Graphics

Display Drawing

- To display graphics we use panels as canvases
- There are cases when you need a custom component.
  - Usually this is when you want to paint custom 2-D graphics.
  - We often call a custom painted component a *canvas*.
- To do so, write a class that extends `JPanel`.
  - Override method `paintComponent` to tell Java how to draw it:

```
public void paintComponent(Graphics g)
```

    - Some programmers extend `JComponent`.

A drawing canvas

- Coordinate system: (0, 0) at top-left,
  x-axis increases rightward, y-axis ***downward*.**

- Component's surface is *transparent* unless drawn on.

- `JComponent`'s `paintComponent` does important things that we don't want to lose. (e.g. paints the component's background)
  - So call the method `super.paintComponent` first thing.

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    ...
}
```

A drawing canvas

- It is important never to call paintComponent directly in order to initiate a redrawing.
- To do that, one calls the repaint()-method of the panel. This is done.
  - to avoid conflicts between redrawing and other operations of the runtime system.
- A call to repaint does not immediately repaint a component.
  - it tells the runtime system that one would like the component to be redrawn.
  - The system calls paintComponent after other pending actions have been completed

Graphics methods

- drawLine(int xstart, int ystart, int xend, int yend)
  - draws a line segment between the points with coordinates (xstart, ystart) and (xend, yend).
- drawRect(int xleft, int ytop, int width, int height)
  - draws the contour of an axes-aligned rectangle. The upper left corner of the rectangle is at (xleft, ytop). It is width pixels wide and height pixels tall
- drawOval(int xleft, int ytop, int width, int height)
  - draws the contour of an ellipse. The axes of the ellipse are parallel to the coordinate axes. The upper left corner of the bounding rectangle of the ellipse is at (xleft, ytop) and is called the reference point. The horizontal axis of the ellipse has length width, the vertical one height.
- drawString(String text, int xleft, int ybottom)
  - draws the string text. The lower left corner of the bounding rectangle of the text is at (xleft, ybottom). This is the reference point.
- fillRect(int xleft, int ytop, int width, int height)
  - draws a filled rectangle using the current colour, otherwise like drawRect.
- fillOval(int xleft, int ytop, int width, int height)
  - draws a filled ellipse using the current colour, otherwise like drawOval
- setColor(Color col)
  - sets the colour of the drawing to col. This colour is used until the setColor command is used again.

Example

```
import java.awt.*;
import javax.swing.JPanel;
public class SimpleGraphicsPanel extends JPanel
{
   public SimpleGraphicsPanel()
   {
     this.setBackground(Color.white);
     this.setPreferredSize(new Dimension(300,300));
   }

   public void paintComponent(Graphics g)
   {
     super.paintComponent(g);
     g.setColor(Color.black);
     g.drawLine(10,10,100,100);
     g.setColor(Color.red);
     g.drawLine(10,100,100,10);
     g.setColor(Color.green);
     g.drawOval(120,60,70,40);
     g.setColor(Color.yellow);
     g.fillOval(230,150,30,30);
     g.setColor(Color.red);
     g.fillOval(245,150,30,30);
     g.setColor(Color.black);
     g.fillOval(238,160,30,30);
     g.setColor(Color.cyan);
     g.fillOval(10,120,100,60);
     g.setColor(this.getBackground());
     g.fillOval(50,140,100,60);
     g.setColor(Color.blue);
     g.drawString("Swing is nice.",100,200);
   }
 }
```

Example (cont.)

```
import java.awt.BorderLayout;
public class SimpleGraphicsFrame extends SimpleFrame
{

  public SimpleGraphicsFrame()
  {
    this.setTitle("Simple Graphics");

    SimpleGraphicsPanel SGP = new SimpleGraphicsPanel();
    this.getContentPane().add(SGP,BorderLayout.CENTER);

    pack();
  }
}
```

```
public class SimpleGraphicsDriver
{

  public SimpleGraphicsDriver()
  {
    SimpleGraphicsFrame SGF = new
SimpleGraphicsFrame();
    SGF.showIt();
  }
  public static void main(String[] args)
  {
    SimpleGraphicsDriver sgt = new
SimpleGraphicsDriver();
  }
}
```

Finding the screen parameters

