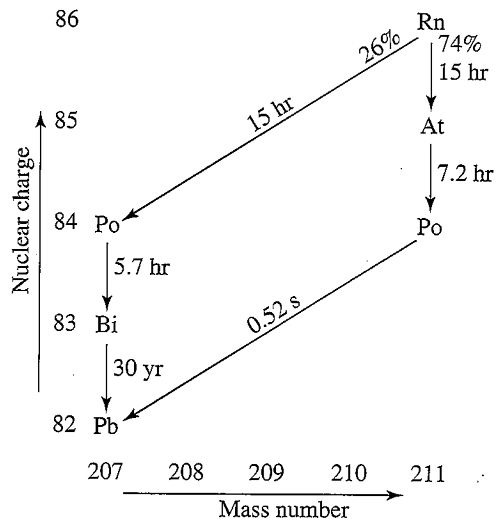

Figure 3.4 The decay scheme of ^{211}Rn. Note that ^{211}Rn decays via two branches, and the final product is the stable isotope ^{207}Pb. All vertical transitions are by electron capture, and all diagonal transitions are by alpha decay. The times represent half-lives.

Problem 3.14 Cooling of a cup of coffee

The nature of the energy transfer from the hot water in a cup of coffee to the surrounding air is complicated and, in general, involves the mechanisms of convection, radiation, evaporation, and conduction. However, if the temperature difference between the water and its surroundings is not too large, the rate of change of the temperature of the water may be assumed to be proportional to the temperature difference. We can formulate this statement more precisely in terms of a differential equation:

$$\frac{dT}{dt} = -r\,(T - T_s), \tag{3.21}$$

where T is the temperature of the water, T_s is the temperature of its surroundings, and r is the cooling constant. The minus sign in (3.21) implies that if $T > T_s$, the temperature of the water will decrease with time. The value of the cooling constant r depends on the heat transfer mechanism, the contact area with the surroundings, and the thermal properties of the water. The relation (3.21) is sometimes known as Newton's law of cooling, even though the relation is only approximate, and Newton did not express the rate of cooling in this form.

(a) Write a program that computes the numerical solution of (3.21). Test your program by choosing the initial temperature $T_0 = 100°\text{C}$, $T_s = 0°\text{C}$, $r = 1$, and $\Delta t = 0.1$.

(b) Model the cooling of a cup of coffee by choosing $r = 0.03$. What are the units of r? Plot the temperature T as a function of the time using $T_0 = 87\,°\text{C}$ and $T_s = 17\,°\text{C}$. Make sure that your value of Δt is sufficiently small so that it does not affect your results. What is the appropriate unit of time in this case?

(c) Suppose that the initial temperature of a cup of coffee is 87°C, but the coffee can be sipped comfortably only when its temperature is $\leq 75°\text{C}$. Assume that the addition of cream cools the coffee by 5°C. If you are in a hurry and want to wait the shortest possible time, should the cream be added first and the coffee be allowed to cool, or should you wait until the coffee has cooled to 80°C before adding the cream? Use your program to "simulate" these two cases. Choose $r = 0.03$ and $T_s = 17°\text{C}$. What is the appropriate unit of time in this case? Assume that the value of r does not change when the cream is added. ■

*3.10 ■ VISUALIZING THREE-DIMENSIONAL MOTION

The world in which we live is three-dimensional (3D), and it sometimes is necessary to visualize phenomena in three dimensions. There are several 3D visualization packages available, including *Java3D* developed by Sun Microsystems and *Java OpenGL* (JOGL), which has been optimized for a variety of graphics hardware. Because we want a three-dimensional visualization framework designed for physics simulations, we have developed our own API.[1]

The Open Source Physics 3D drawing framework is defined in subpackages in the `display3d` package and provides a high level of abstraction for rendering three-dimensional objects. These 3D drawable objects implement the `Element` interface in the `core` package, which enables their position, size, and appearance to be controlled. Elements can be grouped with other elements, can change their visibility, and can respond to mouse actions. Listing 3.11 shows that it is not much more difficult to define and manipulate a three-dimensional model than a two-dimensional model. The most significant change is that the program instantiates a `Display3DFrame` and adds `Element` objects such as spheres and boxes to this frame.

Listing 3.11 A three-dimensional bouncing ball created using the Open Source Physics `display3D.simple3d` package.

```
package org.opensourcephysics.sip.ch03;
import java.awt.*;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.Display3DFrame;
import org.opensourcephysics.display3d.simple3d.*;
import org.opensourcephysics.display3d.core.Resolution;

public class Ball3DApp extends AbstractSimulation {
   Display3DFrame frame = new Display3DFrame("3D Ball");
   Element ball = new ElementEllipsoid();
   double time = 0, dt = 0.1;
   double vz = 0;

   public Ball3DApp() {
      frame.setPreferredMinMax(-5.0, 5.0, -5.0, 5.0, 0.0, 10.0);
      ball.setXYZ(0, 0, 9);
```

[1]A framework consists of several classes and an API that does a particular task. In general, these classes are in different packages.