core

WEB

programming

Java 2D

Drawing in Java 2

Agenda

- **Overview**
- **Drawing Shapes**
- **Paint Styles**
- **Transparency**
- **Using Local Fonts**
- **Stroke Styles**
- **Coordinate Transformations**
- **Requesting Drawing Accuracy**

www.corewebprogramming.com
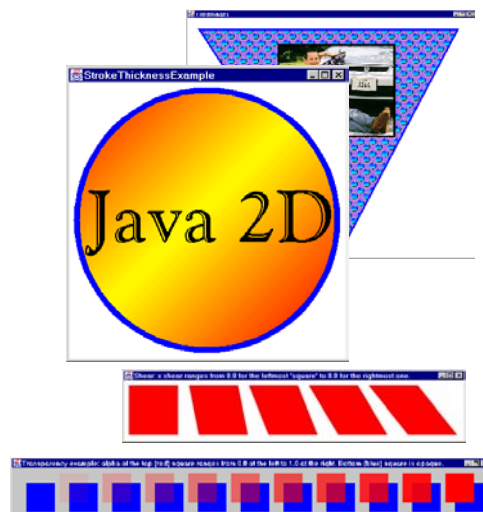
Java 1.1 vs Java 2 Drawing: Overview

Java 1.1

```
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
  …
}
```

Java 2

```
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);
  g2d.translate(…);
  g2d.rotate(…);
  g2d.scale(…);
  g2d.shear(…);
  g2d.setTransform(someAffineTransform);
  // Create a Shape object
  SomeShape s = new SomeShape(…);
  // Draw shape
  g2d.draw(s);  // outline
  g2d.fill(s);  // solid
}
```

www.corewebprogramming.com

Java 2D Drawing Process: Step 1

- **Cast Graphics object to Graphics2D**

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);  // Typical Swing
    Graphics2D g2d = (Graphics2D)g;
    g2d.doSomeStuff(...);
    ...
}
```

- **Note**
  – All methods that return Graphics in Java 1.1 return Graphics2D in Java 2
    - paint, paintComponent
    - getGraphics

www.corewebprogramming.com

Java 2D Drawing Process: Step 2

- **Set pen parameters**
  - g2d.setPaint(fillColorOrPattern);
  - g2d.setStroke(penThicknessOrPattern);
  - g2d.setComposite(someAlphaComposite);
  - g2d.setFont(someFont);
  - g2d.translate(...);
  - g2d.rotate(...);
  - g2d.scale(...);
  - g2d.shear(...);
  - g2d.setTransform(someAffineTransform);

www.corewebprogramming.com

Java 2D Drawing Process: Step 3

- **Create a Shape object.**
  Rectangle2D.Double rect = ...;
  Ellipse2D.Double ellipse = ...;
  Polygon poly = ...;
  GeneralPath path = ...;
  // Satisfies Shape interface
  SomeShapeYouDefined shape = ...;

- **Note**
  - Most shapes are in the java.awt.geom package
  - There is a corresponding Shape class for most of the draw*Xxx* methods of Graphics (see next slide)

www.corewebprogramming.com

Built-in Shape Classes

- **Arc2D.Double, Arc2D.Float**
- **Area (a shape built by union, intersection, subtraction and xor of other shapes)**
- **CubicCurve2D.Double, CubicCurve2D.Float**
- **Ellipse2D.Double, Ellipse2D.Float**
- **GeneralPath (a series of connected shapes), Polygon**
- **Line2D.Double, Line2D.Float**
- **QuadCurve2D.Double, QuadCurve2D.Float (a spline curve)**
- **Rectangle2D.Double, Rectangle2D.Float, Rectangle**
- **RoundRectangle2D.Double, RoundRectangle2D.Float**
  – New shapes are in java.awt.geom. Java 1.1 holdovers (Rectangle, Polygon) are in java.awt. Several classes have similar versions that store coordinates as either double precision numbers (*Xxx*.Double) or single precision numbers (*Xxx*.Float). The idea is that single precision coordinates might be slightly faster to manipulate on some platforms.

Java 2D Drawing Process: Step 4

- **Draw an outlined or filled version of the Shape**
  – g2d.draw(someShape);
  – g2d.fill(someShape);

- **The legacy methods are still supported**
  – drawString still commonly used
  – drawLine, drawRect, fillRect still somewhat used

# Drawing Shapes: Example Code