```
Toolkit tk = Toolkit.getDefaultToolkit();
Dimension screenDim = tk.getScreenSize();
int screenHeight = screenDim.height;
int screenWidth = screenDim.width;
int pixPerInch = tk.getScreenResolution();
```

Example 2

```
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JPanel;

public class ResizePanel extends JPanel{

   public ResizePanel(){
     this.setBackground(Color.yellow);
   }

   public void paintComponent(Graphics g)
   {
     super.paintComponent(g);
     // get the current dimensions of the panel
     int currentWidth  = this.getWidth();
     int currentHeight = this.getHeight();

     // take a third of the current dimensions
     int wThird  = currentWidth/3;
     int hThird  = currentHeight/3;

    //set color to black
     g.setColor(Color.black);
  // and draw the rectangle
     g.fillRect(wThird,hThird,wThird,hThird);
    }


}
```

Example 2 (cont.)

```
import SimpleFrame.SimpleFrame;

public class ResizeFrame extends SimpleFrame{

  public ResizeFrame(){
    ResizePanel rp = new ResizePanel();
    this.setSize(500,300);

    this.getContentPane().add(rp);

  }

}
```

```
public class ResizeDriver
{
  public static void main(String[] args){
    ResizeFrame rf = new ResizeFrame();
    rf.showIt("ResizeFrame");
  }
}
```

Graphics2D

- The `Graphics` object `g` passed to `paintComponent` is a "graphical context" object to draw on the component.
  - The actual object passed in is a `Graphics2D` (can cast).

```
Graphics2D g2 = (Graphics2D) g;
```

- `Graphics2D` is a subclass of `Graphics` that adds new features, new shapes, matrix transformations, color gradients, etc.

Graphics2D methods

method name	description
`draw(`**Shape**`)`	draws the outline of a given shape object *(replaces drawRect, etc.)*
`fill(`**Shape**`)`	draws the outline and interior of a given shape object
`getPaint(), setPaint(`**Paint**`)`	returns or sets the current paint used for drawing (`Color` is one kind of `Paint`, but there are others)
`getStroke(), setStroke(`**Stroke**`)`	returns or sets the current line stroke style used for drawing (can be thin/thick, solid/dashed/dotted, etc.)
`rotate(`**angle**`)`	rotates any future drawn shapes by the given angle (radians)
`scale(`**sx, sy**`)`	resizes any future drawn shapes by the given x/y factors
`translate(`**dx, dy**`)`	moves any future drawn shapes by the given x/y amounts
`setRenderingHint(` **key, value**`)`	sets "rendering hints" such as anti-aliasing and smoothing
`shear(`**shx, shy**`)`	gives a slanted perspective to future drawn shapes
`transform(`**t**`)`	adds a transformation that will be applied to all shapes

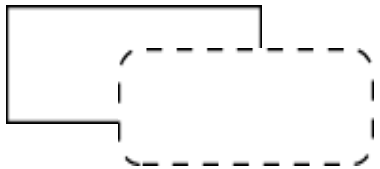
Shapes (java.awt.geom)

- `Arc2D.Double(`**x, y, w, h, start, extent, type**`)`
  An arc, which is a portion of an ellipse.

- `Ellipse2D.Double(`**x, y, w, h**`)`

- `Line2D.Double(`**x1, y1, x2, y2**`)`
  `Line2D.Double(`**p1, p2**`)`
  A line between two points.

- `Rectangle2D.Double(`**x, y, w, h**`)`
- `RoundRectangle2D.Double(`**x, y, w, h, arcx, arcy**`)`



- `GeneralPath()`
  A customizable polygon.

Methods of all shapes

method name	description

`contains(`**`x, y`**`)`
`contains(`**`point`**`)`
`contains(`**`rectangle`**`)`
	whether the given point is inside the bounds of this shape

`getBounds()`	a rectangle representing the bounding box around this shape

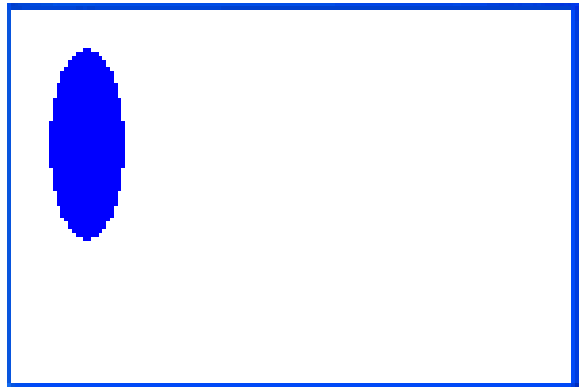
`getCenterX/Y()`
`getMinX/Y()`
`getMaxX/Y()`
	various corner or center coordinates within the shape

`intersects(`**`x, y, w, h`**`)` `intersects(`**`rectangle`**`)`	whether this shape touches the given rectangular region

Drawing with objects

