Programming in Java 2D Package

http://java.sun.com/docs/books/tutorial/2d/index.html

Overview

Provide two-dimensional graphics, text, imaging capabilities

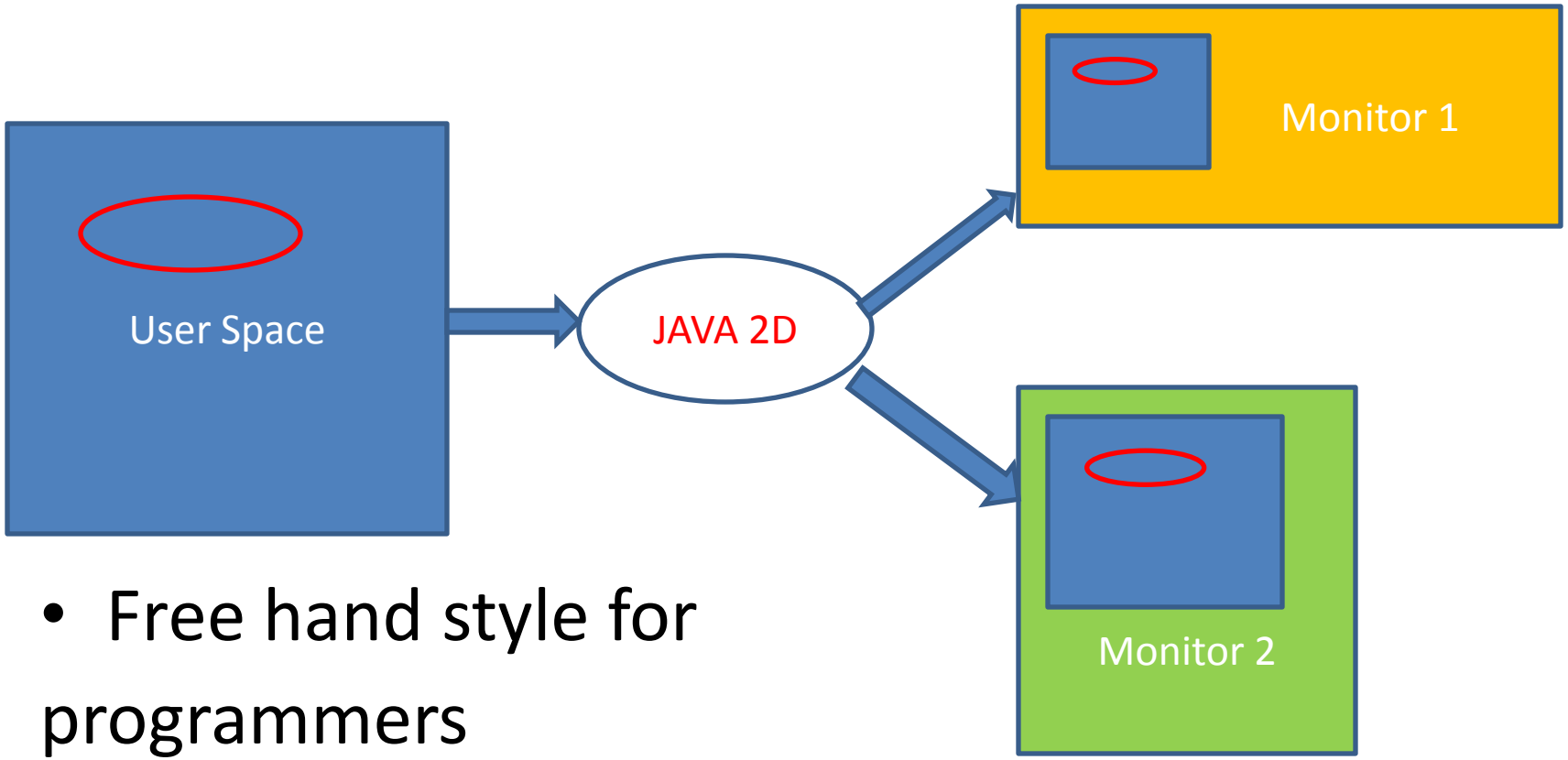
- Uniform rendering model for different output devices (displays and printers)
- Several geometric primitives (shapes)
- Mechanisms for performing hit detection on shapes, text, images
- Compositing model (overlapping objects)
- Color management
- Printing
- Control of quality of the rendering

Coordinates

- User space: the space in which graphics primitives are specified
  - device-independent logical system (could be very large, very small, etc.)
  - used by users
  - every graphics primitive (of the program) is specified in the user space coordinate system
- Device space: the coordinate system of an output device
  - device-dependent (fixed for each device)
  - origin is top-left corner (x increases to the right, y increases downward)
  - coordinates are usually integers

