```
public class Interaction3DApp implements InteractionListener {

   public Interaction3DApp() {
      Display3DFrame frame = new Display3DFrame("3D interactions");
      frame.setPreferredMinMax(-2.5, 2.5, -2.5, 2.5, -2.5, 2.5);
      // accepts interactions from the frame's 3D drawing panel
      frame.addInteractionListener(this);
      Element particle = new ElementCircle();
      particle.setSizeXYZ(1, 1, 1);
      // enables interactions that change positions
      particle.getInteractionTarget(Element.TARGET_POSITION).
           setEnabled(true);
      // accepts interactions from the particle
      particle.addInteractionListener(this);
      frame.addElement(particle);            // adds particle to panel
      ElementArrow arrow = new ElementArrow();
      // enables interactions that change the size
      arrow.getInteractionTarget(Element.TARGET_SIZE).setEnabled(true);
      // accepts interactions from the arrow
      arrow.addInteractionListener(this);
      // adds the arrow to the panel
      frame.addElement(arrow);
      // enables interactions with the 3D Frame
      frame.enableInteraction(true);
      // accepts interactions from the frame
      frame.addInteractionListener(this);
      frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }

   public void interactionPerformed(InteractionEvent _evt) {
      Object source = _evt.getSource();
      if(_evt.getID()==InteractionEvent.MOUSE_PRESSED) {
         System.out.println("Mouse clicked");
      }
      if(source instanceof ElementCircle) {
         System.out.println("A particle has been hit");
      }
   }

   static public void main(String args[]) {
      new Interaction3DApp();
   }
}
```

Elements can be grouped together and manipulated as a single object by creating geometric shapes such as spheres, boxes, and arrows and adding them to a `Group`. An easy way to create an `Element` consisting of many geometric shapes is to subclass `Group` and instantiate the other `Element` objects in the constructor and add them to the `Group`. Listing 17.7 shows an example of such a composite. The entire group acts like a single `Element` that can be translated and rotated. The (x, y, z) parameters passed to the `setXYZ` method of an object placed within a group are relative to the group's position. The (x, y, z) parameters passed to an element's `setSizeXYZ` method are along the group's axes even if the group has been rotated.

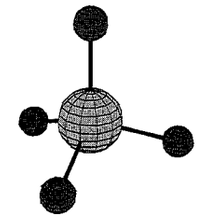

Figure 17.3 A visualization of the methane molecule using the Open Source Physics simple3d package.

Listing 17.7 The `Barbell3D` class creates a compound object by instantiating simpler shapes and adding them to a `Group`.

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

public class Barbell3D extends Group {
   public Barbell3D() {
      ElementCylinder bar = new ElementCylinder();
      bar.setXYZ(0, 0, 5);
      bar.setSizeXYZ(0.2, 0.2, 10);
      addElement(bar);
      Element sphere = new ElementSphere();
      sphere.setXYZ(0, 0, -5);
      sphere.setSizeXYZ(4, 4, 4);
      addElement(sphere);
      sphere = new ElementSphere();
      sphere.setXYZ(0, 0, 5);
      sphere.setSizeXYZ(4, 4, 4);
      addElement(sphere);
   }
}
```

Exercise 17.9 Group test
Write a test program that instantiates and displays a `Barbell`. Describe the change in rendering while dragging within the view. ■

The code package for this chapter includes an Open Source Physics 3D version of the methane molecule (see Figure 17.3) but is not listed here because of its length. As in the previous example, a `Group` is used to define a `Methane` class. This model is instantiated and added to a `Display3DFrame` in the `Methane3DApp` class.

17.4 ■ DYNAMICS OF A RIGID BODY

The dynamical behavior of a rigid body is determined by

$$\frac{d\mathbf{P}}{dt} = \mathbf{F} \tag{17.18a}$$

$$\frac{d\mathbf{L}}{dt} = \mathbf{N}, \tag{17.18b}$$