2D Graphics Programming in Java

Java2D Library – Rendering and Visual Effects

Zbigniew Dendzik

March 2026

Lecture Plan

- Introduction to Java2D library
- Basic geometric shapes
- paintComponent() method
- Colors and filling
- Gradients – linear and radial
- Textures
- Transparency – AlphaComposite
- Geometric transformations
- Line styles – BasicStroke

Java2D Library

Java2D – library for 2D graphics programming. **Part of Java SE platform:**

- Package `java.awt` – basic graphic classes
- Package `java.awt.geom` – geometric shapes
- Package `java.awt.image` – image processing

Capabilities:

- Rendering geometric shapes
- Colors, gradients, textures
- Transparency and color blending
- Geometric transformations
- Custom line styles

Graphics Application Architecture

Basic components:

1. **JFrame** – main application window
2. **JPanel** – drawing panel (custom class)
3. **Graphics2D** – graphics context
4. **Shape** – geometric shapes

Pattern:

JFrame $\rightarrow$ custom JPanel $\rightarrow$ paintComponent() $\rightarrow$ Graphics2D $\rightarrow$ draw(Shape)

Canvas Class – Extending JPanel

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

class Canvas extends JPanel
{
    Shape shape;

    Canvas(Shape shape)
    {
        this.shape = shape;
    }

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

Main Class – Program

```
public class Program
{
    public static void main(String[] args)
    {
        Shape obj1;
        obj1 = new Rectangle2D.Float(100, 100, 140, 100);

        Canvas p = new Canvas(obj1);

        JFrame jf = new JFrame();
        jf.add(p);

        jf.setTitle("Graphics Test");
        jf.setSize(400, 400);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jf.setVisible(true);
    }
}
```

paintComponent() Method

paintComponent(Graphics g) – component rendering method. **Key elements:**

- Override method from JPanel class
- Call `super.paintComponent(g)` – clears background
- Cast Graphics to Graphics2D
- Call drawing methods

Graphics2D methods:

- `draw(Shape)` – draws shape outline
- `fill(Shape)` – fills shape
- `drawString(String, float, float)` – draws text

Shape Interface

Shape – interface representing geometric shape. **Implementations in java.awt.geom package:**

- **Rectangle2D.Float** – rectangle
- **Ellipse2D.Float** – ellipse
- **Line2D.Float** – line
- **Arc2D.Float** – arc
- **QuadCurve2D.Float** – quadratic curve
- **CubicCurve2D.Float** – cubic curve
- **Polygon** – polygon

Rectangle2D.Float – Rectangle

```
// Creating rectangle
Shape rect = new Rectangle2D.Float(x, y, width, height);

// Example
Shape rect = new Rectangle2D.Float(100, 100, 140, 100);
// x=100, y=100, width=140, height=100
```

Parameters:

- `x, y` – coordinates of upper-left corner
- `width` – width
- `height` – height

Ellipse2D.Float – Ellipse

```
// Creating ellipse
Shape ellipse = new Ellipse2D.Float(x, y, width, height);

// Example
Shape ellipse = new Ellipse2D.Float(50, 50, 200, 150);
// Ellipse inscribed in 200x150 rectangle
```

Note:

- Ellipse inscribed in rectangle
- Parameters like rectangle
- Circle: `width == height`

Point2D.Float – Point

```
// Creating point
Point2D.Float point = new Point2D.Float(x, y);

// Example
Point2D.Float p1 = new Point2D.Float(150, 200);
```

Applications:

- Testing point containment in shapes
- Defining gradients
- Curve control points

Testing Shape Relationships

Shape class methods:

- `contains(double x, double y)` – is point inside shape?
- `contains(Point2D p)` – is point inside shape?
- `intersects(Rectangle2D r)` – do shapes intersect?

Use cases:

- Collision detection
- User interaction (clicks)
- Geometric tests

Testing Shape Intersection

```
Rectangle2D.Float rect = new Rectangle2D.Float(50, 50, 100, 80)
    ;
Ellipse2D.Float ellipse = new Ellipse2D.Float(100, 70, 120,
    100);

if(rect.intersects(ellipse.getBounds2D()))
{
    g2d.drawString("Shapes intersect!", 10, 20);
}
else
{
    g2d.drawString("Shapes do not intersect", 10, 20);
}
```

Testing Point Containment

```
Ellipse2D.Float ellipse = new Ellipse2D.Float(50, 50, 200, 150)
    ;
Point2D.Float point = new Point2D.Float(150, 100);

if(ellipse.contains(point))
{
    g2d.drawString("Point is inside", 10, 20);
}
else
{
    g2d.drawString("Point is outside", 10, 20);
}
```

