Appendix F: Java Graphics

CS 121

Department of Computer Science
College of Engineering
Boise State University

November 2, 2015

Topics

- Graphics and Images
- Coordinate System
- Representing Color
- Drawing Shapes
- Scalable Drawings
- Simple Animation

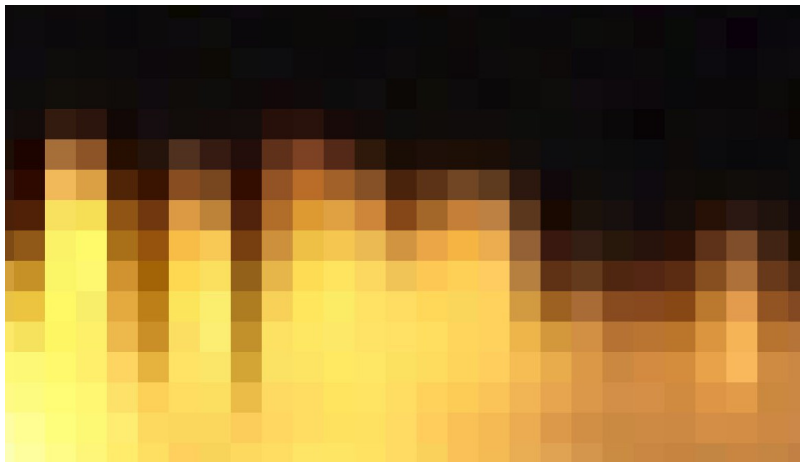
Graphics

- A picture or drawing must be digitized for storage on a computer.
- A picture is made up of pixels (picture elements), and each pixel is stored separately.
- The picture resolution is the number of pixels used to represent a picture.
- The number of pixels that can be displayed on a screen is called the screen resolution

Images

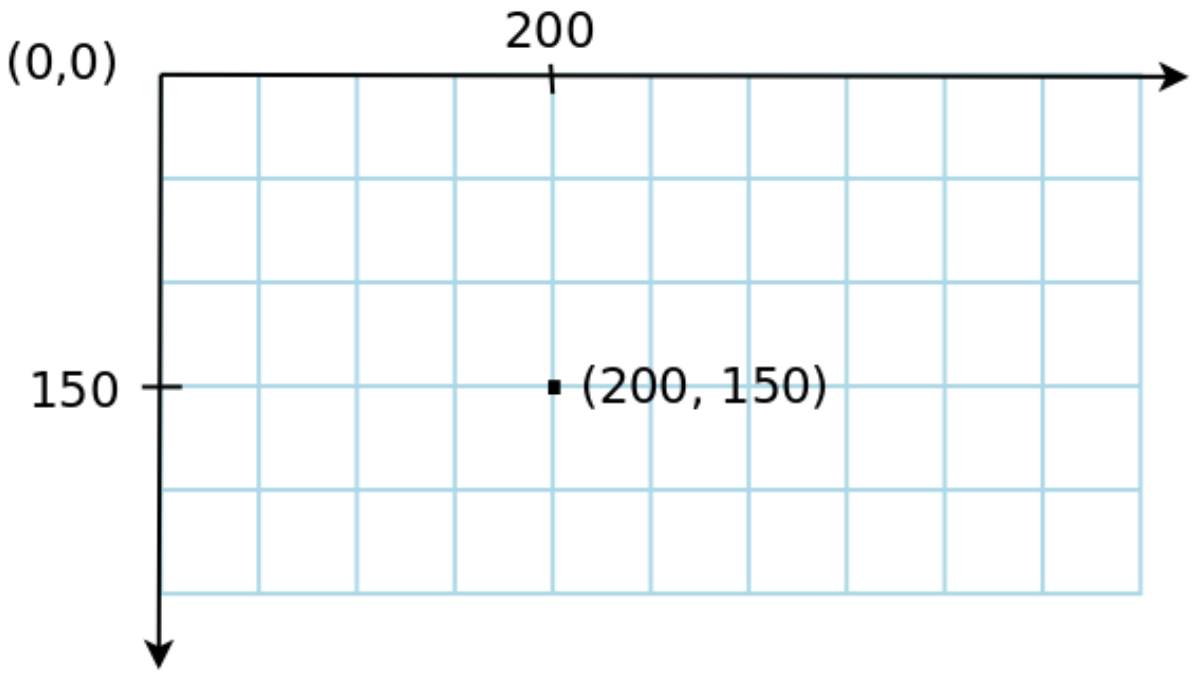
- A medium resolution image (original image was 1000 $\times$ 563 pixels).

- The same image zoomed in to show pixels.