```
public class MyCanvas extends JComponent {
    public MyCanvas() {
        this.setBackground(Color.WHITE);
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        Shape shape = new Ellipse2D.Double(10, 10, 20, 50);
        g2.setPaint(Color.BLUE);
        g2.fill(shape);
    }
}
```

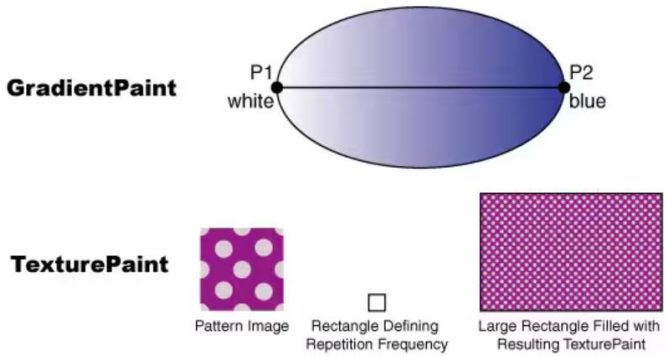


Colors and paints

- **`Color`** (a simple single-colored paint)
  - `public Color(int r, int g, int b)`
  - `public Color(int r, int g, int b, int alpha)`
    - a partially-transparent color (range 0-255, 0=transparent)

- **`GradientPaint`** (a smooth transition between 2 colors)
  - `public GradientPaint(float x1, float y1, Color color1, float x2, float y2, Color color2)`

- `java.awt.TexturePaint`
  (use an image as a "paint" background)



Strokes (pen styles)

`Graphics2D`

- `public void` **`setStroke`**`(Stroke s)`
  Sets type of drawing pen (color, width, style) that will be used by this `Graphics2D`.

- **`BasicStroke`**

  A pen stroke for drawing outlines.

  - `public BasicStroke(float width)`
  - `public BasicStroke(float width, int cap, int join)`
  - `public BasicStroke(float width, int cap,`
    `int join, float miterlimit,`
    `float[] dash, float dash_phase)`
    - `cap` can be: `CAP_BUTT, CAP_ROUND, CAP_SQUARE`
    - `join` can be: `JOIN_BEVEL, JOIN_MITER, JOIN_ROUND`

Repainting

- Most canvases are drawing the state of fields, a model, etc.
  - When the state updates, you must tell the canvas to re-draw itself.
  - But you can't call its `paintComponent` method, because you don't have the `Graphics g` to pass.

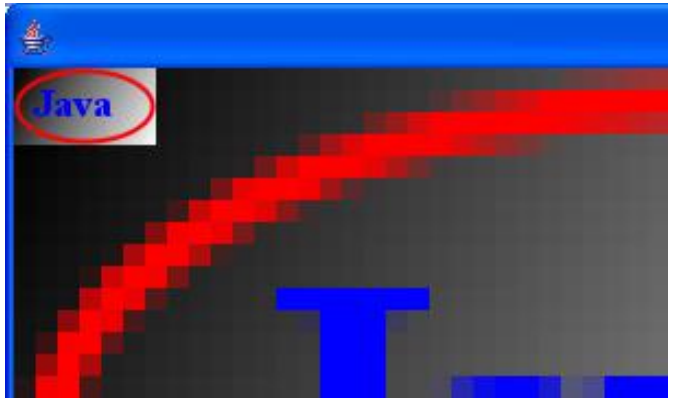
  - The proper way is to call `repaint` on the canvas instead:

```
public void repaint()
```

Anti-aliasing

- Onscreen text and shapes can have jagged edges, or *aliases*. These can be removed by smoothing, or *anti-aliasing*, the component.
  - `public void setRenderingHint(key, value)`
  - Example:

```
g2.setRenderingHint(
        RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);
```



Example 3

```
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Rectangle2D;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Path2D;
import javax.swing.JPanel;

public class DrawingPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        // Draw a rectangle
        Rectangle2D rect = new Rectangle2D.Double(100, 100, 200, 100);
        g2d.draw(rect);

        // Draw an oval
        Ellipse2D oval = new Ellipse2D.Double(400, 100, 200, 100);
        g2d.draw(oval);

        // Draw a polygon
        Path2D polygon = new Path2D.Double();
        polygon.moveTo(200, 300);
        polygon.lineTo(300, 400);
        polygon.lineTo(100, 400);
        polygon.closePath();
        g2d.draw(polygon);
    }
}
```

Example 4

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class DrawingPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        // Set font and color
        g2d.setFont(new Font("Arial", Font.BOLD, 24));
        g2d.setColor(Color.RED);

        // Draw a string
        g2d.drawString("Hello, Java 2D Graphics!", 200, 50);
    }
}
```

Example 5

```
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.JPanel;

public class DrawingPanel extends JPanel {

    private Image image;

    public DrawingPanel() {
        try {
            image = ImageIO.read(new File("path/to/your/image.jpg"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        // Draw an image
        if (image != null) {
            g2d.drawImage(image, 300, 200, this);
        }
    }
}
```

Creating images

```
// import java.awt.image.*;
```

`BufferedImage`

A blank graphic image buffer surface onto which you can draw

- `public BufferedImage(int w, int h, int type)`
  - where `type` is a constant such as `BufferedImage.TYPE_INT_ARGB`
- `public Graphics getGraphics()`
  - returns a graphical pen for "drawing on" this image
- you can draw a `BufferedImage` onto the screen from within the `paintComponent` method of your canvas:
  - `g.drawImage(`**`BufferedImage, x, y,`** `this);`

Example 6

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class DrawingPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        // Create a buffered image texture patch of size 5x5
          BufferedImage bi = new BufferedImage(5, 5,
                                BufferedImage.TYPE_INT_RGB);
          Graphics2D big = bi.createGraphics();
          // Render into the BufferedImage graphics to create the texture
          big.setColor(Color.green);
          big.fillRect(0,0,5,5);
          big.setColor(Color.lightGray);
          big.fillOval(0,0,5,5);

          // Create a texture paint from the buffered image
          Rectangle r = new Rectangle(0,0,5,5);
          TexturePaint tp = new TexturePaint(bi,r,TexturePaint.NEAREST_NEIGHBOR);

          // Add the texture paint to the graphics context.
          g2.setPaint(tp);

          // Create and render a rectangle filled with the texture.
          g2.fillRect(0,0,200,200);
     }
```

Example 7

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class DrawingPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

     //Create a BufferedImage object from image file
     BufferedImage image = ImageIO.read(new File("java.png"));

     //Create image area doubled in width
     Rectangle2D.Float imageArea = new Rectangle2D.Float(0, 0, image.getWidth() * 2, image.getHeight());
     //Create texture brush from the image
     TexturePaint paint = new TexturePaint(image, imageArea);

     //Create rectangle
     Rectangle2D.Float shape = new Rectangle2D.Float(0, 0, 200, 100);

     //Set this texture brush as current paint
     g2.setPaint(paint);
     //Fill rectangle
     g2.fill(shape);
```

Example 8

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class DrawingPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        GradientPaint gp = new GradientPaint(50.0f, 50.0f, Color.blue
                    50.0f, 250.0f, Color.green);
     g2.setPaint(gp);
     g2.fillRect(50, 50, 200, 200); }
```