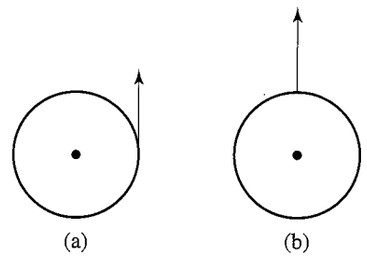

Figure 5.4 (a) An impulse applied in the tangential direction. (b) An impulse applied in the radial direction.

the print statements with useful methods. Other mouse actions, such as `MOUSE_CLICKED`, `MOUSE_MOVED`, and `MOUSE_ENTERED` are defined in the `InteractivePanel` class.

Listing 5.4 `InteractiveMouseHandler` interface test program.

```
package org.opensourcephysics.sip.ch05;
import java.awt.event.*;
import javax.swing.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.frames.*;

public class MouseApp implements InteractiveMouseHandler {
   PlotFrame frame = new PlotFrame("x", "y", "Interactive Handler");

   public MouseApp() {
      frame.setInteractiveMouseHandler(this);
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }

   public void handleMouseAction(InteractivePanel panel,
         MouseEvent evt) {
      switch(panel.getMouseAction()) {
      case InteractivePanel.MOUSE_DRAGGED:
         panel.setMessage("Dragged");
         break;
      case InteractivePanel.MOUSE_PRESSED:
         panel.setMessage("Pressed");
         break;
      case InteractivePanel.MOUSE_RELEASED:
         panel.setMessage(null);
         break;
      }
   }

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

The `switch` statement is used in Listing 5.4 instead of a chain of `if` statements. The panel's `getMouseAction` method returns an integer. If this integer matches one of the named constants following the `case` label, then the statements following that constant are executed until a `break` statement is encountered. If a `case` does not include a `break`, then the execution continues with the next `case`. The equivalent of the `else` construct in an `if` statement is `default` followed by statements that are executed if none of the explicit cases occur.

We now challenge your intuitive understanding of Newton's laws of motion by considering several perturbations of the motion of an orbiting object. Modify your planet program to simulate the effects of the perturbations in Problem 5.6. In each case answer the questions before doing the simulation.

Problem 5.6 Tangential and radial perturbations

(a) Suppose that a small tangential "kick" or impulsive force is applied to a satellite in a circular orbit about the Earth (see Figure 5.4a). Choose Earth units so that the numerical value of the product Gm is given by (5.23). Apply the impulsive force by stopping the program after the satellite has made several revolutions and click the mouse to apply the force. Recall that the impulse changes the momentum in the desired direction. In what direction does the orbit change? Is the orbit stable; for example, does a small impulse lead to a small change in the orbit? Does the orbit retrace itself indefinitely if no further perturbations are applied? Describe the shape of the perturbed orbit.

(b) How does the change in the orbit depend on the strength of the kick and its duration?

(c) Determine if the angular momentum and the total energy are changed by the perturbation.

(d) Apply a radial kick to the satellite as in Figure 5.4b and answer the same questions as in parts (a)–(c).

(e) Determine the stability of the inverse-cube force law (see Problem 5.4) to radial and tangential perturbations. ■

Mouse actions are not the only possible way to affect the simulation. We can also add *custom buttons* to the control. These buttons are added when the program is instantiated in the `main` method.

```
public static void main(String[] args) {
   // OSPControl is a superclass of SimulationControl
   OSPControl control = SimulationControl.createApp(new PlanetApp());
   control.addButton("doRadialKick", "Kick!", "Perform a radial kick");
}
```

Note that `SimulationControl` (and `CalculationControl`) extend the `OSPControl` superclass and, therefore, support the `addButton` method where this method is defined. We assign the variable returned by the static `createApp` method to a variable of type `OSPControl` to highlight the object-oriented structure of the Open Source Physics library.

The first parameter in the `addButton` method specifies the method that will be invoked when the button is clicked, the second parameter specifies the text label that will appear on the button, and the third parameter specifies the *tool tip* that will appear when the mouse hovers over the button. Custom buttons can be used for just about anything, but the corresponding method must be defined.