JAVA™ PROGRAMMING

Chapter 16: Graphics

Objectives

- Learn about rendering methods
- Draw strings
- Draw lines and shapes
- Learn more about fonts
- Draw with Java 2D graphics

Learning About Rendering Methods

- **Render**
  - To redisplay a display surface
- **Painting**
  - System-triggered painting
  - Application-triggered painting
- **`paint()` method**
  - Write your own method to override the default
  - Method header
    - `public void paint(Graphics g)`

Learning About Rendering Methods (cont'd.)

- `Graphics` object
  - Preconfigured with the appropriate values for drawing on the component
- **`repaint()` method**
  - Use when a window needs to be updated
  - Calls the `paintComponent()` method
  - Creates a `Graphics` object

Learning About Rendering Methods (cont'd.)

- When Swing object calls `repaint()` these methods are called:
  - `paint(), paintComponent(), paintBorder(), paintChildren()`
- Place all drawing code in `paintComponent()` method

```
Public void paintComponent (Graphics g);
```

  - Typical first statement

```
Super.paintComponent(g);
```

Learning About Rendering Methods (cont'd.)

```
import javax.swing.*;
import java.awt.*;
import java.awt.Color;
public class JColorPanel extends JPanel
{
   int count = 0;
   String colorString;
   public JColorPanel(Color color)
   {
      if(color.equals(Color.RED))
         colorString = "red";
      else
         colorString = "blue";
      setBackground(color);
   }
   @Override
   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      ++count;
      System.out.println("In paintComponent() method -- " +
         colorString + " " + count);
   }
}
```

Figure 16-1 The JColorPanel class

Learning About Rendering Methods (cont'd.)

```
import javax.swing.*;
import java.awt.*;
import java.awt.Color;
public class JDemoPaintComponent extends JFrame
{
   JColorPanel p1 = new JColorPanel(Color.RED);
   JColorPanel p2 = new JColorPanel(Color.BLUE);
   public JDemoPaintComponent()
   {
      setLayout(new BorderLayout());
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      getContentPane().setBackground(Color.YELLOW);
      add(p1, BorderLayout.EAST);
      add(p2, BorderLayout.SOUTH);
   }
   public static void main(String[] args)
   {
      JDemoPaintComponent frame = new JDemoPaintComponent();
      frame.setSize(150, 100);
      frame.setVisible(true);
   }
}
```

Figure 16-2 The JDemoPaintComponent class

Drawing Strings

- **`drawString()` method**
  - Allows you to draw a `String` in a `JFrame` window
  - Requires three arguments:
    - `String`
    - x-axis coordinate
    - y-axis coordinate
  - Is a member of the `Graphics` class
  - Repainting: Can execute `paintComponent()` even when no draw-triggering changes have been made

Setting A Font

- `setFont()` method
  - Requires a `Font` object
- You can instruct a `Graphics` object to use a font
  - `somegraphicsobject.setFont(someFont);`

Using Color

- `setColor()` method
  - Designates a `Graphics` color
  - Use 13 `Color` class constants as arguments

```
brush.setColor(Color.GREEN);
```

Drawing Lines and Shapes

- Java provides several methods for drawing a variety of lines and geometric shapes

Drawing Lines

- **`drawLine()` method**
  - Draws a straight line between any two points
  - Takes four arguments:
    - x- and y-coordinates of the line's starting point
    - x- and y-coordinates of the line's ending point

Drawing Unfilled and Filled Rectangles

- **`drawRect()` method**
  - Draws the outline of a rectangle
- **`fillRect()` method**
  - Draws a solid or filled rectangle
- Both require four arguments:
  - x- and y-coordinates of the upper-left corner of the rectangle
  - The width and height of the rectangle

Drawing Clear Rectangles

- **`clearRect()` method**
  - Draws a rectangle
  - Requires four arguments:
    - x- and y-coordinates of the upper-left corner of the rectangle
    - The width and height of the rectangle
  - Appears empty or “clear”

Drawing Rounded Rectangles

- **`drawRoundRect()`** method
  - Creates rectangles with rounded corners
  - Requires six arguments
- **`fillRoundRect()`** method

