```
            mouseX = panel.getMouseX();
            mouseY = panel.getMouseY();
            break;
         }
      }

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

Exercise 17.7 Methane rotation
Modify the methane program by replacing the `direct` transformation with the `inverse` transformation. Run and compare this program to the original program. How did this change affect the mouse actions? Why? ■

Exercise 17.8 Hidden surface removal
`MethaneApp` draws atoms using the same color which hides the fact that the drawing is not correct. Modify the `Methane` class so that the carbon atom is drawn as a green circle and run the `MethaneApp` program again. Notice that the carbon atom is hidden by the hydrogen atoms that are behind the carbon atom. Rewrite the `Methane` class so that the drawing order is determined by the atom's z-coordinate. Ordering along the line of sight is often used in graphics programs for hidden line and hidden surface removal. ■

17.3 ■ THE THREE-DIMENSIONAL OPEN SOURCE PHYSICS LIBRARY

The `display3d` package contains a three-dimensional drawing framework that makes it easy to create simple visualizations. In the spirit of object-oriented programming, we will not study the Open Source Physics 3D implementation in detail but will describe only its API. The `Box3DApp` class creates a simple three-dimensional visualization. Run the program and drag the mouse within the panel. We need not concern ourselves with details such as hidden line removal, perspective, or rotation.

The Open Source Physics 3D API is defined in the `org.opensourcephysics.display3d` package. This API can be be implemented using any 3D library, and we have done so in the `simple3d` package using only standard Java. We use the `simple3d` package in this book because programs written using this package will run on any Java enabled computer. If truly high resolution rendering is required, it is best to import an OSP 3D implementation that uses an add-on library such as Java for Open GL (JOGL) or Java 3D. These libraries must be installed, but they support hardware accelerated rendering using Open GL graphics language drivers. A JOGL implementation of Open Source Physics 3D is being developed. All that is required to use this implementation is to change the import statement to use the `jogl` package.

Listing 17.5 demonstrates that it is just as easy to create a 3D visualization as it is to create a 2D visualization.

Listing 17.5 `Box3DApp` creates a box within a `Display3DFrame`.

```
package org.opensourcephysics.sip.ch17;
import java.awt.*;
```

```
import org.opensourcephysics.frames.*;
import org.opensourcephysics.display3d.simple3d.*;

public class Box3DApp {
   public static void main(String[] args) {
      // creates a drawing frame and a drawing panel
      Display3DFrame frame = new Display3DFrame("3D demo");
      frame.setPreferredMinMax(-10, 10, -10, 10, -10, 10);
      frame.setDecorationType(VisualizationHints.DECORATION_AXES);
      // use shading when rotating Element
      frame.setAllowQuickRedraw(false);
      block = new ElementBox();
      block.setXYZ(0, 0, 0);
      block.setSizeXYZ(6, 6, 3);
      block.getStyle().setFillColor(Color.RED);
      // divides block into subblocks
      block.getStyle().setResolution(new Resolution(6, 6, 3));
      frame.addElement(block);
      frame.setVisible(true);
      frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
   }
}
```

Drawable objects in the `simple3d` package implement the `Element` interface. The `Box3DApp` program demonstrates that this interface mimics simple geometric objects in the physical world. Elements have position and size properties and contain `Style` and `Resolution` objects to control their visual representation.

The `Display3DFrame` class behaves very much like its two-dimensional `DisplayFrame` counterpart. Some methods such as `setPreferredMinMax` have been extended by adding parameters for the third dimension. Methods such as `getStyle` and `setDecorationType` enable the programmer to control the three-dimensional viewing perspective and axis types, respectively. Three-dimensional elements, such as `ElementBox`, `ElementCylinder`, and `ElementCircle`, are added to a container using the `addElement` method. As in the two-dimensional case, the high level `Display3DFrame` is composed of lower level objects such as `DrawingPanel3D` which fill the frame's viewing area. The program must repaint the frame for property changes to take effect.

Elements generate interaction events and these events can have one or more interaction targets. Interaction events are action events that are generated when an object is accessed using the mouse. Interaction targets receive and process these events. Listing 17.6 creates a particle and an arrow and responds to interaction events when these objects are moved. You can move an interactive element by dragging it. You can also generate and process interaction events at an arbitrary location within the viewing space, but you might wish to first click the ⟨alt⟩ (⟨option⟩ on Mac OS X) key to disable the viewing space rotation. Clicking the ⟨shift⟩ key while dragging enables you to zoom in and out. Clicking the ⟨control⟩ key while dragging enables you to translate.

Listing 17.6 The `Interaction3DApp` class demonstrates how to respond to interaction events.

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