```
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);  // ie super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    g2d.fill(circle);
    g2d.draw(square);
  }
    // Code to put JPanel in JFrame omitted.
}
```

Drawing Shapes: Example Output
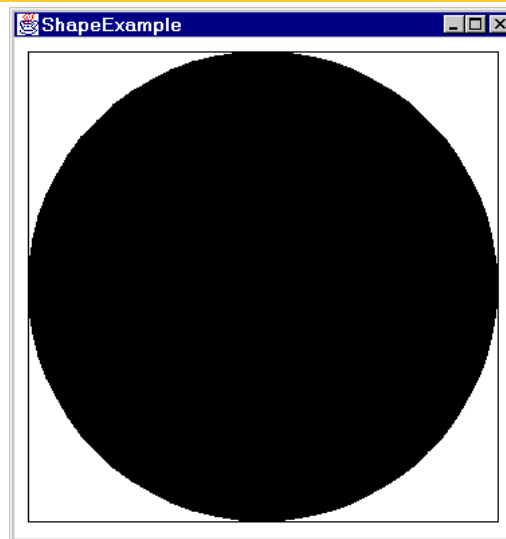
Paint Styles in Java 2D: Overview

- **Use setPaint and getPaint to change and retrieve the Paint settings.**
  - Note that setPaint and getPaint supersede the setColor and getColor methods that were used in Graphics (and inherited in Graphics2D).
- **When you fill a Shape, the current Paint attribute of the Graphics2D object is used. Possible arguments to setPaint are:**
  - A Color (solid color--Color implements Paint interface)
  - A GradientPaint (gradually-changing color combination)
  - A TexturePaint (tiled image)
  - A new version of Paint that you write yourself.

www.corewebprogramming.com

Paint Classes: Details

- **Color**
  - Has the same constants (Color.red, Color.yellow, etc.) as the AWT version, plus some extra constructors.
- **GradientPaint**
  - Constructors take two points, two colors, and optionally a boolean flag that indicates that the color pattern should cycle. Colors fade from one color to the other.
- **TexturePaint**
  - Constructor takes a BufferedImage and a Rectangle2D, maps the image to the rectangle, then tiles the rectangle.
    - Creating a BufferedImage from a GIF or JPEG file is tedious. First load an Image normally, get its size, create a BufferedImage that size with BufferedImage.TYPE_INT_ARGB as the image type, and get the BufferedImage's Graphics object via createGraphics. Then, draw the Image into the BufferedImage using drawImage.

www.corewebprogramming.com

Gradient Fills: Example Code

```
public class GradientPaintExample extends ShapeExample {
  private GradientPaint gradient =
    new GradientPaint(0, 0, Color.red, 175, 175,
                      Color.yellow, true);
                                // true means repeat pattern