Drawing Rounded Rectangles (cont'd.)

```
import javax.swing.*;
import java.awt.*;
public class JDemoRoundedRectangles extends JPanel
{
   @Override
   public void paintComponent(Graphics gr)
   {
      super.paintComponent(gr);
      int x = 20;
      int y = 40;
      final int WIDTH = 80, HEIGHT = 80;
      final int ARC_INCREASE = 20;
      final int HORIZONTAL_GAP = 100;
      for(int size = x; size <= HEIGHT; size += ARC_INCREASE)
      {
         gr.drawRoundRect(x, y, WIDTH, HEIGHT, size, size);
         x += HORIZONTAL_GAP;
      }
   }
   public static void main(String[] args)
   {
      JFrame frame = new JFrame();
      frame.add(new JDemoRoundedRectangles());
      frame.setSize(430, 180);
      frame.setVisible(true);
   }
}
```

Figure 16-17 The JDemoRoundedRectangles class

Drawing Rounded Rectangles (cont'd.)

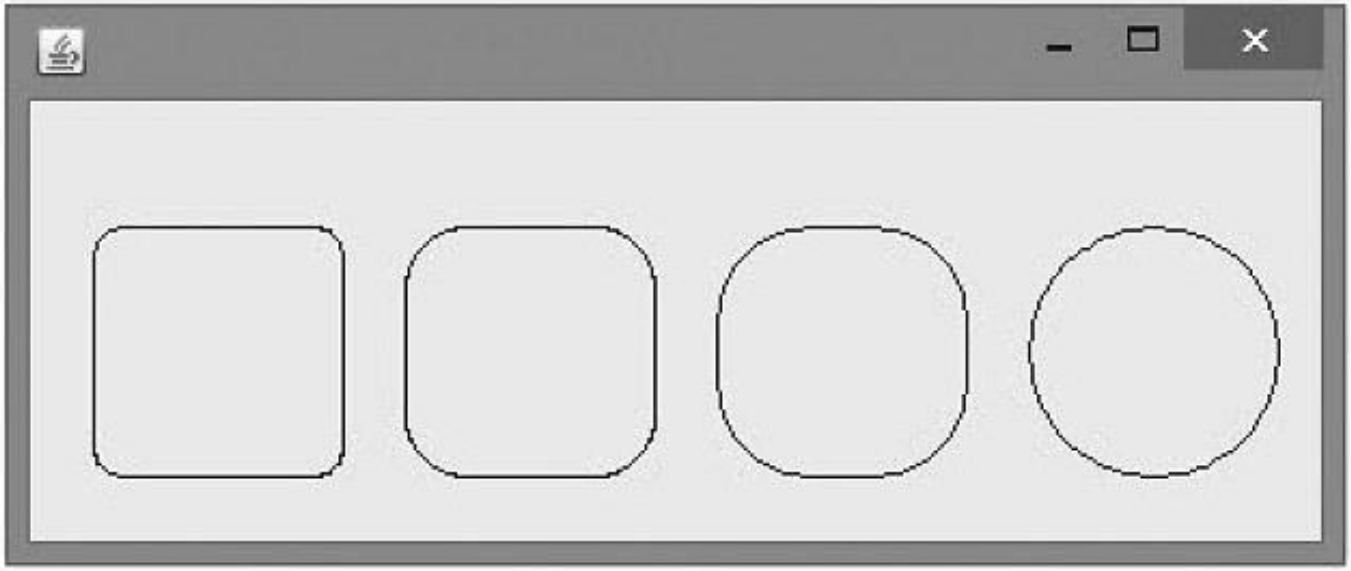

Figure 16-18 Output of the `JDemoRoundedRectangles` program

Creating Shadowed Rectangles

- **`draw3DRect()` method**
  - A minor variation on the `drawRect()` method
  - Draws a rectangle that appears to have “shadowing” on two edges
  - Contains a Boolean value argument:
    - `true` if the rectangle is darker on the right and bottom
    - `false` if the rectangle is darker on the left and top
- **`fill3DRect()` method**
  - Creates filled three-dimensional rectangles

Drawing Ovals

- **`drawOval()` and `fillOval()` methods**
  - Draw ovals using the same four arguments that rectangles use

Drawing Arcs

- **`drawArc()` method** arguments
  - x- and y-coordinates of the upper-left corner of an imaginary rectangle that represents the bounds of the imaginary circle that contains the arc
  - The width and height of the imaginary rectangle that represents the bounds of the imaginary circle that contains the arc
  - The beginning arc position
  - The arc angle

Drawing Arcs (cont'd.)