Graphics Coordinate System

- Each pixel is identified using a two-dimensional coordinate system.
- In graphics, the origin is at the top left corner with x coordinate increasing to the right and y coordinate increasing going down.



Representing Color

- A black and white picture can be represented with 1 bit per pixel (0 = white, 1 = black). A grayscale picture can be represented with 8 bits per pixel (0-255).
- A colored picture can be represented as a mixture of primary colors Red, Green, and Blue. Each color is represented by three numbers between 0 and 255 that collectively are called an RGB value. How many colors can we represent with the RGB representation?
- In Java, color is represented as a `Color` class (from the `java.awt` package)

```
Color myColor = new Color(0, 255, 255);
```

- Some predefined colors in the `Color` class.

Color	Object	RGB value
black	`Color.black`	0, 0, 0
white	`Color.white`	255, 255, 255
red	`Color.red`	255, 0, 0
green	`Color.green`	0, 255, 0
blue	`Color.blue`	0 , 0, 255
yellow	`Color.yellow`	255, 255, 0
cyan	`Color.cyan`	0, 255, 255

Graphics Class (1)

- We will use the `Graphics` class from the `java.awt` package for drawing shapes.
- The `Graphics` class provides methods for drawing lines, rectangles, ovals, arcs and strings among others.
- Shapes drawn by the `Graphics` class can be *unfilled* or *filled*.
- The method parameters specify coordinates and sizes.
- Shapes with curves, like an oval, are usually drawn by specifying the shape's *bounding rectangle*.
- An *arc* is a section of an oval.

Graphics Class (2)

- Selected methods from the Graphics class.

`drawLine(int x1, int y1, int x2, int y2)` Draws a line between the points (x1, y1) and (x2, y2)

`drawRect(int x, int y, int width, int height)`
`fillRect(int x, int y, int width, int height)`
Draws/fills the specified rectangle.

`drawOval(int x, int y, int width, int height)`
`fillOval(int x, int y, int width, int height)`
Draws/fills the oval bounded by the specified rectangle.

`drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)`
`fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)`
Draws/fills the arc bounded by the specified rectangle.

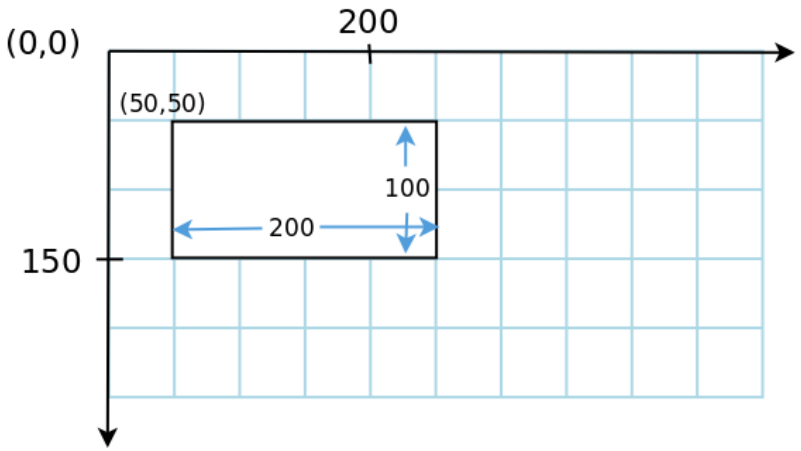
`drawString(String str, int x, int y))` Draws the text given by the specified string.

`getColor()`
`setColor(Color c)`
Gets/sets the current color.

Graphics Class (3)

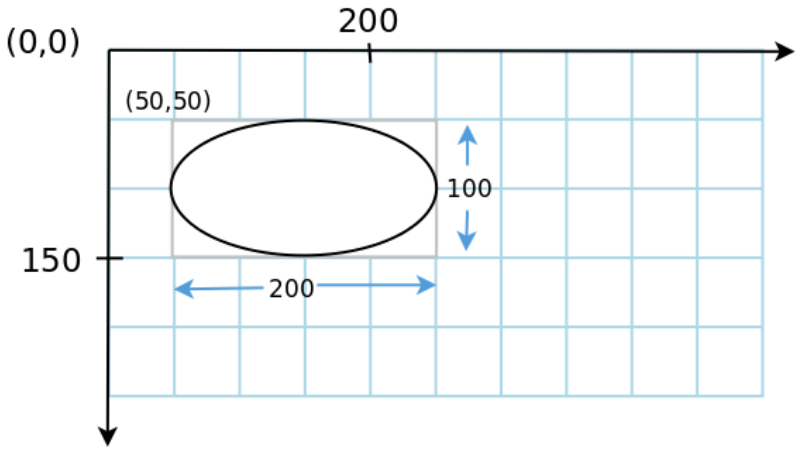
- The `drawRect` method:

```
page.drawRect(50, 50, 200, 100);
```



