 0.03;
66      int kMax = 300; //300
67      int fMode = FieldLine.RUNGE_KUTTA;
68      double friction = 0.;
69
70      public Tutorial_02_08() {
71          super();
72
73          TDebug.setGlobalLevel(0);
74
75          title = "Tutorial_02_08";
76
77              ///// Set properties on the SimEngine /////
78              // Bounding area is the size of simulation space.
79              // setDeltaTime() sets the time step of simulation.
80              // setDamping() sets the damping on system.
81          EMEngine emEngine = new EMEngine();
82          setModel(emEngine);
83          BoundingSphere bs = new BoundingSphere(
84                    new Point3d(0, 1.6, 0), 03.5);
85          theEngine.setBoundingArea(bs);
86          theEngine.setDeltaTime(0.02);
87          ((EMEngine)theEngine).setGravity(
88                    new Vector3d(0., -9.81, 0.));
89          ((EMEngine)theEngine).setDamping(friction);
90          mViewer.setBoundingArea(bs);
91
92          // Create electromagnetic objects
93          magDipole = new MagneticDipole();
94          magDipole.setMu(10.);
95          magDipole.setPosition(new Vector3d(0., 0., 0.));
96          magDipole.setDirection(new Vector3d(0, 1, 0));
97          magDipole.setPickable(false);
98          magDipole.setRotable(false);
99          magDipole.setMoveable(false);
100         magDipole.setRadius(magR);
101         magDipole.setLength(magLen);
102         addElement(magDipole);
103
104         floatingCoil = new RingOfCurrent();
105         floatingCoil.setID("Ring");
106         floatingCoil.setDirection(new Vector3d(0.,1.,0.));
107         floatingCoil.setPickable(true);
108         floatingCoil.setRotable(true);
109         floatingCoil.setMoveable(true);
110         floatingCoil.setInducing(false);
111         floatingCoil.setRadius(ringRad);
112         floatingCoil.setTorusRadius(torR);
113         floatingCoil.setMass(ringMass);
114         floatingCoil.setInducing(false);
```

```
115         floatingCoil.setInductance(0.1);
116         floatingCoil.setCurrent(current);
117 
118             // Add a collisionController to the RingOfCurrent
119         // so that it will be registered as a colliding object
120             // and react appropriately when it touches the Wall.
121         SphereCollisionController sccx =
122             new SphereCollisionController(floatingCoil);
123         sccx.setRadius(torR);
124         sccx.setTolerance(0.01);
125         floatingCoil.setColliding(true);
126         floatingCoil.setCollisionController(sccx);
127         floatingCoilPos = new Vector3d(0., 1.25+ ringRad +
128                   (ringRad * 0.02), 0.);
129         addElement(floatingCoil);
130 
131         // Create a wall that the floating coil collides with
132             // Wall constructor.
133         Wall wall = new Wall(new Vector3d(0., 0, 0.),
134                   new Vector3d(2., 0., 0.), new Vector3d(0., 0., 2.));
135         wall.setElasticity(1.);
136         addElement(wall);
137 
138         // create the sliders for the amount of current
139 
140         currentSlider.setText("I");
141         currentSlider.setMinimum(-100);
142         currentSlider.setMaximum(100);
143         currentSlider.setPaintTicks(true);
144         currentSlider.addPropertyChangeListener("value", this);
145         currentSlider.setValue(current);
146         currentSlider.setVisible(true);
147 
148         // add the slider to a control group
149 
150         ControlGroup controls = new ControlGroup();
151         controls.setText("Parameters");
152         controls.add(currentSlider);
153         addElement(controls);
154 
155             // Create a graph of the height of the coil
156             // Graph constructor.
157             position_graph = new Graph();
158             position_graph.setSize(150, 200);
159             position_graph.setXRange(0., 6.);
160             position_graph.setYRange(-0., 2.);
161             position_graph.setWrap(true);
162             position_graph.setClearOnWrap(true);
163             position_graph.setXLabel("Time");
164             position_graph.setYLabel("position");
165             // Here we create the PlotItem being drawn
166             position_plot = new PlotProperties();
167             position_plot.setObjectX(theEngine);
168             position_plot.setPropertyX("time");
169             position_plot.setObjectY(floatingCoil);
170             position_plot.setPropertyY("y");
171             // adds the supplied PlotItem to the graph.
```

```
            position_graph.addPlotItem(position_plot);

            // Here we create a new Control Group
            ControlGroup graphPanel = new ControlGroup();
            graphPanel.setText("Graphs");
            graphPanel.addElement(position_graph);
            addElement(graphPanel);

            // add field lines

          fmanager = new FieldLineManager();
        fl = makeFLine(-200.0, floatingCoil, null,
                  fLen, kMax, fMode);
        fmanager.addFieldLine(fl);
        fl = makeFLine(-1000.0, floatingCoil, null,
                  fLen, kMax, fMode);
        fmanager.addFieldLine(fl);
        fl = makeFLine(120.0, magDipole, null,
                  fLen, kMax, fMode);
        fl.setBuildDir(FieldLine.BUILD_NEGATIVE);
        ((FluxFieldLine) fl).setBrakSteps(600);
        ((FluxFieldLine) fl).setBrentSteps(600);
        fmanager.addFieldLine(fl);
        fl = makeFLine(220.0, magDipole, null,
                  fLen, kMax, fMode);
        fl.setBuildDir(FieldLine.BUILD_NEGATIVE);
        fmanager.addFieldLine(fl);
        fl = makeFLine(400.0, magDipole, null,
                  fLen, kMax, fMode);
        fl.setBuildDir(FieldLine.BUILD_NEGATIVE);
          fmanager.setElementManager(this);

