Java2D Graphics – Part 1
Introduction

- In Java 1.2, the paintComponent method is supplied with a Graphics2D object (a subclass of Graphics), which includes a more robust set of drawing operations, such as:
  - pen widths
  - dashed lines
  - image and gradient color fill patterns
  - the use of arbitrary local fonts
  - a floating point coordinate system
  - a number of coordinate transformation operations
Introduction

- To maintain compatibility with Swing as used in Java 1.1, the declared type of the `paintComponent` argument is `Graphics`, so you have to cast it to `Graphics2D` before using it.
Java 2D Graphics

- Introduction
  - **Java 1.1**: Example
    ```java
    public void paint (Graphics g)
    {
        // Set pen parameters
        g.setColor (someColor);
        g.setFont (someLimitedFont);

        // Draw a shape
        g.drawString (...);
        g.drawLine (...)
        g.drawRect (...);       // outline
        g.fillRect (...);      // solid
        g.drawPolygon (...);  // outline
        g.fillPolygon (...);  // solid
        g.drawOval (...);  // outline
        g.fillOval (...);   // solid
        ...
    }
    ```
Introduction

- **Java 1.2**: Example

```java
public void paintComponent (Graphics g) {
    // Clear off-screen bitmap
    super.paintComponent (g);
    // Cast Graphics to Graphics2D
    Graphics2D g2d = (Graphics2D) g;
    // Set pen parameters
    g2d.setPaint (fillColorOrPattern);
    g2d.setStroke (penThicknessOrPattern);
    g2d.setComposite (someAlphaComposite);
    g2d.setFont (anyFont);
}
```
• Introduction
  ◦ **Java 1.2**: Example

```java
    g2d.translate (...);
g2d.rotate (...);
g2d.scale (...);
g2d.shear (...);
g2d.setTransform (someAffineTransform);
// Allocate a shape
SomeShape s = new SomeShape (...);
// Draw shape
    g2d.draw (s); // outline
    g2d.fill (s); // solid
```
General Approach

1. Cast the Graphics object to a Graphics2D object

```java
public void paintComponent (Graphics g)
{
    super.paintComponent (g); // Typical Swing approach
    Graphics2D g2d = (Graphics2D) g;
    g2d.doSomeStuff (...);
    ...
}
```
• General Approach
  2. Create a Shape object

        Rectangle2D.Double rect = ...;
        Ellipse2D.Double ellipse = ...;
        Polygon poly = ...;
        GeneralPath path = ...;
        SomeShapeYouDefined shape = ...;  // Satisfies
                                       // Shape interface

        ...
• General Approach
  ◦ 3. Optional: modify drawing parameters

  g2d.setPaint (fillColorOrPattern);
g2d.setStroke (penThicknessOrPattern);
g2d.setComposite (someAlphaComposite);
g2d.setFont (someFont);
g2d.translate(...);
g2d.rotate(...);
g2d.scale(...);
g2d.shear(...);
g2d.setTransform (someAffineTransform);
• General Approach
  ◦ 4. Draw an outlined or solid version of the Shape

  g2d.draw (someShape);
g2d.fill (someShape);
• Drawing Shapes in Java2D
  ◦ Overview
    • With AWT/Swing, you generally draw a shape by calling the drawXxx () or fillXxx () method of the Graphics object

    • In Java2D, you generally first create a Shape object, then call either the draw () or fill () method of the Graphics2D object, supplying the Shape object as an argument
• Drawing Shapes in Java2D
  ◦ Overview
  • Example:

  ```java
  public void paintComponent (Graphics g)
  {
      super.paintComponent (g);
      Graphics2D g2d = (Graphics2D) g;

      // Assume x, y, and diameter are instance variables
      Ellipse2D.Double circle = new Ellipse2D.Double (x, y, diameter, diameter);
      g2d.fill (circle);
      ...
  }
  ```
Drawing Shapes in Java2D

- Overview
  - You can still call the drawXxx() methods if you like, however
  
  - This is necessary for drawString() and drawImage(), and possibly convenient for draw3DRect()