- The `drawOval` method:

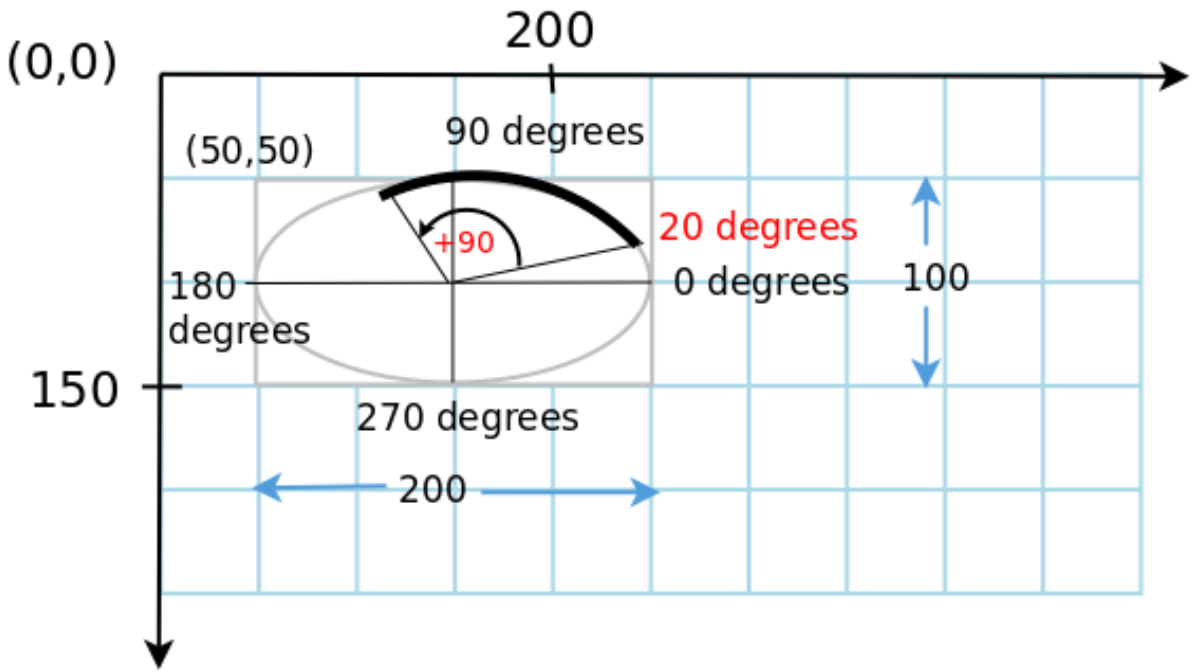
```
page.drawOval(50, 50, 200, 100);
```



Graphics Class (4)

- The `drawArc` method:

```
page.drawArc(50, 50, 200, 100, 20, 90);
```



Examples

- We will use a template that extends the `JPanel` class from the `javax.swing` package. Focus only on the `paintComponent` method!
- Examples:
  - BasicShapes.java
  - Shapes.java
  - Snowman.java
  - **In-class Exercise**: Modify the snowman program as follows:
    - Move the sun to the upper right
    - Display your name in the upper left corner of the picture
    - Make the snowman frown instead of smile
    - Shift the entire snowman to the left by 20 pixels
    - **Further Exercise**. Scale the Snowman to half the size!

Graphics Techniques

- Basic techniques for drawing:
  - Translation (Illustrated in the Snowman example using the MID variable)
  - Centering
  - Scaling
- This example illustrates how to make the graphics center and scale automatically if the user resizes the window.
  - DrawPieChart.java
  - DrawPieChartScalable.java
- An example that shows how to use a custom font and center a String using font metrics: CenterText.java
- Another example that shows how to draw thicker lines: Strokes.java
- Another example that shows how to load an image: ImageAvatar.java

Animation

- Animation involves drawing the picture multiple times (with incremental variation) per second using a timer to create the illusion of movement.
- The individual pictures are referred to as frames in movies (animated or otherwise)
- These examples show how we can animate our drawings!
  - SimpleAnimation.java
  - DigitalClock.java

Summary

- How graphics coordinate system works
- How color is represented
- How to center, scale and translate drawings
- How animation works
- Using `Graphics`, `Color`, `Font` and related classes

Exercises

- Read Appendix F (pp. 965–973).
- **Recommended Homework**:
  - Exercises: EX F.2 – F.6.
  - Projects: PP F.4, PP F.15.