Rendering model

- Uniform across different types of devices



- Free hand style for programmers

Rendering model

- Outline of any geometry primitive (stroke, paint attribute)
- Filling is interior with the color or pattern specified by the paint attributes
- Text string
- Image

Geometry Primitives

- Point
- Lines
- Rectangular Shapes
- Quadratic and Cubic Curves
- Arbitrary Shapes
- Areas

http://java.sun.com/docs/books/tutorial/2d/geometry/primitives.html

(see example ShapesDemo2D.java)

Text

- Selecting a Font
- Measuring Text
- Advanced Text Display

ShapesDemo2D.java

Packages:

```
import java.awt.*;     //abstract window toolkit
import java.awt.event.*;  //interaction handling
import java.awt.geom.*;  //geometry primitives
import javax.swing.*;       //components, look
                            //and feel
```

ShapesDemo2D.java (Class/Main)

```
public static void main(String s[]) {
    JFrame f = new JFrame("ShapesDemo2D");      // create a JFrame with title
    f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e){
                    System.exit(0);
            }
    });
    JApplet applet = new ShapesDemo2D();        // create an JApplet
    f.getContentPane().add("Center", applet);   // add the applet to the content of the
                                                  // Jframe
    applet.init();
    f.pack();
    f.setSize(new Dimension(550,100));
    f.setVisible(true);
}
```

ShapesDemo2D.java (*init*)

```
public class ShapesDemo2D extends JApplet {
    final static int maxCharHeight = 15;
    final static int minFontSize = 6;
    final static Color bg = Color.white;
    final static Color fg = Color.black;
    final static Color red = Color.red;
    final static Color white = Color.white;
    final static BasicStroke stroke = new BasicStroke(2.0f);
    final static BasicStroke wideStroke = new BasicStroke(8.0f);
    final static float dash1[] = {10.0f};
    final static BasicStroke dashed = new BasicStroke(1.0f,          BasicStroke.CAP_BUTT, BasicStroke.JOIN_MITER,
    10.0f, dash1, 0.0f);
     Dimension totalSize;
    FontMetrics fontMetrics;

    public void init() {
            //Initialize drawing colors
            setBackground(bg);
            setForeground(fg);
    }
```

ShapesDemo2D.java (*pickFont*)

```
FontMetrics pickFont(Graphics2D g2, String longString, int xSpace) {
      boolean fontFits = false;
      Font font = g2.getFont();
      FontMetrics fontMetrics = g2.getFontMetrics();
      int size = font.getSize();
      String name = font.getName();
      int style = font.getStyle();
      while ( !fontFits ) {
            if ( (fontMetrics.getHeight() <= maxCharHeight) &&
                        (fontMetrics.stringWidth(longString) <= xSpace) ) {
                        fontFits = true; }
            else { if ( size <= minFontSize ) { fontFits = true; }
                        else { g2.setFont(font = new Font(name, style, --size));
                        fontMetrics = g2.getFontMetrics(); }
            }
      }
      return fontMetrics;
}  // figure out the right font given the dimension of the applet
```

ShapesDemo2D.java (*paint*)

```
public void paint(Graphics g) {
  // preparation
    Graphics2D g2 = (Graphics2D) g;
    g2.setRenderingHint(RenderingHints.KEY_ANTIAL
          IASING, RenderingHints.VALUE_ANTIALIAS_ON);
    Dimension d = getSize();
    int gridWidth = d.width / 6;
    int gridHeight = d.height / 2;
    fontMetrics = pickFont(g2, "Filled and Stroked
                              GeneralPath", gridWidth);
    Color fg3D = Color.lightGray;
```

ShapesDemo2D.java (*paint-3D rectangle*)

```
g2.setPaint(fg3D);  // set the paint color
g2.draw3DRect(0, 0, d.width - 1, d.height - 1, true);
// draw a 3d-highlighted outline of rectangle with
   the color lightGray, appears to be raised above
   the surface
g2.draw3DRect(3, 3, d.width - 7, d.height - 7, false);
// draw a 3d-highlighted outline of rectangle with
   the color lightGray, appears to be sunk into the
   surface
g2.setPaint(fg);    // set the paint color
```

ShapesDemo2D.java (*paint-Line2D double*)

```
int x = 5; int y = 7;
int rectWidth = gridWidth - 2*x;
int stringY = gridHeight - 3 - fontMetrics.getDescent();
int rectHeight = stringY - fontMetrics.getMaxAscent() - y - 2;
// draw Line2D.Double (line segment: from -- to)
g2.draw(new Line2D.Double(x, y+rectHeight-1, x + rectWidth,
    y));
// draw a string at coordinate (x,StringY)
g2.drawString("Line2D", x, stringY);
x += gridWidth;
```