  - Several classes have similar versions that store coordinates as either double precision numbers (Xxx.Double()) or single precision numbers (Xxx.Float())

  - The reason for this is that single precision coordinates might be slightly faster to manipulate on some platforms.
• **Shape classes**
  ◦ Arguments to the Graphics2D draw() and fill() methods **must** implement the Shape interface
  ◦ You can create your own shapes, naturally, but following are the major built-in ones
  ◦ Except for Rectangle() and Polygon(), which are left over from Java 1.1, these appear in the java.awt.geom package
• Shape classes
  ◦ Arc2D.Double, Arc2D.Float
  ◦ Area (a shape built by adding/subtracting other shapes)
  ◦ CubicCurve2D.Double, CubicCurve2D.Float
  ◦ Ellipse2D.Double, Ellipse2D.Float
  ◦ GeneralPath (a series of connected shapes)
  ◦ Line2D.Double, Line2D.Float
  ◦ Polygon
  ◦ QuadCurve2D.Double, QuadCurve2D.Float
  ◦ Rectangle2D.Double, Rectangle2D.Float, Rectangle
  ◦ RoundRectangle2D.Double, RoundRectangle2D.Float
import javax.swing.*;  // For JPanel, etc.
import java.awt.*;  // For Graphics, etc.
import java.awt.geom.*; // For Ellipse2D, etc.

public class ShapeExample extends JPanel
{
    private Ellipse2D.Double circle = new Ellipse2D.Double (10, 10, 350, 350);
    private Rectangle2D.Double square = new Rectangle2D.Double (10, 10, 350, 350);

    public void paintComponent (Graphics g) 
    {
        clear (g);
        Graphics2D g2d = (Graphics2D) g;
        g2d.fill (circle);
        g2d.draw (square);
    }
• Drawing Shapes Example
  // super.paintComponent clears offscreen pixmap,
  // since we're using double buffering by default
  protected void clear (Graphics g)
  {
    super.paintComponent (g);
  }

  protected Ellipse2D.Double getCircle ()
  {
    return (circle);
  }

  public static void main (String [] args)
  {
    WindowUtilities.openInJFrame (new ShapeExample (), 380, 400);
  }
Drawing Shapes Example

`WindowUtilities.java`

```java
import javax.swing.*;
import java.awt.);

class WindowUtilities
{
    // Tell system to use native look and feel, as in previous releases. Metal (Java) LAF is
    // the default otherwise.
    public static void setNativeLookAndFeel()
    {
        try
        {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch (Exception e)
        {
            System.out.println("Error setting native LAF: "+ e);
        }
    }
}
```
Drawing Shapes Example

**WindowUtilities.java**

// A simplified way to see a JPanel or other Container. Pops up // a JFrame with specified Container as the content pane.

```java
public static JFrame openInJFrame (Container content, int width, int height, String title, Color bgColor)
{
    JFrame frame = new JFrame (title);
    frame.setBackground (bgColor);
    content.setBackground (bgColor);
    frame.setSize (width, height);
    frame.setContentPane (content);
    frame.addWindowListener (new ExitListener ());
    frame.setVisible (true);
    return (frame);
}
```
### Drawing Shapes Example

*WindowUtilities.java*

```java
public static JFrame openInJFrame (Container content, int width, int height, String title)
{
    return (openInJFrame (content, width, height, title, Color.white));
}

// Uses Color.white as the background color, and the name of
// the Container's class as the JFrame title.
public static JFrame openInJFrame (Container content, int width, int height)
{
    return (openInJFrame (content, width, height, content.getClass ().getName (), Color.white));
}
```

*Java2D Graphics*
• Drawing Shapes Example

ExitListener.java

```java
import java.awt.*;
import java.awt.event.*;

public class ExitListener extends WindowAdapter
{
    public void windowClosing (WindowEvent event)
    {
        System.exit (0);
    }
}
```

Java2D Graphics
• Drawing Shapes Example
  Result

Java2D Graphics
• Questions/comments/know a guy?