            // Here we add a FieldConvolution generator
        RectangularPlane rec = new RectangularPlane(
                  new Vector3d(-3., -3., 0.),
                        new Vector3d(-3., 3., 0.),
                        new Vector3d(3., 3., 0.));
            mDLIC = new FieldConvolution();
            mDLIC.setSize(new Dimension(512, 512));
            mDLIC.setVisible(false);
            mDLIC.setComputePlane(rec);

        // Here we create a FieldDirectionGrid
            // FieldDirectionGrid constructor.
        fv = new FieldDirectionGrid();
        // setType() sets the type of field
        fv.setType(Field.B_FIELD);

        //now add all these to a VisualizationControl
        VisualizationControl vis = new VisualizationControl();
        vis.setFieldLineManager(fmanager);
        vis.setFieldVisGrid(fv);
        vis.setFieldConvolution(mDLIC);
            vis.setConvolutionModes(
                        DLIC.DLIC_FLAG_B | DLIC.DLIC_FLAG_BP);
        addElement(vis);

```

```
        // import a .3DS files object using Loader3DS
        double scale3DS = 0.02;
        Loader3DS max = new Loader3DS();

        BranchGroup bg01 =
         max.getBranchGroup("resources/models/rear.3DS",
         "resources/models/");
        Node3D node01 = new Node3D();
        node01.setScale(scale3DS);
        node01.addContents(bg01);
        Rendered importedObject01 = new Rendered();
        importedObject01.setNode3D(node01);
        importedObject01.setPosition(new Vector3d(0., 0., 0.));
       addElement(importedObject01);

        // set paramters for mouseScale
        Vector3d mouseScale = mViewer.getVpTranslateScale();
        mouseScale.x *= 0.05;
        mouseScale.y *= 0.05;
        mouseScale.z *= 0.5;
        mViewer.setVpTranslateScale(mouseScale);

        mSEC.init();
        resetCamera();
        // addAction for pulldown menus on TEALsim windows
        addActions();
        reset();

    }

    void addActions() {
        TealAction ta = new TealAction("Tutorial_02_08", this);
        addAction("Help", ta);
    }

    public void actionPerformed(ActionEvent e) {
        TDebug.println(1, " Action comamnd: "
                  + e.getActionCommand());
        if (e.getActionCommand().compareToIgnoreCase(
                  "Tutorial_02_08") == 0) {
            mFramework.openBrowser(
                        "resources/help/tutorial_02_08.html");
        }  else {
            super.actionPerformed(e);
        }
    }

    public void reset() {
        floatingCoil.setPosition(floatingCoilPos);
        floatingCoil.setVelocity(new Vector3d());
        floatingCoil.setDirection(new Vector3d(0., 1., 0.));
        position_graph.clear(0);
        ((EMEngine)theEngine).setDamping(friction);
        currentSlider.setValue(current);
            theEngine.requestRefresh();
    }

```

```
286     public void resetCamera() {
287         mViewer.setLookAt(new Point3d(0.0, 0.025, 0.4),
288                   new Point3d(0., 0.025, 0.),
289                   new Vector3d(0., 1., 0.));
290     }
291 
292     public void propertyChange(PropertyChangeEvent pce) {
293         Object source = pce.getSource();
294         if (source == currentSlider) {
295             current = ((Double) pce.getNewValue()).doubleValue();
296             floatingCoil.setCurrent(current);
297         } else {
298             super.propertyChange(pce);
299         }
300     }
301 
302     protected FieldLine makeFLine(double val, PhysicalObject obj,
303       Color color, double fLen, int kMax, int fMode) {
304         Color col = color;
305         Vector3d start = new Vector3d(0, 0, 0);
306         Vector3d positive = new Vector3d(1, 0, 0);
307         FluxFieldLine fl;
308         if (obj == null) {
309             fl = new FluxFieldLine(val, start, positive, searchRad);
310         } else {
311             if (obj instanceof RingOfCurrent) {
312                 fl = new FluxFieldLine(val, obj, true, true);
313             } else if (obj instanceof MagneticDipole) {
314                 fl = new FluxFieldLine(val, obj, true, false);
315                 fl.setObjRadius(searchRad);
316             } else {
317                 return null;
318             }
319         }
320         fl.setMinDistance(minD * 0.5);
321         fl.setIntegrationMode(fMode);
322         fl.setKMax(kMax);
323         fl.setSArc(fLen);
324         //fl.setSymmetryCount(symCount);
325         fl.setColorMode(FieldLine.COLOR_VERTEX);
326         fl.setReceivingFog(true);
327         if (col != null) {
328             fl.setColor(col);
329         }
330         return fl;
331     }
332 }
```