Colors and Filling

Color class – RGB color representation. **Creating color:**

- `new Color(int r, int g, int b)` – RGB 0-255
- `new Color(int r, int g, int b, int a)` – RGBA with transparency
- Constants: `Color.RED`, `Color.GREEN`, `Color.BLUE`

Setting color:

- `g2d.setColor(Color)` – sets drawing color
- `g2d.setPaint(Paint)` – sets fill pattern

Filling with Color

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

    // Green filled rectangle
    Rectangle2D.Float rect = new Rectangle2D.Float(50, 50, 200,
        150);

    g2d.setColor(Color.GREEN);
    g2d.fill(rect);  // Fill

    g2d.setColor(Color.BLACK);
    g2d.draw(rect);  // Outline
}
```

Gradients – GradientPaint

GradientPaint – linear gradient between two points. **Constructor:**

- `GradientPaint(float x1, float y1, Color c1, float x2, float y2, Color c2)`
- Point 1: `(x1, y1)` with color `c1`
- Point 2: `(x2, y2)` with color `c2`
- Linear interpolation between points

Usage:

- `g2d.setPaint(gradientPaint)`
- `g2d.fill(shape)`

Linear Gradient – Example

```
Ellipse2D.Float ellipse = new Ellipse2D.Float(50, 50, 300, 200)
    ;

// Gradient from green to white
GradientPaint gp = new GradientPaint(
    50, 125, Color.GREEN,
    350, 125, Color.WHITE
);

g2d.setPaint(gp);
g2d.fill(ellipse);

g2d.setColor(Color.BLACK);
g2d.draw(ellipse);
```

Radial Gradients – RadialGradientPaint

RadialGradientPaint – radial gradient from center. **Constructor:**

- `RadialGradientPaint(Point2D center, float radius, float[] fractions, Color[] colors)`
- `center` – center point
- `radius` – gradient radius
- `fractions` – color positions (0.0-1.0)
- `colors` – colors at those positions

Radial Gradient – Example

```
Ellipse2D.Float ellipse = new Ellipse2D.Float(50, 50, 300, 200)
    ;

Point2D center = new Point2D.Float(200, 150);
float radius = 150f;
float[] fractions = {0.0f, 1.0f};
Color[] colors = {Color.WHITE, Color.GREEN};

RadialGradientPaint rgp = new RadialGradientPaint(
    center, radius, fractions, colors
);

g2d.setPaint(rgp);
g2d.fill(ellipse);
```

Textures – TexturePaint

TexturePaint – texture fill from image. **Required elements:**

- **BufferedImage** – texture image
- **Rectangle2D** – repeat area

Constructor:

- `TexturePaint(BufferedImage img, Rectangle2D anchor)`
- Texture tiled across entire drawing area

Loading Image

```
import java.io.*;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

BufferedImage img = null;

try
{
    File f = new File("texture.jpg");
    img = ImageIO.read(f);
}
catch(IOException e)
{
    System.err.println("File problem: " + e);
}
```

TexturePaint – Example

```
// Load image (as above)
BufferedImage img = ImageIO.read(new File("texture.jpg"));

// Create texture
Rectangle2D anchor = new Rectangle2D.Float(0, 0,
    img.getWidth(), img.getHeight());
TexturePaint tp = new TexturePaint(img, anchor);

// Apply
Rectangle2D.Float rect = new Rectangle2D.Float(50, 50, 300,
   200);
g2d.setPaint(tp);
g2d.fill(rect);
```

Transparency – AlphaComposite

AlphaComposite – transparency and blending control. **Creation:**

- `AlphaComposite.getInstance(int rule, float alpha)`
- `rule` – blending rule (e.g., `SRC_OVER`)
- `alpha` – transparency coefficient 0.0-1.0

Blending rules:

- `SRC_OVER` – standard overlay
- `SRC` – replacement
- `DST_OVER` – draw under existing

AlphaComposite – Example

```
Ellipse2D.Float e1 = new Ellipse2D.Float(50, 50, 200, 200);
Ellipse2D.Float e2 = new Ellipse2D.Float(150, 100, 200, 200);

// First ellipse - opaque
g2d.setColor(Color.BLUE);
g2d.fill(e1);

// Set 50% transparency
AlphaComposite ac = AlphaComposite.getInstance(
    AlphaComposite.SRC_OVER, 0.5f
);
g2d.setComposite(ac);