  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
  }

  protected void drawGradientCircle(Graphics2D g2d) {
    g2d.setPaint(gradient);
    g2d.fill(getCircle());
    g2d.setPaint(Color.black);
    g2d.draw(getCircle());
  }
```
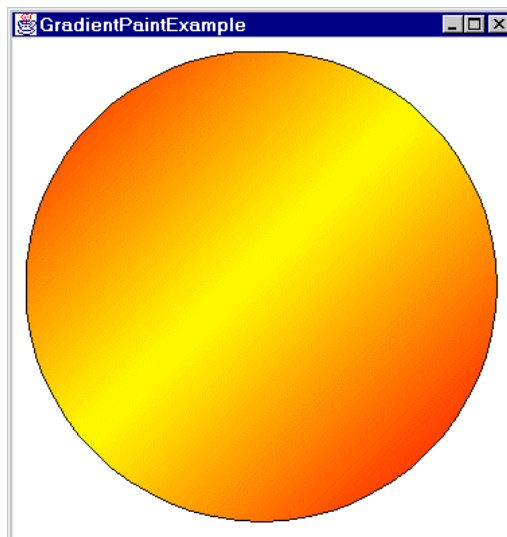
www.corewebprogramming.com

Gradient Fills: Example Output

www.corewebprogramming.com

Tiled Images as Fill Patterns (TexturePaint): Overview

- **Create a TexturePaint object. TexturePaint constructor takes:**
  - A BufferedImage (see following pages)
    - Specifies what to draw
  - A Rectangle2D
    - Specifies where tiling starts
- **Use the setPaint method of Graphics2D to specify that this TexturePaint object be used.**
  - Applies to strings and outlines (i.e., draw operations), not just solid shapes (i.e., fill operations).

Creating a BufferedImage for Custom Drawing

- **Call the BufferedImage constructor with**
  - A width,
  - A height, and
  - A value of BufferedImage.TYPE_INT_RGB,
- **Call createGraphics on the result to get a Graphics2D that refers to image**
  - Use that Graphics2D object to draw onto the BufferedImage

Custom BufferedImage: Example Code

```
int width = 32;
int height = 32;
BufferedImage bufferedImage =
  new BufferedImage(width, height
                     BufferedImage.TYPE_INT_RGB);
Graphics2D g2dImg = bufferedImage.createGraphics();
g2dImg.draw(...); // Draws onto image
g2dImg.fill(...); // Draws onto image
TexturePaint texture =
  new TexturePaint(bufferedImage,
                     new Rectangle(0, 0, width, height));
g2d.setPaint(texture);
g2d.draw(...); // Draws onto window
g2d.fill(...); // Draws onto window
```

Creating a BufferedImage from an Image File

- **Quick summary**
  - Load an Image from an image file via getImage
  - Use MediaTracker to be sure it is done loading
  - Create an empty BufferedImage using the Image width and height
  - Get the Graphics2D via createGraphics
  - Draw the Image onto the BufferedImage
- **This process has been wrapped up in the getBufferedImage method of the ImageUtilities class**
  - Like all examples, code available at www.corewebprogramming.com

BufferedImage from Image File: Example Code

```
public class ImageUtilities {
  public static BufferedImage getBufferedImage
                                    (String imageFile,
                                     Component c) {
    Image image = c.getToolkit().getImage(imageFile);
    waitForImage(image, c);  // Just uses MediaTracker
    BufferedImage bufferedImage =
      new BufferedImage(image.getWidth(c),
                        image.getHeight(c),
                        BufferedImage.TYPE_INT_RGB);
    Graphics2D g2dImg = bufferedImage.createGraphics();
    g2dImg.drawImage(image, 0, 0, c);
    return(bufferedImage);
  }
...
}
```

Tiled Images as Fill Patterns: Example Code

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

public class TiledImages extends JPanel {
  private String dir = System.getProperty("user.dir");
  private String imageFile1 = dir + "/images/marty.jpg";
  private TexturePaint imagePaint1;
  private Rectangle imageRect;
  private String imageFile2 = dir +
                              "/images/bluedrop.gif";
  private TexturePaint imagePaint2;
  private int[] xPoints = { 30, 700, 400 };
  private int[] yPoints = { 30, 30, 600 };
  private Polygon imageTriangle =
    new Polygon(xPoints, yPoints, 3);
```