ShapesDemo2D.java (*paint-Rectangle 2D double*)

```
// draw Rectangle2D.Double
g2.setStroke(stroke);
g2.draw(new Rectangle2D.Double(x, y,
    rectWidth, rectHeight));
g2.drawString("Rectangle2D", x, stringY);
x += gridWidth;
```

ShapesDemo2D.java (*paint-Rectangle 2D double*)

```
// draw RoundRectangle2D.Double
g2.setStroke(dashed);
g2.draw(new RoundRectangle2D.Double(x, y,
    rectWidth, rectHeight, 10, 10));
g2.drawString("RoundRectangle2D", x, stringY);
x += gridWidth;
```

ShapesDemo2D.java
(*paint-Arc 2D double, Eclipse, Polygon*)

```
// draw Arc2D.Double g2.setStroke(wideStroke);
g2.draw(new Arc2D.Double(x, y, rectWidth, rectHeight, 90, 135, Arc2D.OPEN));
g2.drawString("Arc2D", x, stringY); x += gridWidth;

// draw Ellipse2D.Double
g2.setStroke(stroke);
g2.draw(new Ellipse2D.Double(x, y, rectWidth, rectHeight));
g2.drawString("Ellipse2D",
x, stringY); x += gridWidth;

// draw GeneralPath (polygon)
int x1Points[] = {x, x+rectWidth, x, x+rectWidth};
int y1Points[] = {y, y+rectHeight, y+rectHeight, y};
GeneralPath polygon = new GeneralPath(GeneralPath.WIND_EVEN_ODD, x1Points.length);
polygon.moveTo(x1Points[0], y1Points[0]);
for ( int index = 1; index < x1Points.length; index++ ) {
        polygon.lineTo(x1Points[index], y1Points[index]);
};
polygon.closePath();
g2.draw(polygon);
g2.drawString("GeneralPath", x, stringY);
```

ShapesDemo2D.java (*paint-Polygon*)

```
// NEW ROW
x = 5;
y += gridHeight;
stringY += gridHeight;
// draw GeneralPath  (polyline)
int x2Points[] = {x, x+rectWidth, x, x+rectWidth};
int y2Points[] = {y, y+rectHeight, y+rectHeight, y};
GeneralPath polyline = new GeneralPath(GeneralPath.WIND_EVEN_ODD,
    x2Points.length);
polyline.moveTo (x2Points[0], y2Points[0]);
for ( int index = 1; index < x2Points.length; index++ ) { polyline.lineTo(x2Points[index],
    y2Points[index]);
};
g2.draw(polyline);
g2.drawString("GeneralPath (open)", x, stringY);
x += gridWidth;
```

ShapesDemo2D.java (*paint-fill rectangles, arc*)

```
// fill Rectangle2D.Double (red)
g2.setPaint(red);
g2.fill(new Rectangle2D.Double(x, y, rectWidth, rectHeight));
 g2.setPaint(fg);
g2.drawString("Filled Rectangle2D", x, stringY);
x += gridWidth;

// fill RoundRectangle2D.Double
GradientPaint redtowhite = new GradientPaint(x,y,red,x+rectWidth, y,white);
 g2.setPaint(redtowhite);
g2.fill(new RoundRectangle2D.Double(x, y, rectWidth, rectHeight, 10, 10));
g2.setPaint(fg);
g2.drawString("Filled RoundRectangle2D", x, stringY); x += gridWidth;

// fill Arc2D
g2.setPaint(red);
g2.fill(new Arc2D.Double(x, y, rectWidth, rectHeight, 90, 135, Arc2D.OPEN));
g2.setPaint(fg);
g2.drawString("Filled Arc2D", x, stringY);
x += gridWidth;
```

ShapesDemo2D.java (*paint-fill Eclipse, Polygon*)

```
// fill Ellipse2D.Double
redtowhite = new GradientPaint(x,y,red,x+rectWidth, y,white);
g2.setPaint(redtowhite);
g2.fill (new Ellipse2D.Double(x, y, rectWidth, rectHeight));
g2.setPaint(fg);
g2.drawString("Filled Ellipse2D", x, stringY);
x += gridWidth;
// fill and stroke GeneralPath
int x3Points[] = {x, x+rectWidth, x, x+rectWidth};
int y3Points[] = {y, y+rectHeight, y+rectHeight, y};
GeneralPath filledPolygon = new GeneralPath(GeneralPath.WIND_EVEN_ODD, x3Points.length);
filledPolygon.moveTo(x3Points[0], y3Points[0]);
for ( int index = 1; index < x3Points.length; index++ ) {
      filledPolygon.lineTo(x3Points[index], y3Points[index]);
};
filledPolygon.closePath();
g2.setPaint(red); g2.fill(filledPolygon);
g2.setPaint(fg); g2.draw(filledPolygon);
g2.drawString("Filled and Stroked GeneralPath", x, stringY);
```