CS324e - Elements of Graphics and Visualization

Gradients
GUIs

Windows 3 1990

Mac 1984
GUls
Gradients

• Gradient: vary color from one to another interpolating in between

• Current trend (~2010) is to use gradients everywhere in interfaces and graphic design

• Java provides some built-in classes to create and apply gradients
  – GradientPaint
  – LinearGradientPaint
  – RadialGradientPaint
GradientPaint

• Define two end points and two colors
• color varies along a line drawn between two points
  – same colors along lines drawn perpendicular to line connecting points
• Set Graphics2D object paint to the GradientPaint
• Anything drawn or filled uses gradient
GradientPaint Example
GradientPaint

```java
public GradientPaint(float x1,
    float y1,
    Color color1,
    float x2,
    float y2,
    Color color2)

Constructs a simple acyclic GradientPaint object.

Parameters:
    x1 - x coordinate of the first specified Point in user space
    y1 - y coordinate of the first specified Point in user space
    color1 - Color at the first specified Point
    x2 - x coordinate of the second specified Point in user space
    y2 - y coordinate of the second specified Point in user space
    color2 - Color at the second specified Point
```

Rectangle2D temp

```java
Rectangle2D.Double(50, 25, 400, 300);
```

GradientPaint gp = new GradientPaint(100, 50, Color.BLUE,
                                                             350, 300, Color.RED);

```java
g2.setPaint(gp);
g2.fill(temp);
```
GradientPaint

• By default gradient is acyclic
  – beyond endpoints color is same as color at endpoint
GradientPaint

• Set GradientPaint to cyclical
  — another constructor
LinearGradientPaint

- Until Java 6.0 in 2006 GradientPaint was only gradient
- LinearGradientPaint added in version 6.0
- GradientPaint only allowed 2 paints
- LinearGradientPaint allows 2 or more colors
- Can alter "distance" between two paints
- More options
LinearGradientPaint

• 4 Colors
LinearGradientPaint

- Define 2 endpoints
- array of fractions and array of colors must be equal in length
- fractions defines portion to use that color

```
public LinearGradientPaint(float startX, float startY, float endX, float endY, float[] fractions, Color[] colors)
```

Constructs a LinearGradientPaint with a default NO_CYCLE repeating method and SRGB color space.

**Parameters:**

- `startX` - the X coordinate of the gradient axis start point in user space
- `startY` - the Y coordinate of the gradient axis start point in user space
- `endX` - the X coordinate of the gradient axis end point in user space
- `endY` - the Y coordinate of the gradient axis end point in user space
- `fractions` - numbers ranging from 0.0 to 1.0 specifying the distribution of colors along the gradient
- `colors` - array of colors corresponding to each fractional value
private void linearGradientPaint(Graphics2D g2) {
    Rectangle2D temp = new Rectangle2D.Double(0, 0, getWidth(), getHeight());

    float[] fractions = {0, 0.33f, 0.67f, 1};
    Color[] colors = {Color.RED, Color.YELLOW, Color.GREEN, Color.MAGENTA};

    LinearGradientPaint lgp = new LinearGradientPaint(100, 100, getWidth() - 100, getHeight() - 50, fractions, colors);

    g2.setPaint(lgp);
    g2.fill(temp);

    g2.setColor(Color.BLACK);
    g2.fillOval(100, 100, 5, 5);
    g2.fillOval(getWidth() - 100, getHeight() - 50, 5, 5);
}
LinearGradientPaint

• Change Fractions

• \{0, 0.1f, 0.9f, 1\} // f for float

• fractions must be in strictly increasing order
LinearGradientPaint - REFLECT

• Previous examples were acyclic
• Cycle can be set to reflect or repeat
LinearGradientPaint - REPEAT

![Gradient Examples](image-url)
LinearGradientPaint

```java
LinearGradientPaint lgp = new LinearGradientPaint(150, 150, 
getTL() - 150, getHeight() - 150,
fractions, colors,
MultipleGradientPaint.CycleMethod.REPEAT);
```

• or REPEAT
RadialGradientPaint

• Specify two or more colors, a center point, and a radius
• can also set the focal point to a spot other than the center
• May be acyclic (NO_CYCLE), REFLECT, or REPEAT similar to LinearGradientPaint
RadialGradientPaint
private void radialGradientPaint(Graphics2D g2) {

    int rad = 300;
    int cx = 100;
    int cy = 100;

    Ellipse2D temp = new Ellipse2D.Double(cx, cy, rad, rad);

    float[] fractions = {0, 1};
    Color[] colors = {Color.WHITE, Color.RED};

    RadialGradientPaint rgp = new RadialGradientPaint(
        cx + rad / 2, cy + rad / 2, 50,
        fractions, colors);

    g2.setPaint(rgp);
    g2.fill(temp);
}
RadialGradientPaint Constructor

public RadialGradientPaint(float cx,
    float cy,
    float radius,
    float[] fractions,
    Color[] colors)

Constructs a RadialGradientPaint with a default NO_CYCLE repeating method and SRGB color space, using the center as the focus point.

Parameters:

cx - the X coordinate in user space of the center point of the circle defining the gradient. The last color of the gradient is mapped to the perimeter of this circle.

cy - the Y coordinate in user space of the center point of the circle defining the gradient. The last color of the gradient is mapped to the perimeter of this circle.

radius - the radius of the circle defining the extents of the color gradient

fractions - numbers ranging from 0.0 to 1.0 specifying the distribution of colors along the gradient

colors - array of colors to use in the gradient. The first color is used at the focus point, the last color around the perimeter of the circle.

• Not only one!
RadialGradientPaint

• Example with ellipse not centered
• Can either move ellipse or change focal point of gradient paint.
• Slightly different effects
RadialGradientPaint - Altered Focal Point
Putting Gradients to Use

• Create a reflection of an image using gradients and alpha composite
• Draw an image
• Draw the same image below it, reflected
• Create a paint that varies alpha from half to 0
• Use Alpha Composite to combine alpha paint and upside down image
private BufferedImage
    createReflection(BufferedImage image) {

        int height = image.getHeight();

        BufferedImage result = new BufferedImage(image.getWidth(),
            height * 2,
            BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2 = result.createGraphics();

        // Paints original image
        g2.drawImage(image, 0, 0, null);
// Paints mirrored image
g2.scale(1.0, -1.0);
g2.drawImage(image, 0,
     -height - height, null);
g2.scale(1.0, -1.0);
Gradient
(alpha mask)

```java
g2.setColor(Color.BLACK);
g2.fillRect(0, height,
    image.getWidth(),
    image.getHeight());

    // Move to the origin of the clone
    g2.translate(0, height);

    Color grayAlpha
        = new Color(1.0f, 1.0f,
                        1.0f, 0.5f);

    Color transparent
        = new Color(1.0f, 1.0f,
                        1.0f, 0.0f));

    // Creates the alpha mask
    GradientPaint mask;
    mask = new GradientPaint(0, 0,
            grayAlpha,
            0, height / 2,
            transparent);
    Paint oldPaint = g2.getPaint();
g2.setPaint(mask);
```
Final result using
AlphaComposite
DstIn

```java
Paint oldPaint = g2.getPaint();
g2.setPaint(mask);

// Sets the alpha composite
g2.setComposite(AlphaComposite.DstIn);

// Paints the mask
g2.fillRect(0, 0,
    image.getWidth(), height);
```