Tiled Images as Fill Patterns: Example Code (Continued)

```
public TiledImages() {
  BufferedImage image =
    ImageUtilities.getBufferedImage(imageFile1, this);
  imageRect =
    new Rectangle(235, 70,
                  image.getWidth(), image.getHeight());
  imagePaint1 =
    new TexturePaint(image, imageRect);
  image =
    ImageUtilities.getBufferedImage(imageFile2, this);
  imagePaint2 =
    new TexturePaint(image,
                     new Rectangle(0, 0, 32, 32));
}
```

www.corewebprogramming.com

Tiled Images as Fill Patterns: Example Code (Continued)

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2d = (Graphics2D)g;
  g2d.setPaint(imagePaint2);
  g2d.fill(imageTriangle);
  g2d.setPaint(Color.blue);
  g2d.setStroke(new BasicStroke(5));
  g2d.draw(imageTriangle);
  g2d.setPaint(imagePaint1);
  g2d.fill(imageRect);
  g2d.setPaint(Color.black);
  g2d.draw(imageRect);
}
...
}
```

www.corewebprogramming.com

Tiled Images as Fill Patterns: Example Output

Transparent Drawing: Overview

- **Idea**
  - Assign transparency (alpha) values to drawing operations so that the underlying graphics partially shows through when you draw shapes or images.
- **Normal steps**
  - Create an AlphaComposite object
    - Call AlphaComposite.getInstance with a mixing rule designator and a transparency (or "alpha") value.
    - There are 8 built-in mixing rules (see the AlphaComposite API for details), but you only care about AlphaComposite.SRC_OVER.
    - Alpha values range from 0.0F (completely transparent) to 1.0F (completely opaque).
  - Pass the AlphaComposite object to the setComposite method of the Graphics2D

Transparent Drawing: Example Code

```java
public class TransparencyExample extends JPanel {
  ...
  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);
  }
...
```

www.corewebprogramming.com

Transparent Drawing: Example Code (Continued)

```java
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2d = (Graphics2D)g;
  for(int i=0; i<11; i++) {
    drawSquares(g2d, i*0.1F); // 2nd arg is transparency
    g2d.translate(deltaX, 0);
  }
}
```


Transparency example: alpha of the top (red) square ranges from 0.0 at the left to 1.0 at the right. Bottom (blue) square is opaque.
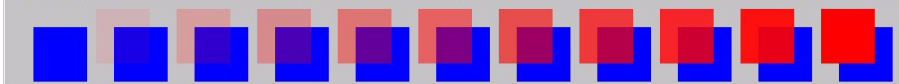
www.corewebprogramming.com

Using Logical (Java-Standard) Fonts

- **Logical font names: use same names as in Java 1.1.**
  - Serif (aka TimesRoman)
  - SansSerif (aka Helvetica -- results in Arial on Windows)
  - Monospaced (aka Courier)
  - Dialog
  - DialogInput.

Using Local (System-Specific) Fonts

- **Local fonts: <u>Must</u> Lookup Fonts First**
  - Use the getAvailableFontFamilyNames or getAllFonts methods of GraphicsEnvironment. E.g.:
    ```
    GraphicsEnvironment env =
      GraphicsEnvironment.getLocalGraphicsEnvironment();
    ```
    Then
    ```
    env.getAvailableFontFamilyNames();
    ```
    or
    ```
    env.getAllFonts();
    // Much slower than just getting names!
    ```
- **Safest Option:**
  - Supply list of preferred font names in order, loop down looking for first match. Supply standard font name as backup.

Example 1: Printing Out All Local Font Names

```
import java.awt.*;

public class ListFonts {
  public static void main(String[] args) {
    GraphicsEnvironment env =
      GraphicsEnvironment.getLocalGraphicsEnvironment();
    String[] fontNames =
      env.getAvailableFontFamilyNames();
    System.out.println("Available Fonts:");
    for(int i=0; i<fontNames.length; i++)
      System.out.println("  " + fontNames[i]);
  }
}
```

Example 2: Drawing with Local Fonts