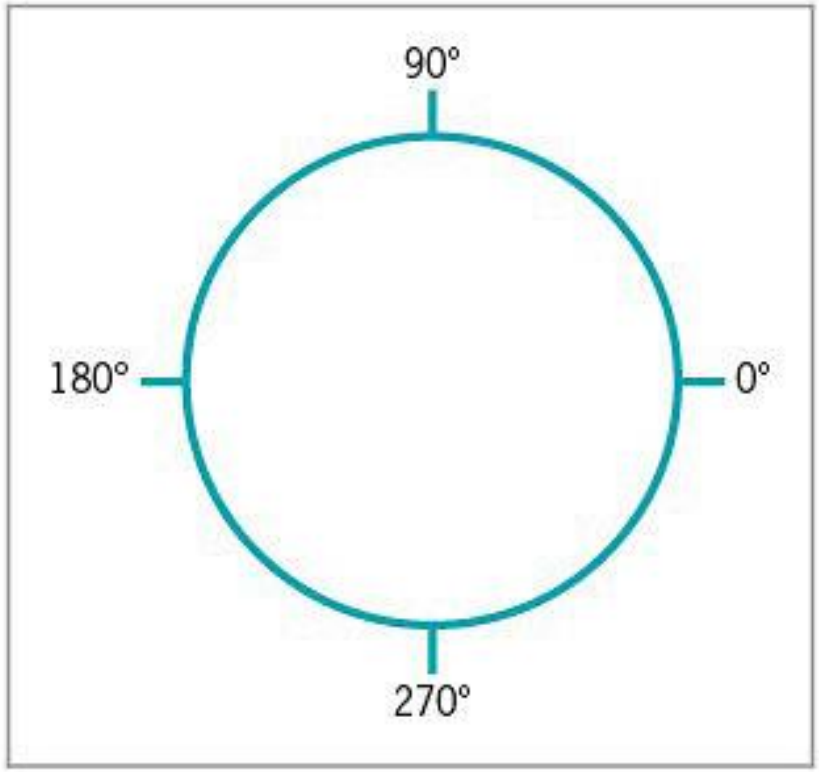

Figure 16-22 Arc positions

Drawing Arcs (cont'd.)

- **`fillArc()` method**
  - Creates a solid arc
    - Two straight lines are drawn from the arc endpoints to the center of the imaginary circle whose perimeter the arc occupies

Creating Polygons

- **`Polygon:`** `geometric figure with straight sides`
- **`drawPolygon()` method**
  - Draws complex shapes
  - Requires three arguments:
    - The integer array, which holds a series of x-coordinate positions
    - The second array, which holds a series of corresponding y-coordinate positions
    - The number of pairs of points to connect

Creating Polygons (cont'd.)

- **`fillPolygon()` method**
  - Draws a solid shape
  - If the beginning and ending points are not identical, two endpoints are connected by a straight line before the polygon is filled with color
- **`addPoint()` method**
  - Adds points to a polygon indefinitely

Creating Polygons (cont'd.)

```
import javax.swing.*;
import java.awt.*;
public class JStar extends JPanel
{
   int xPoints[] = {42, 52, 72, 52, 60, 40, 15, 28, 9, 32, 42};
   int yPoints[] = {38, 62, 68, 80, 105, 85, 102, 75, 58, 60, 38};
   @Override
   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      g.drawPolygon(xPoints, yPoints, xPoints.length);
   }
   public static void main(String[] args)
   {
      JFrame frame = new JFrame();
      frame.add(new JStar());
      frame.setSize(140, 160);
      frame.setVisible(true);
   }
}
```

Figure 16-24 The JStar class

Creating Polygons (cont'd.)