// Second ellipse - transparent
g2d.setColor(Color.RED);
g2d.fill(e2);
```

Arc2D – Arcs

Arc2D.Float – arc or circle segment. **Constructor:**

- `Arc2D.Float(float x, float y, float w, float h, float start, float extent, int type)`
- `x, y, w, h` – rectangle describing ellipse
- `start` – start angle (degrees)
- `extent` – arc extent (degrees)
- `type` – type: `OPEN`, `CHORD`, `PIE`

Arc2D – Example

```
// Arc from 0 to 90 degrees (quarter circle)
Arc2D.Float arc = new Arc2D.Float(
    50, 50,         // x, y
    200, 200,       // width, height
    0,              // start angle
    90,             // extent
    Arc2D.OPEN      // type
);

g2d.setColor(Color.BLUE);
g2d.draw(arc);
```

BasicStroke – Line Styles

BasicStroke – defines line drawing style. **Parameters:**

- **width** – line thickness
- **cap** – line endings
- **join** – line joining method
- **dash** – dashed line pattern

Cap types:

- `CAP_BUTT` – flat
- `CAP_ROUND` – rounded
- `CAP_SQUARE` – square

BasicStroke – Example

```
Line2D.Float line1 = new Line2D.Float(50, 50, 300, 50);
Line2D.Float line2 = new Line2D.Float(50, 100, 300, 100);
Line2D.Float line3 = new Line2D.Float(50, 150, 300, 150);

// Different line caps
BasicStroke s1 = new BasicStroke(20, BasicStroke.CAP_BUTT,
                                 BasicStroke.JOIN_MITER);
BasicStroke s2 = new BasicStroke(20, BasicStroke.CAP_ROUND,
                                 BasicStroke.JOIN_MITER);
BasicStroke s3 = new BasicStroke(20, BasicStroke.CAP_SQUARE,
                                 BasicStroke.JOIN_MITER);

g2d.setStroke(s1); g2d.draw(line1);
g2d.setStroke(s2); g2d.draw(line2);
g2d.setStroke(s3); g2d.draw(line3);
```

Geometric Transformations

Graphics2D – coordinate system transformation methods. **Basic transformations:**

- `translate(double tx, double ty)` – translation
- `rotate(double theta)` – rotation around (0,0)
- `rotate(double theta, double x, double y)` – rotation around point
- `scale(double sx, double sy)` – scaling

Note:

- Transformations are cumulative
- Order matters
- Use `AffineTransform` for complex operations

Transformations – Example

```
Rectangle2D.Float rect = new Rectangle2D.Float(0, 0, 100, 60);

// Translation
g2d.translate(150, 150);

// Rotation 45 degrees
g2d.rotate(Math.toRadians(45));

// Scaling 1.5x
g2d.scale(1.5, 1.5);

// Drawing
g2d.setColor(Color.BLUE);
g2d.fill(rect);
```

Polygon – Polygons

Polygon – polygon defined by points. **Creation:**

- `Polygon()` – empty polygon
- `addPoint(int x, int y)` – add vertex
- `Polygon(int[] xpoints, int[] ypoints, int npoints)` – from arrays

Properties:

- Automatic contour closing
- Can be filled and outlined
- Used for complex shapes

Polygon – Example

```
// Triangle
int[] xpoints = {150, 250, 200};
int[] ypoints = {50, 50, 150};
Polygon triangle = new Polygon(xpoints, ypoints, 3);

g2d.setColor(Color.CYAN);
g2d.fill(triangle);

g2d.setColor(Color.BLACK);
g2d.draw(triangle);
```

Antialiasing

Antialiasing – edge smoothing. **Rendering hints:**

- Rendering quality control
- `RenderingHints.KEY_ANTIALIASING`
- `RenderingHints.KEY_TEXT_ANTIALIASING`

Enabling antialiasing:

- Better visual quality
- Slower rendering
- Recommended for presentations

Antialiasing – Example

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

    // Enable antialiasing
    g2d.setRenderingHint(
        RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON
    );

    g2d.setRenderingHint(
        RenderingHints.KEY_TEXT_ANTIALIASING,
        RenderingHints.VALUE_TEXT_ANTIALIAS_ON
    );

    // Drawing with smooth edges...
}
```

Practical Applications

Graphing calculator:

- Standard calculator functionality
- Drawing function graphs
- Mathematical expression parsing
- Coordinate system transformations

Other applications:

- Graphics editors
- 2D games
- Data visualization
- Diagrams and schematics
- Animations

Summary

- **Java2D** – 2D graphics library
- **Shape** – geometric shapes
- **Graphics2D** – graphics context
- **paintComponent()** – rendering
- **Colors and gradients** – filling
- **Textures** – TexturePaint
- **Transparency** – AlphaComposite
- **BasicStroke** – line styles
- **Transformations** – translate, rotate, scale