```
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);
  }

  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawBigString(g2d);
  } ...
```

Drawing with Local Fonts: Example Output
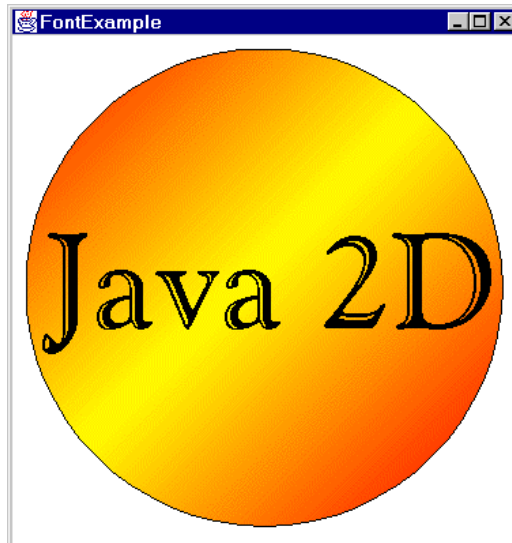
Stroke Styles: Overview

- **AWT**
  - draw*Xxx* methods of Graphics resulted in solid, 1-pixel wide lines.
  - Predefined line join/cap styles for drawRect, drawPolygon, etc.
- **Java2D**
  - Pen thickness
  - Dashing pattern
  - Line join/cap styles
- **Setting styles**
  - Create a BasicStroke object
  - Use the setStroke method to tell the Graphics2D object to use it

Stroke Attributes

- **Normal use: Use setStroke to assign a BasicStroke. BasicStroke constructors:**
  - BasicStroke()
    - Creates a BasicStroke with a pen width of 1.0, the default cap style of CAP_SQUARE, and the default join style of JOIN_MITER.
  - BasicStroke(float penWidth)
    - Uses the specified pen width and the default cap/join styles.
  - BasicStroke(float penWidth, int capStyle, int joinStyle)
    - Uses the specified pen width, cap style, and join style.
  - BasicStroke(float penWidth, int capStyle, int joinStyle, float miterLimit)
    - Limits how far up the miter join can go (default is 10.0). Stay away from this.
  - BasicStroke(float penWidth, int capStyle, int joinStyle, float miterLimit, float[] dashPattern, float dashOffset)
    - Lets you make dashed lines by specifying an array of opaque (entries at even array indices) and transparent (odd indices) segments. The offset, which is often 0.0, specifies where to start in the dashing pattern.

Thick Lines: Example Code

```
import java.awt.*;

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 pen
    g2d.draw(getCircle());
  }
...
```

Thick Lines: Example Output
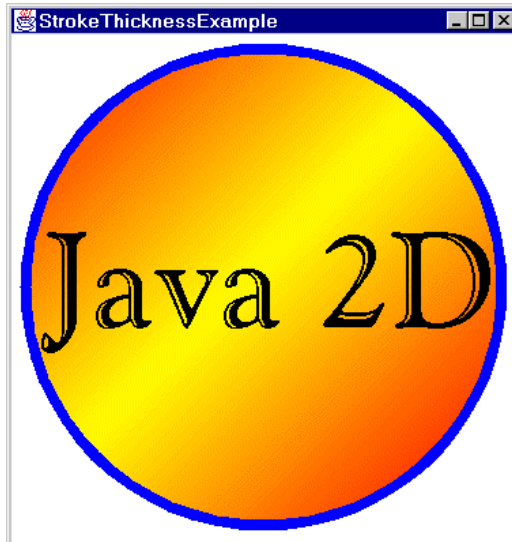
Dashed Lines: Example Code