Figure 16-25 Output of the `JStar` program

Copying an Area

- **`copyArea()` method**
  - Requires six parameters:
    - The x- and y-coordinates of the upper-left corner of the area to be copied
    - The width and height of the area to be copied
    - The horizontal and vertical displacement of the destination of the copy

Using the `paint` Method with `JFrames`

- Use the `paintComponent()` method when creating drawings on a `JPanel`
- `JFrame` is not a child of `JComponent`
  - Does not have its own `paintComponent()` method
  - Must override the `paint()` method

Learning More About Fonts

- **`getAvailableFontFamilyNames()` method**
  - Is part of the `GraphicsEnvironment` class defined in the `java.awt` package
  - Returns an array of `String` objects that are names of available fonts
- You cannot instantiate the `GraphicsEnvironment` object directly
  - Get the reference object to the current computer environment
    - Call the static `getLocalGraphicsEnvironment()` method

Discovering Screen Statistics

- **`getDefaultToolkit()` method**
  - Provides information about the system in use
- **`getScreenResolution()` method**
  - Returns the number of pixels as an integer
- You can create a `Toolkit` object and get the screen resolution using the following code:

```
Toolkit tk = Toolkit.getDefaultToolkit();
int resolution = tk.getScreenResolution();
```

Discovering Screen Statistics (cont'd.)

- `Dimension` class
  - Use for representing the width and height of a user interface component
  - Constructors
    - `Dimension()` creates an instance of `Dimension` with a width and height of 0
    - `Dimension(Dimension d)` creates an instance of `Dimension` whose width and height are the same as for the specified dimension
    - `Dimension(int width, int height)` constructs a `Dimension` and initializes it to the specified width and height

Discovering Screen Statistics (cont'd.)

- **`getScreenSize()` method**
  - Is a member of the `Toolkit` object
  - Returns an object of type `Dimension`, which specifies the width and height of the screen in pixels

Discovering Font Statistics

- **Leading**
  - The amount of space between baselines
- **Ascent**
  - The height of an uppercase character from the baseline to the top of the character
- **Descent**
  - Measures the parts of characters that “hang below” the baseline
- **Height of a font**
  - The sum of leading, ascent, and descent

Discovering Font Statistics (cont'd.)

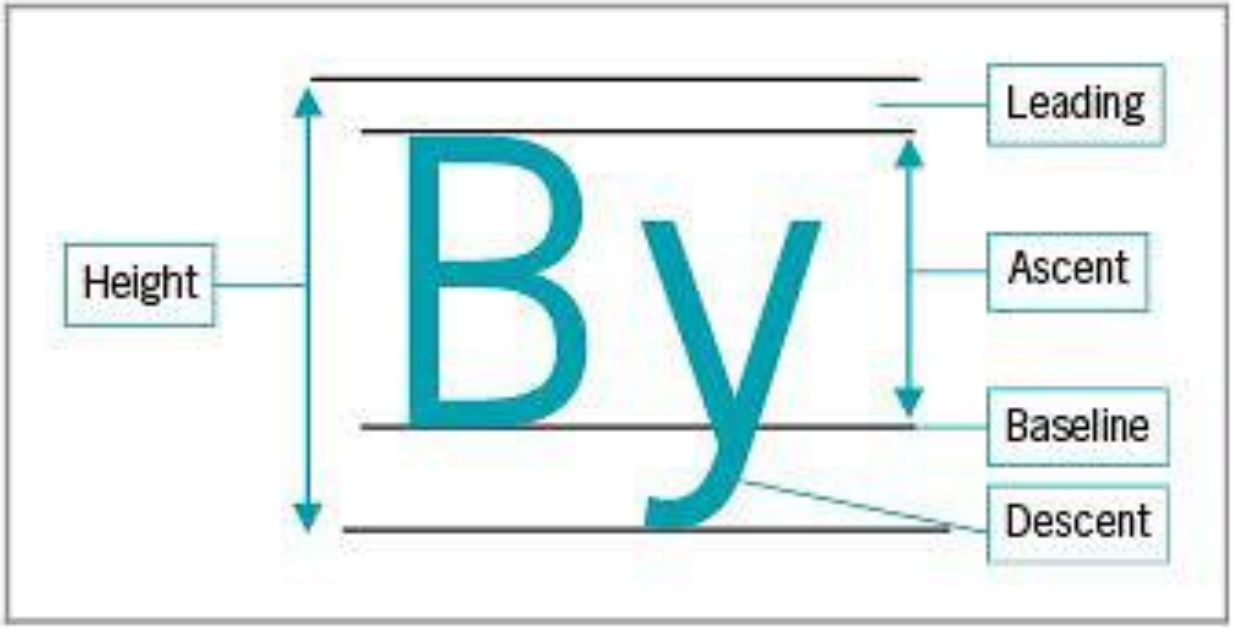

Figure 16-34 Parts of a font's height

Discovering Font Statistics (cont'd.)

