Richard G Baldwin (512) 223-4758, baldwin@austin.cc.tx.us,
http://www2.austin.cc.tx.us/baldwin/

The AWT Package, Graphics- Working with Shapes

Java Programming, Lecture Notes # 166, Revised 02/16/98.

Preface

Students in Prof. Baldwin's **Advanced Java Programming** classes at ACC are responsible for knowing and understanding all of the material in this lesson.

Introduction

A previous lesson provided an overview of the **Graphics** class, and grouped the methods of that class into several different categories. This lesson will explore some of the methods in the category of **Drawing and Filling Shapes**.

To review, the following methods were put into this category in the previous lesson.

drawLine(int, int, int, int) - Draws a line, using the current color, between two points in this graphics context's coordinate system.

drawPolyline(int[], int[], int) - Draws a sequence of connected lines defined by arrays of *x* and *y* coordinates. The figure will not be closed if the first point differs from the last point.

drawRect(int, int, int, int) - Draws the outline of the specified rectangle using the current color of the graphics context..

fillRect(int, int, int, int) - Fills the specified rectangle with the context's current color. Be sure to check the documentation regarding the coordinates of the right edge and bottom edge of the rectangle before using. This comment applies to all the fill methods.

drawRoundRect(int, int, int, int, int, int) - Draws an outlined round-cornered rectangle using this graphics context's current color. You might need to look at a book containing a diagram to learn how to specify how the corners are rounded.

fillRoundRect(int, int, int, int, int, int) - Fills the specified rounded corner rectangle with the current color.

draw3DRect(int, int, int, int, boolean) - Draws a 3-D highlighted outline of the specified rectangle. The edges of the rectangle are highlighted so that they appear to be beveled and lit from the upper left corner. The boolean parameter determines whether the rectangle appears to be raised above the surface or sunk into the surface. It is raised when the parameter is true.

fill3DRect(int, int, int, int, boolean) - Paints a 3-D highlighted rectangle filled with the current color.

drawOval(int, int, int, int) - Draws the outline of an oval in the current color. When the last two parameters are equal, this method draws a circle.

fillOval(int, int, int, int) - Fills an oval bounded by the specified rectangle with the current color. As with drawOval(), when the last two parameters are equal, the method fills a circle.

drawArc(int, int, int, int, int, int) - Draws the outline of a circular or elliptical arc covering the specified rectangle. You will probably need to examine the documentation to figure out how to specify the parameters for this method as well as the fillArc() method.

fillArc(int, int, int, int, int, int) - Fills a circular or elliptical arc covering the specified rectangle.

drawPolygon(Polygon) - Draws the outline of a polygon defined by the specified **Polygon** object. Another overloaded version is available that accepts a list of coordinate values to specify the polygon. The following description of a **Polygon** object was taken from the JavaSoft documentation for JDK 1.1.3.

fillPolygon(Polygon) - Fills the polygon defined by the specified Polygon object with the graphics context's current color. Another overloaded version is available that accepts a list of coordinate values to specify the polygon.

The material in this lesson is not very difficult in comparison with material in previous lessons. Only a very few of these methods (such as **drawArc()**) involve any complexity at all, and once you see an explanation of how to construct the parameter list, even those methods turn out to be easy to use.

Sample Program

The following program contains sample drawings of all the shapes in the above list. A few of the shapes are drawn more than once to illustrate different aspects of the associated method (ovals and circles for example).

If you compile and run the program, and then review the source code while viewing the screen output (and consult the JDK documentation describing the various methods of the **Graphics** class along the way), you should be able to understand everything that you need to know about drawing shapes.