```java
public class DashedStrokeExample extends FontExample {
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawBigString(g2d);
    drawDashedCircleOutline(g2d);
  }

  protected void drawDashedCircleOutline(Graphics2D g2d) {
    g2d.setPaint(Color.blue);
    // 30 pixel line, 10 pxl gap, 10 pxl line, 10 pxl gap
    float[] dashPattern = { 30, 10, 10, 10 };
    g2d.setStroke(new BasicStroke(8, BasicStroke.CAP_BUTT,
                                  BasicStroke.JOIN_MITER, 10,
                                  dashPattern, 0));

    g2d.draw(getCircle());
  }
```
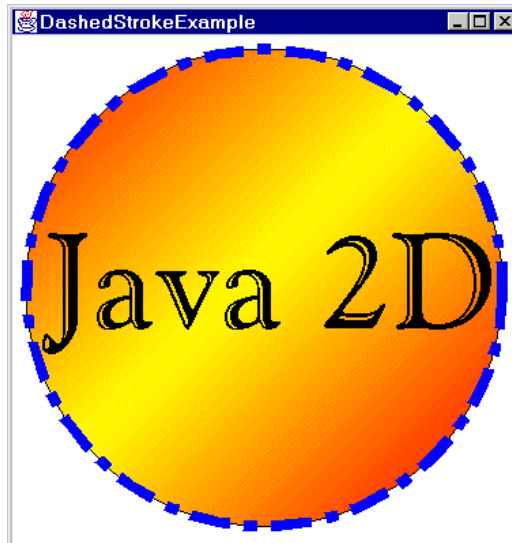
Dashed Lines: Example Output

Join Styles

- **JOIN_MITER**
  - Extend outside edges of lines until they meet
    - This is the default
- **JOIN_BEVEL**
  - Connect outside corners of outlines with straight line
- **JOIN_ROUND**
  - Round off corner with a circle that has diameter equal to the pen width

Cap Styles

- **CAP_SQUARE**
  - Make a square cap that extends past the end point by half the pen width
    - This is the default
- **CAP_BUTT**
  - Cut off segment exactly at end point
    - Use this one for dashed lines.
- **CAP_ROUND**
  - Make a circle centered on the end point. Use a diameter equal to the pen width.

Cap and Join Styles: Example Code

```
public class LineStyles extends JPanel {
  private int[] caps =
    { BasicStroke.CAP_SQUARE, BasicStroke.CAP_BUTT,
      BasicStroke.CAP_ROUND };
  private int[] joins =
    { BasicStroke.JOIN_MITER, BasicStroke.JOIN_BEVEL,
      BasicStroke.JOIN_ROUND };

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    g2d.setColor(Color.blue);
    for(int i=0; i>caps.length; i++) {
      BasicStroke stroke =
        new BasicStroke(thickness, caps[i], joins[i]);
      g2d.setStroke(stroke);
      g2d.draw(path);
      ...
```

Cap and Join Styles: Example Output
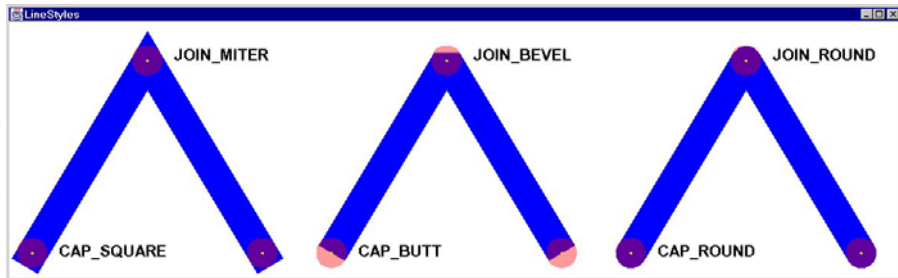
www.corewebprogramming.com

Coordinate Transformations

- **Idea:**
  - Instead of computing new coordinates, move the coordinate system itself.
- **Available Transformations**
  - Translate (move).
  - Rotate (spin).
  - Scale (stretch evenly)
  - Shear (stretch more as points get further from origin)
  - Custom. New point (x2, y2) derived from original point (x1, y1) as follows:

```
[ x2]    [  m00  m01  m02  ] [ x1 ]    [ m00x1 + m01y1 + m02 ]
[ y2] = [  m10  m11  m12  ] [ y1 ] = [ m10x1 + m11y1 + m12 ]
[ 1 ]    [   0    0    1   ] [ 1 ]    [          1          ]
```

www.corewebprogramming.com

Translations and Rotations: Example Code

```
public class RotationExample
        extends StrokeThicknessExample {
  private Color[] colors = { Color.white, Color.black };

  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++) {
      // Rotate the coordinate system around current
      // origin, which is at the center of the circle.
      g2d.rotate(Math.PI/8.0);
      g2d.setPaint(colors[i%2]);
      g2d.drawString("Java", 0, 0);
```

Translations and Rotations: Example Output

Shear Transformations

- **Meaning of Shear**
  - X Shear
  If you specify a non-zero x shear, then x values will be more and more shifted to the right the farther they are away from the y axis. For example, an x shear of 0.1 means that the x value will be shifted 10% of the distance the point is away from the y axis.
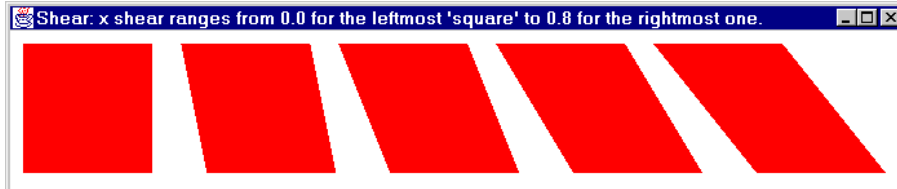 - Y Shear
  Points are shifted down in proportion to the distance they are away from the x axis.

Shear: Example Code

```java
public class ShearExample extends JPanel {
  private static int gap=10, width=100;
  private Rectangle rect =
    new Rectangle(gap, gap, 100, 100);

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    for (int i=0; i<5; i++) {
      g2d.setPaint(Color.red);
      g2d.fill(rect);
      // Each new square gets 0.2 more x shear
      g2d.shear(0.2, 0.0);
      g2d.translate(2*gap + width, 0);
    }
  }
```

Shear: Example Output

Shear: x shear ranges from 0.0 for the leftmost 'square' to 0.8 for the rightmost one.

Rendering Hints

- **Default:**
  - Faster drawing, possibly less accuracy
- **Rendering Hints:**
  - Let you request more accurate (but generally slower) drawing. Eg:

```
RenderingHints renderHints =
  new RenderingHints(RenderingHints.KEY_ANTIALIASING,
                     RenderingHints.VALUE_ANTIALIAS_ON);
renderHints.put(RenderingHints.KEY_RENDERING,
                RenderingHints.VALUE_RENDER_QUALITY);
...
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2d = (Graphics2D)g;
  g2d.setRenderingHints(renderHints);
  ...
```

Summary

- **General**
  - If you have Graphics, cast it to Graphics2D
  - Create Shape objects, then call Graphics2D's draw and fill methods with shapes as args.
- **Paint styles**
  - 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.
- **Transparent drawing**
  - Use AlphaComposite for transparency. Create one via AlphaComposite.getInstance with a type of AlphaComposite.SRC_OVER.

www.corewebprogramming.com

Summary (Continued)

- **Local fonts**
  - Before using them you must call getAllFonts or getAvailableFontFamilyNames. Then supply name to Font constructor and specify font via setFont.
- **Stroke styles**
  - BasicStroke lets you set pen thickness, dashing pattern, and line cap/join styles. Then call setStroke.
- **Coordinate transformations**
  - Let you move the coordinate system rather than changing what you draw. Simple transforms: call translate, rotate, scale, and shear. More complex transforms: supply matrix to AffineTransform constructor, then call setTransform.
- **Rendering Hints**
  - Improve drawing quality or enable antialiasing

www.corewebprogramming.com

Questions?