JAVA 2D

What is Java 2D ?
Who needs Java 2D ?
What features does Java 2D add to the Java platform ?

Jason Hoang
April 28, 2005

Overview

- Sun Microsystems has developed Java language with Object Oriented language .It has a very large class library and hierarchy which helps programmers to simplify their codes such as simplicity and portability by using readily made classes.
- The Java 2D API introduced in JDK 1.2 provides enhanced two–dimensional graphics, text, and image capabilities for Java programs through extensions to the Abstract Windowing Toolkit (AWT).

What is Java 2D ?

- It is a set of classes that extent the capabilities of Sun 's Abstract Windowing Toolkit.
- It handles arbitrary shapes, text, and images and provides a uniform mechanism for performing transformations, such as rotation and scaling.
- It supports a wide array of presentation devices such as displays and printers, as well as image formats and encoding, color spaces, and rendering techniques and effects.
- It also offers comprehensive text and font handling, as well as color support.

Who needs Java 2D ?

- The Java 2D API provides many benefits to developers who want to incorporate graphics, text, and images into their applications and applets.
- In other words, Java 2D benefits virtually all Java developers.
- By enabling the incorporation of sophisticated graphics and text, the Java 2D makes it possible for developers to create Java programs that provide a richer end-user experience.

Features:

Drawing Shapes in Java 2D.

- A shape in Java 2D is an infinitely thin boundary which defines an inside and an outside. Pixels inside the shape are affected by the drawing operation, those outside are not.
- In Java 2D, we generally create a Shape object, then call either the drawXxx or fillXxx method of the Graphics2D object, supplying the shape object as an argument.

Features (cont)

ShapeExample.

```
public void paintComponent (Graphics g){
   super.paintComponet (g);
   Graphics2D g2d = (Graphics2D) g;
// Assume x, y and diameter are instance variables
   Ellipse2D.Double circle =
   newEllipse2D.double(x, y, diameter, diameter);
   g2d.fill(circle);
…}
```

Features (cont)

Paint Styles in Java 2D

- A paint generates the colors to be used for each pixel of the fill operation.
- When we fill a shape, the current Paint attribute of Graphics2D object is used. This can be a Color (solid color), a gradientPaint (gradient fill gradually combining two colors), a texturePaint (tiled image), or a new version of paint that we write ourselves.
- Use setPaint and getPaint to change and retrieve the Paint settings.

Features (cont)

PaintExample.

```
public void paintComponet (Graphics g){
    clear (g);
    Graphics2D g2d = (Graphics 2D) g;
    drawGradientCircle (g2d); }
protected void drawGradientCircle(Graphics2D g2d) {
   g2d.setPaint(gradient);
   g2d.fill(getCircle());
   g2d.setPaint(Color.black);
   g2d.draw(getCircle());
…}
```

Features (cont)

Transparency in Java 2D.

- Setting transparency by creating an AlphaComposite object then passing it to the setComposite method of the Graphics2D object.
- Create an AlphaComposite by calling AlphaComposite.getInstance with a mixing rule designator and a transparency value.

Features (cont)

TransparencyExample.

```
private AlphaComposite makeComposite(float alpha) {
   int type = AlphaComposite.SRC_OVER;
   return(AlphaComposite.getInstance(type, alpha));  }
private void drawSquares(Graphics2D g2d, float alpha) {
   Composite originalComposite = g2d.getComposite();
   g2d.setPaint(Color.blue);
   g2d.fill(blueSquare);
   g2d.setComposite(makeComposite(alpha));
   g2d.setPaint(Color.red);
   g2d.fill(redSquare);
   g2d.setComposite(originalComposite);  …}
```

Features (cont)

Fonts in Java 2D

- A fonts can be thought of as collection of glyphs (the shapes that font uses to represent the characters in a string).
- A single font might have many faces, such as heavy, medium, oblique, and regular.
- Using an instance of Font by calling the static method GraphicsEnviroment.getLocalGraphicsEnviroment and then querying the returned GraphicEnviroment.

Features (cont)

FontExample.

```
public class FontExample extends GradientPaintExample {
  public FontExample() {
    GraphicsEnvironment env =
      GraphicsEnvironment.getLocalGraphicsEnvironment();
    env.getAvailableFontFamilyNames();
    setFont(new Font("Goudy Handtooled BT", Font.PLAIN,
    100)); }
protected void drawBigString(Graphics2D g2d) {
    g2d.setPaint(Color.black);
    g2d.drawString("Java 2D", 25, 215);  …}
```

Features (cont).

Stroke Styles in Java2D

- In addition to the fill operation, Java 2D provides a draw operation, while fill draws the interior of a shape, draw draws its outline.
- The outline can be as simple as a thin line, or as complicated as a dashed line with each dash having rounded edges.
- Java 2D permits us to create a BasicStroke object and tell the Graphics2D object to use it via the setStroke method.

Features (cont).

StrokeExample.

```
public class StrokeThicknessExample extends
    FontExample {
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawBigString(g2d);
    drawThickCircleOutline(g2d); }
  protected void drawThickCircleOutline(Graphics2D g2d) {
    g2d.setPaint(Color.blue);
    g2d.setStroke(new BasicStroke(8)); // 8-pixel wide
    g2d.draw(getCircle()); … }
```

Features (cont).

Coordinate transformation in Java 2D.

- Every Java2D operation is subject to a transform, so that shapes may be translated, rotated, shear, and scaled as they are drawn.
- Two basic ways to use transformations: create an AffineTransform object, set its parameters, and the assign that object to the Graphics2D object via setTransform. Alternatively, call translate, rotate, scale, and shear directly on the Graphics2D object.

Features (cont).

TransformExample.

```
public void paintComponent(Graphics g) {
   clear(g);
   Graphics2D g2d = (Graphics2D)g;
   drawGradientCircle(g2d);
   drawThickCircleOutline(g2d);
   // Move the origin to the center of the circle.
   g2d.translate(185.0, 185.0);
   for (int i=0; i<16; i++) {
     g2d.rotate(Math.PI/8.0);
     g2d.setPaint(colors[i%2]);
     g2d.drawString("Java", 0, 0);
   }
```

Summary

Create Shape objects, then call Graphics2D's draw and fill methods with shapes as args.

- Use setPaint to specify a solid color (Color), a gradient fill (GradientPaint), or tiled image (TexturePaint). TexturePaint requires a BufferedImage, which you can create from an image file by creating empty BufferedImage then drawing image into it.
- Use AlphaComposite for transparency. Create one via AlphaComposite.getInstance with a type of AlphaComposite.SRC_OVER.

Summary (cont)

- Local fonts: before using them you must call GraphicsEnvironment.getLocalGraphicsEnvironment().getAllFonts() or GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamilyNames(). Then supply name to Font constructor and specify font via setFont.
- BasicStroke lets you set pen thickness, dashing pattern, and line cap/join styles. Supply it to Graphics2D via setStroke.
- Coordinate transforms let you move the coordinate system rather than changing what you draw. Do simple transforms via the translate, rotate, scale, and shear methods of Graphics2D. Do more complex transforms by supplying transformation matrix to AffineTransform constructor, then calling Graphics2D's setTransform method.

Conclusions

- Java 2D does a lot of calculations compare to the old AWT.
- Java 2D performs the minimum amount of work necessary to make it seems as if it is performing all of these steps for each operation, therefore retaining both great flexibility and high performance.