

# Graphics and Painting

## Lecture 17 CGS 3416 Spring 2016

April 17, 2017

## paint() methods

- Lightweight Swing components that extend class `JComponent` have a method called `paintComponent`, with this prototype:

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

- Another similar method is the `paint` method in class `Component` (and thus all its children) with this prototype:

```
public void paint(Graphics g)
```

- The `JComponent` version of `paint()` actually delegates its work to three methods: `paintComponent`, `paintBorder`, and `paintChildren`

# paint() methods

- The idea behind `paint()` is that they are invoked for any component *automatically* whenever that component needs to be drawn or re-drawn. Some examples of triggering events:
  - When the component first is placed on the application.
  - When the component is resized.
  - When the component is covered by some other application, then uncovered and comes to the forefront again.
- Since this is triggered by such events, the programmer seldom needs to call `paint()` or `paintComponent()` explicitly.
- The programmer can call `repaint()` (also a Component method) to force the paint operation, if the need arises (i.e. some situation not covered by the automatic calls to `paint()`).

## More on paint()

- These methods both take as a parameter a reference variable of type `Graphics` – which is an abstract class.
- The object will be a subtype that handles the drawing context for the given platform.
- For Swing components, it is usually sufficient to just define `paintComponent()` for drawing aspects, unless you want to control the other parts (border, children) as well.
- So, what can we DO in the `paint()` or `paintComponent()` methods? Pretty much anything that's available in the `Graphics` class, and then some.

# class Graphics and other useful helper classes

## The Graphics Class

- Helps manage drawing on the screen for GUI applications and applets.
- Keeps track of state information like current font, current color, the Component object being drawn on, and more.
- Has methods for drawing various kinds of shapes (lines, ovals, polygons, rectangles, etc) as well as strings.
- Also has methods for setting the font, the color, the current clipping area, the paint mode, and other status information.

# The Color Class

- Used for specifying colors in components and drawings.
- Colors stored and specified with RGB (Red Green Blue) values.
- RGB values can be specified with ints (0-255) or floats (0.0-1.0).
- Color constants exist for common colors (Color.BLUE, Color.GREEN, etc).
- To find out or set the current drawing color, use the Graphics methods `getColor()` and `setColor()`. Example:

```
g.setColor(Color.MAGENTA);  
g.setColor(new Color(255, 128, 3)); //RGB values
```

- `JColorChooser` - a `javax.swing` component that enables application users to choose colors.

# The Font Class

- Specify fonts used in Graphics drawings.
- Physical fonts are actual fonts on a system – these depend on platform and what fonts are installed on a system.
- Logical fonts are the 5 font families supported in Java: Serif, Sans Serif, Monospaced, Dialog, and DialogInput. When using logical fonts, an appropriate font on the given system will be chosen.
- Font constructor takes three parameters: font name, font style, font size
  - Font name can be physical or logical.
  - Font styles are plain, italics, or bold.
  - Font size measured in *points*.

# Font Examples

- To set or find out the current drawing font, use the Graphics methods `getFont()` and `setFont()`. Example:

```
Font f = g.getFont(); // retrieve current font
g.setFont(new Font("Serif", Font.ITALICS, 12));
```

- Other methods available in class `Font` to set or retrieve properties for a `Font` object.



# The FontMetrics class

- Abstract class. Encapsulates information and properties about the rendering of a font on screen.
- Helps track more specific font information like height, descent, ascent, and leading (interline spacing).
- Graphics class has a couple of methods named

```
getFontMetrics():  
FontMetrics m1, m2;  
m1 = g.getFontMetrics(); //current font info  
m2 = g.getFontMetrics(f1); //info about font f1
```

# The Polygon Class

- Helper class for representing information about Polygons.
- Stores a list of  $(x,y)$  coordinate pairs, representing vertices of a polygon.
- Several Graphics class methods are for drawing polygons - `drawPolygon()`, `drawPolyLine`, `fillPolygon`.
- There are versions of these last two that take a Polygon object as a parameter.

# Java2D

The Java2D API provides advanced graphics capabilities, for more detailed and complex two-dimensional drawing.

- Allows more complex drawing, like lines of varying thickness, filling shapes with colors and patterns, drawing dashed lines, composite overlapping text and graphics, gradients and textures, and more.
- Need to use an instance of class `Graphics2D`, which is a subclass of class `Graphics`.
- Must cast the `Graphics` object in the `paintComponent()` method into a `Graphics2D` reference when using:

```
Graphics2D g2d = (Graphics2D) g;
```

- For more details, look up the class on the Oracle Java Documentation website.

# Java2D Packages

Java2D involves a variety of packages:

- `java.awt`
- `java.awt.image`
- `java.awt.color`
- `java.awt.font`
- `java.awt.geom`
- `java.awt.print`
- `java.awt.image.renderable`