- **`getFontMetrics()` method**
  - Discovers a font's height
  - Returns the `FontMetrics` object
- Use one of the `FontMetrics` class methods with the object to return one of a `Font`'s statistics:
  - `public int getLeading()`
  - `public int getAscent()`
  - `public int getDescent()`
  - `public int getHeight()`

Discovering Font Statistics (cont'd.)

- **`stringWidth()` method**
  - Returns the integer width of a `String`
  - Requires the name of the `String`
  - Is a member of the `FontMetrics` class

Drawing with Java 2D Graphics

- Java 2D
  - Higher quality, two-dimensional (2D) graphics, images, and text
- **`Graphics2D` class**
  - Features include:
    - Fill patterns
    - Strokes
    - Anti-aliasing

Drawing with Java 2D Graphics (cont'd.)

- **`Graphics2D` class** (cont'd.)
  - Found in the `java.awt` package
  - Produced by casting, or converting, and promoting a `Graphics` object
- The process of drawing with Java 2D objects
  - Specify the rendering attributes
  - Set a drawing stroke
  - Create objects to draw

Specifying the Rendering Attributes

- Use the `setColor()` method
  - Specify 2D colors
  - Use a `Graphics2D` object and set the color to black
    - `gr2D.setColor(Color.BLACK);`
- **Fill patterns**:
  - Control how a drawing object is filled in
  - Can be a solid, gradient, texture, or pattern
  - Created by using the `setPaint()` method of `Graphics2D` with a fill pattern object

Specifying the Rendering Attributes (cont'd.)

- **Gradient fill**
  - A gradual shift from one color at one coordinate point to a different color at a second coordinate point
  - **Acyclic gradient**
  - **Cyclic gradient**

```java
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
import java.awt.Color;
public class JGradient extends JPanel
{
   int x, y, x2, y2;
   public void paintComponent(Graphics gr)
   {
      super.paintComponent(gr);
      x = 20;
      y = 40;
      x2 = 180;
      y2 = 100;
      Graphics2D gr2D = (Graphics2D)gr;
      gr2D.setPaint(new GradientPaint(x, y, Color.LIGHT_GRAY,
         x2, y2, Color.DARK_GRAY, false));
      gr2D.fill(new Rectangle2D.Double(x, y, x2, y2));
      x = 210;
      gr2D.setPaint(new GradientPaint(x, y, Color.LIGHT_GRAY,
         x2, y2, Color.DARK_GRAY, true));
      gr2D.fill(new Rectangle2D.Double(x, y, x2, y2));
   }
   public static void main(String[] args)
   {
      JFrame frame = new JFrame();
      frame.add(new JGradient());
      frame.setSize(440, 200);
      frame.setVisible(true);
   }
}
```

Figure 16-36 The `JGradient` class

Setting a Drawing Stroke

- **Stroke**
  - Represents a single movement
  - **`setStroke()` method**
- `Stroke` interface
- **`BasicStroke`** class
- **Endcap styles**
  - Apply to the ends of lines that do not join with other lines
- **Juncture styles**
  - For lines that join

Setting a Drawing Stroke (cont'd.)

