Drawings
MODULAR IV

1

Objectives

How to draw simple shapes on an image.

How to set the color to draw with

How to set the font to write with

How to draw text on top of an image

2

- We already learned a lot about how to create and manipulate images in various ways in Java.

- Now we will look at a class called java.awt.Graphics, which allows us to draw lines, shapes and text on an image.

- To use this class, you'll need to import java.awt.Graphics or import java.awt.*

3

Basic Drawing

- Drawing on a picture is really just setting pixels to the color we want.

- For example, we can draw a horizontal line across a picture by setting all of the pixels in one row of the picture to the color black
  - We could add a lines to a picture every 20 pixels:
  - For Vertical lines:
    - Start with x = 20, x < width, x += 20
      - Start with y = 0, y < height, y++
  - For horizontal lines:
    - Start with x = 0, x < width, x++
      - Start with y = 20, y < height, y+=20

4

learn by doing

- Write a method drawGrid in your Picture class to add horizontal and vertical lines to a picture every 20 pixels.

- Test your method using:
  Picture p = new Picture ("<file location>");
  p.drawGrid();
  p.show();

5

solution

```
public void drawGrid() {
    for (int x = 20; x < this.getWidth(); x+=20) {
        for (int y = 0; y < this.getHeight(); y++) {
            this.getPixel(x,y).setColor(Color.BLACK);
        }
    }
    for (int x = 0; x < this.getWidth(); x++) {
        for (int y = 20; y < this.getHeight(); y+=20) {
            this.getPixel(x,y).setColor(Color.BLACK);
        }
    }
}
```

6

Java Graphics

- You can use the same basic idea as above to draw a circle, a string of characters, lines, or other shapes on an image.
- But determining which pixels to change would require some thought. Since circles, strings, lines, and other shapes are basic elements of almost any on-screen drawing, good programming practice dictates that we don't ask every programmer to re-invent the wheel.
- Instead, Java provides the Graphics class to do this sort of stuff.
- Every picture object has a Graphics object associated with it, which has methods for drawing simple lines, shapes, and text. The method for retrieving the Graphics object is:

  pictureObj.getGraphics()

7

Java Graphics

- The methods in the java.awt.Graphics class let you paint on an image as follows:
  - Pick any color to draw with.
  - Draw shapes: circles, ellipses, rectangles, lines, arcs, polygons
  - Shapes drawn will cover whatever is on the image already, including other drawings.
 - Draw text: Set the font and write text strings in that font.

8

Java Graphics

- Set the current color using:
  - graphicsObj.setColor(Color c);
- Similarly, you can get the current color using:
  - graphicsObj.getColor();
- As described when we first began manipulating images, colors can be created from scratch using new Color(red, green, blue), referenced using pre-defined constants (such as Color.RED), and so on. Make sure that you import java.awt.* or java.awt.Color before you use colors, so the class is defined.

9

Graphics Environment

- As we have discussed, the upper left corner of a Picture is at (0,0), while the bottom right corner is at (width, height). Every Graphics object in Java functions the same way: Any commands you give it assume that the x and y coordinates of a point are counted from (0,0) at top left of the image, something like this:

(0,0)

(0,0) is at the top left
X increases to the right
Y increases to the bottom
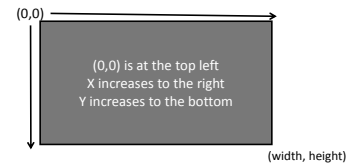
(width, height)

10

Drawing Circles and Ovals

- To draw a circle or oval, we can use

  drawOval(x,y,width,height) or fillOval(x,y,width,height)

- x and y are the coordinates of the top left corner of the enclosing rectangle, and width and height are the width and height of the enclosing rectangle.

(x,y)
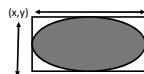
- fillOval(x,y,width,height) does the same thing as drawOval(...), except it fills in the inside of the oval.

11

Fonts and Strings

- One of the most common things drawn on screen in today's interactive systems is text. Text is drawn using a method call drawString().
- In addition to the color of text, you can also control its font.
  - First import java.awt.Font
  - Font labelFont = new Font("TimesRoman", Font.BOLD, 24); (styles are PLAIN, BOLD, ITALIC, and BOLDITALIC. These are all available as fields in the Font class).
  - Tell the graphics object to use this font: graphicsObj.setFont(labelFont)
  - use the graphics object to get font information:
    - graphicsObj.getStyle() to get font style
    - graphicsObj.getSize() for the point size,
    - graphicsObj.getName() for the font name (e.g. "TimesRoman")
    - graphicsObj.getFamily() for the font family (e.g. Helvetica).

12

Fonts and Strings

- The graphics object provides a method to draw a string on an image:
  graphicsObj.drawString("string", leftX, baselineY).
- The location coordinates are defined as follows:
  - *leftX* simply refers to the leftmost part of the string.
  - *baselineY* refers to the Y position of the base of characters that do not have a descender. For ex., the 'g' descends below baselineY.
- We can get the font information by using the FontMetrics class.
  import java.awt.FontMetrics;
  FontMetrics currFontMetrics = graphicsObj.getFontMetrics();
  int baselineY = currFontMetrics.getHeight() -
    currFontMetrics.getDescent();
  int width = currFontMetrics.stringWidth("string");
- Note that width isn't required to draw a string, but can be useful regardless.

13

Other types of drawing elements:

graphicsObj.drawLine(x1,y1,x2,y2);

graphicsObj.fillPolygon(xArray, yArray, numPoints);
  Where xArray holds all the x points and yArray holds all the y points

graphicsObj.drawArc(x, y, width, height, startAngle, arcAngle);

graphicsObj.fillArc(x, y, width, height, startAngle, arcAngle);

graphicsObj.drawRect(x, y, width, height);

graphicsObj.drawRoundRect(x, y, width, height, arcWidth, arcHeight);

14

Summary

- You can draw by changing the color of the pixels directly. Or you can use *java.awt.Graphics* to do drawings
  - *import java.awt.Graphics*
  - Get a graphics object from a picture object with:
    Graphics graphics = pictureObj.getGraphics();
  - Set the color to draw with: *graphics.setColor(color);*
  - Do simple drawing.
  - Draw a filled in version of everything (but line) using *fillobject*

15