The UBC Java graphics tutorial—a simple animated applet

One of the really enjoyable features of Java is the ability to make animations. Most animations you see on the Internet are made up of a sequence of pre-drawn images, but is also possible to do the drawing in real time. There may be performance limitations, but simple mathematical 'movies' are quite practical. Here is one of the simplest mathematical animations you can make using `psplot`:

```
import java.applet.*;
import java.awt.*;
import psplot.*;
import real.*;

class psAnimationCanvas extends PlotCanvas implements Runnable {
    public psAnimationCanvas(Applet app, int w, int h) {
        super(app, w, h);
        double ht = 3;
        // preserve aspect ratio
        double wd = w*(ht/h);
        setCorners(-wd, -ht, wd, ht);
    }

    public void draw() {
        PlotPath p;
        p = new PlotPath(this);
        p.moveto(-0.5, -0.5);
        p.lineto( 0.5, -0.5);
        p.lineto( 0.5,  0.5);
        p.lineto(-0.5, 0.5);
        p.close();
        p.fill(Color.red);
        p.stroke(Color.black);
    }

boolean ok = true;

    public void run() {
        double t = 0.05;
        while (ok) {
            rotate(t);
            // translate(1+t, 1+t);
            // scale(1+t, 1+t);
            repaint();
            try {
                Thread.sleep(100);
            } catch (InterruptedException ie) { ; }
        }
    }
}

public class psAnimationApplet extends Applet {
    psAnimationCanvas p;
    Thread th;
```

```
    public void init() {
        p = new psAnimationCanvas(this, bounds().width, bounds().height);
        add(p);
        show();
    }

    public void start() {
        p.ok = true;
        th = new Thread(p);
        th.start();
    }

    public void stop() {
        p.ok = false;
    }
}
```

This has much in common with the simple graphics applet we looked at before, but there are also a number of new features. They are almost all related to the one basic Java class `Thread`. Threads are a general concept in all of computing, but in Java they play a prominent role. They are explained at length in most Java texts, and I am not going to repeat what they say. I think (hope)that it will suffice to have for our purposes a rough intuitive notion in mind: just think of the animation as being a little engine running away inside your canvas. Like all engines, we have to turn it on when we want it to go and turn it off when we want it to stop. The routine `start()` is called automatically by the browser when your applet is displayed, much like `init()` is called when your applet is created. When your applet is shut down by the browser, it calls `stop()`. In this `stop()` procedure the variable `ok` in the canvas is set to false. This causes the thread running in the canvas to exit from its loop and then exit entirely.

It is extremely important to have this `stop()` *procedure in your applet.* Without it, your program will keep running even after the browser quits the site running your applet. This is terrible behaviour, and will cause all sorts of strangers to refer to you in foul language. It is not at all unusual to find your computer speeding away after you thought you had quit browsing, because of applets which didn't stop themselves correctly.

When the thread is running in this applet, it rotates the coordinate system just as it might in PostScript. It might also have scaled it, or translated it. After changing the coordinate system it redraws the picture, and then it turns itself off for a little while, in a sense to let things cool off, and also to get the animation at the correct speed. There are lots of very technical issues connected with sleep time. One thing to keep in mind is the practical observation that if an animation doesn't sleep long enough, some computers (often running Windows '95, it seems) have trouble—you might say they get overheated.