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
public class JStroke extends JPanel
{
   public void paintComponent(Graphics gr)
   {
      super.paintComponent(gr);
      Graphics2D gr2D = (Graphics2D)gr;
      BasicStroke aStroke = new BasicStroke(15.0f,
         BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND);
      gr2D.setStroke(aStroke);
      gr2D.draw(new Rectangle2D.Double(20, 20, 100, 100));
   }
   public static void main(String[] args)
   {
      JFrame frame = new JFrame();
      frame.add(new JStroke());
      frame.setSize(160, 180);
      frame.setVisible(true);
   }
}
```

Figure 16-38 The JStroke class

Setting a Drawing Stroke (cont'd.)

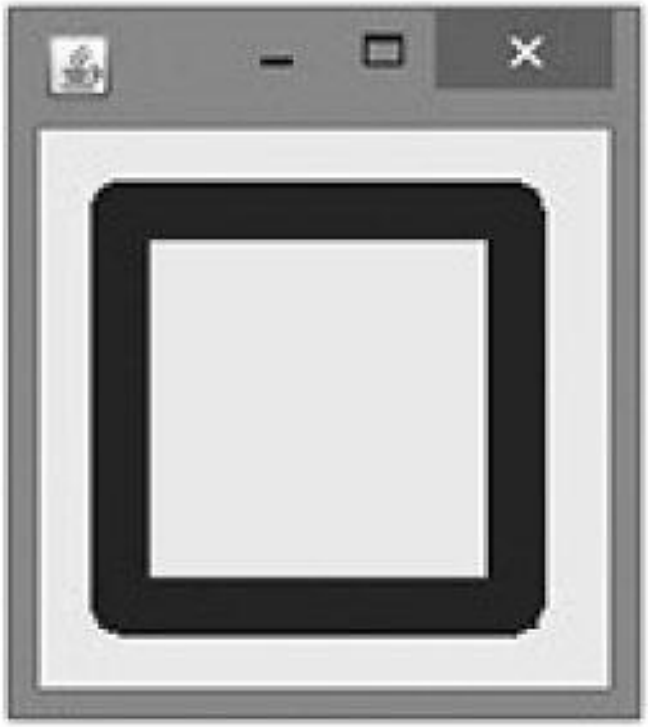

Figure 16-39 Output of the `JStroke` program

Creating Objects to Draw

- Objects drawn in Java 2D are created by defining them as geometric shapes
  - Use the `java.awt.geom` package classes
  - Define the shape
  - Use the shape as an argument to the `draw()` or `fill()` methods

Lines

- `Line2D.Float`
- `Line2D.Double`
- `Point2D.Float`
- `Point2D.Double`

Creating Objects to Draw (cont'd.)

Rectangles

- `Rectangle2D.Float`
- `Rectangle2D.Double`
- `Rectangle2D.Float rect = new Rectangle2D.Float(10F, 10F, 50F, 40F);`

Ovals

- `Ellipse2D.Float`
- `Ellipse2D.Double`
- `Ellipse2D.Float ell = new Ellipse2D.Float(10F, 73F, 40F, 20F);`

Creating Objects to Draw (cont'd.)

Arcs

- `Arc2D.Float`
- `Arc2D.Double`
- `Arc2D.PIE`
- `Arc2D.CHORD`
- `Arc2D.OPEN`
- `Arc2D.Float ac = new Arc2D.Float (10,133,30,33,30,120,Arc2D.PIE);`

Creating Objects to Draw (cont'd.)

Polygons

- Define movements from one point to another
- `GeneralPath` object
- `GeneralPath pol = new GeneralPath();`
- `moveTo()`
- `lineTo()`
- `closePath()`

You Do It

- Using the `drawString()` Method
- Creating a `JFrame` to Hold `JStringPanel` Objects
- Creating a `JPanel` with a `JButton` and `Graphics`
- Observing the Effect of the `super.paintComponent()` Call
- Copying an Area

You Do It (cont'd.)

- Using `FontMetrics` to Compare fonts
- Using Drawing Strokes
- Working with Shapes
- Creating an Interactive Program that Draws Lines

Don’t Do It

- Don’t forget to call `super.paintComponent()` as the first statement in the `paintComponent()` method
- Don’t forget that the lower-left corner of a `String` is placed at the coordinates used when you call `drawString()`
- Don’t forget that the name of the Graphics method that draws rectangles is `drawRect()` and not `drawRectangle()`

Summary

- `paint()` method
- `drawString()` method
  - Draws a `String` in a `JApplet` window
- Methods for drawing a variety of lines and geometric shapes
- `getAvailableFontFamilyNames()` method
  - Discovers fonts available on a system
- Java 2D
  - Higher quality