the body is computed in `RigidBody`. The angular momentum and angular velocity vectors are retrieved from `RigidBody` and used to set the direction of the corresponding arrows. The path of the angular velocity arrow through space is recorded by adding points to an `ElementTrail` object.

Exercise 17.17 Rigid body model

Study the `FreeRotationSpaceView` and how it is used by `FreeRotationApp`. Adapt the `FeynmanPlateView` so that it too uses the `RigidBody` class to compute the dynamics. Do you obtain the same results as in Problem 17.13? ■

Listing 17.13 The `FreeRotationSpaceView` class displays the rotation of a rigid body as seen in the space frame.

```
package org.opensourcephysics.sip.ch17;
import org.opensourcephysics.display3d.simple3d.*;
import org.opensourcephysics.frames.*;

public class FreeRotationSpaceView {
   Element ellipsoid = new ElementEllipsoid();
   Element omega = new ElementArrow();
   Element angularMomentum = new ElementArrow();
   ElementTrail omegaTrace = new ElementTrail();
   Display3DFrame frame = new Display3DFrame("Space view");
   FreeRotation rigidBody;
   double scale = 1;

   public FreeRotationSpaceView(FreeRotation rigidBody) {
      this.rigidBody = rigidBody;
      frame.setSize(600, 600);
      frame.setDecorationType(VisualizationHints.DECORATION_AXES);
      omega.getStyle().setFillColor(java.awt.Color.RED);
      omegaTrace.getStyle().setLineColor(java.awt.Color.RED);
      angularMomentum.getStyle().setFillColor(java.awt.Color.GREEN);
      ellipsoid.setTransformation(rigidBody.getTransformation());
      frame.addElement(ellipsoid);
      frame.addElement(omega);
      frame.addElement(angularMomentum);
      frame.addElement(omegaTrace);
   }

   void initialize(double a, double b, double c) {
      ellipsoid.setSizeXYZ(2*a, 2*b, 2*c);
      // bounding dimension for space frame
      scale = Math.max(Math.max(3*a, 3*b), 3*c);
      frame.setPreferredMinMax(-scale, scale, -scale, scale,
                               -scale, scale);
      omegaTrace.clear();
      update();
   }

   void update() {
      ellipsoid.setTransformation(rigidBody.getTransformation());
      double[] vec =
           ellipsoid.toSpaceFrame(rigidBody.getBodyFrameOmega());
      double norm =
           Math.sqrt(vec[0]*vec[0]+vec[1]*vec[1]+vec[2]*vec[2]);
      norm = Math.max(norm, 1.0e-6);
      double s = 0.75*scale/norm;
      omega.setSizeXYZ(s*vec[0], s*vec[1], s*vec[2]);
      omegaTrace.addPoint(s*vec[0], s*vec[1], s*vec[2]);
      vec = ellipsoid.toSpaceFrame(
           rigidBody.getBodyFrameAngularMomentum());
      norm = Math.sqrt(vec[0]*vec[0]+vec[1]*vec[1]+vec[2]*vec[2]);
      norm = Math.max(norm, 1.0e-6);
      s = 0.75*scale/norm;
      angularMomentum.setSizeXYZ(s*vec[0], s*vec[1], s*vec[2]);
      frame.repaint();
   }
}
```


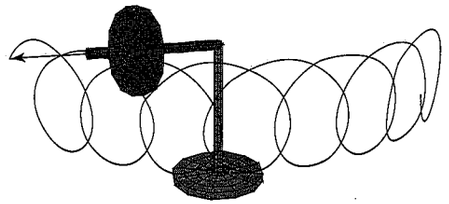

Figure 17.6 The dynamics of a spinning top are computed using a `RigidBodyModel` and displayed using a `SpinningTopSpaceView`.

17.8 ■ MOTION OF A SPINNING TOP

Object-oriented programming makes it easy to add a torque to the `RigidBody` class. All that needs to be done is to extend `RigidBody` and override the `computeBodyFrameTorque` method. The spinning top shown in Figure 17.6 is a rigid body that rotates about a pivot. Because the pivot point is not the center of mass, there is an external torque due to the weight of the top. We must compute the torque's vector components in the body frame to use Euler's equation of motion.

The `SpinningTop` shown in Listing 17.14 models a symmetric body rotating about the axis of symmetry. The axis of symmetry is taken to be the $\hat{3}$-axis. If we assume that the center of mass lies a unit distance from the pivot along the $\hat{3}$-axis, then the torque is computed by taking the cross product after projecting the force into the body frame $\tau = \hat{3} \times F_{\text{body}}$. Listing 17.14 assumes that the center of mass is one unit from the pivot and is acted on by an external force in the z direction (0, 0, 1).

Listing 17.14 The `SpinningTop` class models a symmetric body rotating about the axis of symmetry.

```
package org.opensourcephysics.sip.ch17;
public class SpinningTop extends RigidBody {
   SpinningTopSpaceView spaceView = new SpinningTopSpaceView(this);

   void setInertia(double Is, double Iz) {
      I1 = Is;
      I2 = Is;
```