I do want to comment on the methods **draw3DRect()** and **fill3DRect()**. These two methods are supposed to produce rectangles that appear to be raised from the screen or depressed into the screen. This effect is achieved by highlighting two edges while darkening the other two edges to give the illusion of light and shadow. (Also the entire rectangle is darkened for the depressed version of **fill3DRect()**.) However, it doesn't work very well using JDK 1.1.3 under Win95. It was necessary for me to select my drawing and background colors very carefully to create any illusion of 3D at all. Simply drawing or filling them in red on a white background did not produce images that had a 3D effect.

```
/*File Graphics07.java
Copyright 1997, R.G.Baldwin

This program illustrates the drawing of all the shapes
available in JDK 1.1.3.

You will probably need to compile and run this program
to gain a good feel for how each shape is controlled by
its argument list.

This program was tested using JDK 1.1.3 under Win95.

**********************************************************/
import java.awt.*;
import java.awt.event.*;

class Graphics07 extends Frame{ //controlling class
  //Override the paint method
  public void paint(Graphics g){
    g.setColor(Color.red);//set the drawing color to red

    //Translate the 0,0 coordinate of the graphics context
    // to the upper left-hand corner of the client area of
    // the Frame object.
    g.translate(
               this.getInsets().left,this.getInsets().top);
    //Draw a simple line
    g.drawLine(0,0,50,50);

    //Create two arrays of coordinate data and draw a
    // polyline using that data
    int xData[] = {100,110,120,130,140,150};
    int yData[] = {0,25,37,43,47,50};
    g.drawPolyline(xData,yData,6);

    //Draw a rectangle
    g.drawRect(200,0,50,25);

    //Draw a filled rectangle
    g.fillRect(300,0,25,50);
    //Draw a rounded rectangle
    g.drawRoundRect(400,0,50,50,45,35);

    //Fill a rounded rectangle
    g.fillRoundRect(500,0,50,50,35,45);

    //Draw a 3D rectangle, raised, doesn't look very 3D,
    // but a slight hint of 3D can be seen if drawn in
    // gray on on a white background.
    g.setColor(Color.gray);//draw the 3D stuff in gray
    g.draw3DRect(100,100,50,25,true);
    //Draw a 3D rectangle, depressed
    g.draw3DRect(200,100,50,25,false);

    //Fill a 3D rectangle, raised.  Can be made to look
    // pretty decent if drawn in gray on a gray background.
    // Otherwise, doesn't look very 3D.
    g.fillRect(275,75,75,100);//draw a gray background
    g.fill3DRect(300,100,25,50,true);
    //Fill a 3D rectangle, depressed, doesn't look very 3D
    g.fillRect(375,75,75,100);//draw a gray background
    g.fill3DRect(400,100,25,50,false);
    g.setColor(Color.red);//restore red color
    //Overall, the appearance of the 3DRect is not very
    // good because it is necessary to select special
    // colors and backgrounds to get the illusion of 3D.

    //Draw an oval, long axis on horizontal axis
    g.drawOval(0,200,50,25);
    //Fill an oval, long axos on vertical axis
    g.fillOval(100,200,25,50);
    //Fill an oval, which is a circle
    g.fillOval(200,200,50,50);

    //Draw a 225-degree arc inside a bounding rectangle
    g.drawRect(300,200,25,50);
    g.drawArc(300,200,25,50,0,225);

    //Fill a 225-degree arc inside a bounding rectangle
    g.drawRect(400,200,25,50);
    g.fillArc(400,200,25,50,0,225);

    //Draw a polygon using array data as parameters
    int xxData[] = {0,10,20,30,40,50};
    int yyData[] = {300,325,337,343,347,350};
    g.drawPolygon(xxData,yyData,6);

    //Fill a polygon using array data as parameters
    int xxxData[] = {100,110,120,130,140,150};
    int yyyData[] = {300,325,337,343,347,350};
    g.fillPolygon(xxxData,yyyData,6);

  }//end paint()

  public Graphics07(){//constructor
    this.setTitle(
             "Shape Samples,Copyright 1997, R.G.Baldwin");
    this.setSize(600,400);
    this.setVisible(true);

    //Anonymous inner-class listener to terminate program
    this.addWindowListener(
      new WindowAdapter(){//anonymous class definition
        public void windowClosing(WindowEvent e){
          System.exit(0);//terminate the program
        }//end windowClosing()
      }//end WindowAdapter
    );//end addWindowListener
  }//end constructor

  public static void main(String[] args){
    new Graphics07();//instantiate this object
  }//end main
}//end Graphics07 class
//=========================================================//
```

-end-