CS 152L
Computer Programming Fundamentals
Using Joel's Picture Class

Instructor:
Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/
Office: Farris Engineering Center (FEC) room 319

Pixel

A **pixel** is a square of light on a computer display.

Window with 200 x 200 pixel drawing area.

12 x 12 pixel magnification of drawing area.
Computer Coordinate System

x-axis

<table>
<thead>
<tr>
<th>0, 0</th>
<th>1, 0</th>
<th>2, 0</th>
<th>3, 0</th>
<th>4, 0</th>
<th>5, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1</td>
<td>1, 1</td>
<td>2, 1</td>
<td>3, 1</td>
<td>4, 1</td>
<td>5, 1</td>
</tr>
<tr>
<td>0, 2</td>
<td>1, 2</td>
<td>2, 2</td>
<td>3, 2</td>
<td>4, 2</td>
<td>5, 2</td>
</tr>
<tr>
<td>0, 3</td>
<td>1, 3</td>
<td>2, 3</td>
<td>3, 3</td>
<td>4, 3</td>
<td>5, 3</td>
</tr>
<tr>
<td>0, 4</td>
<td>1, 4</td>
<td>2, 4</td>
<td>3, 4</td>
<td>4, 4</td>
<td>5, 4</td>
</tr>
<tr>
<td>0, 5</td>
<td>1, 5</td>
<td>2, 5</td>
<td>3, 5</td>
<td>4, 5</td>
<td>5, 5</td>
</tr>
</tbody>
</table>

y-axis

6 x 6 Pixels

“Line” drawn from: (0,0) to (3,5)

Getting Started with the Picture Class

1) Create a new, project with empty src and bin directories.
2) Download the Picture class from the class website and save it in the src directory of your new project.
3) Right-click on the project name and select “refresh”.
4) Right-click on the src directory of the new project and select: new→class. This is the class must contain the main method and is where you will add your own code.
Using Picture to Create an Empty JFrame

The Picture class is a utility class that is designed to be instantiated by other classes. Therefore, it does not need to have main.

The next few slides show how to use the Picture class from other classes.

//Create and display an empty Picture JFrame that is // 1000×500 pixels.
Picture p1 = new Picture(1000, 500);

//Picture extends JFrame. Therefore, all methods of // JFrame (i.e. setLocation are also methods of Picture.
p1.setLocation(100, 100);
p1.setTitle("Empty Picture");

My First Picture

import java.awt.Color;
import java.awt.Graphics;
public class FunWithDrawing
{
    public static void main(String[] args)
    {
        Picture myPic = new Picture(100,50);
        Graphics canvas = myPic.getOffScreenGraphics();

        canvas.setColor(Color.RED);
        canvas.drawLine(0, 0, 99, 49);

        canvas.setColor(Color.BLUE);
        canvas.drawLine(99, 0, 0, 49);

        myPic.repaint();
    }
}
import java.awt.Color;
import java.awt.Graphics;
public class DrawCommands
{
    public static void main(String[] args)
    {
        Picture myPic = new Picture(500, 500);
        Graphics canvas = myPic.getOffScreenGraphics();

        canvas.setColor(Color.WHITE);
        canvas.fillRect(0, 0, 500, 500);

        canvas.setColor(Color.BLUE);
        canvas.fillRect(200, 250, 100, 250);
        canvas.drawLine(250, 0, 250, 250);
        canvas.drawLine(251, 0, 251, 250);
        myPic.repaint();
    }
}

All drawing to canvas is done in an off-screen BufferedImage.
The repaint() method of the Picture class copies this off-screen BufferedImage to the screen.
There is no need to call repaint() after every draw command - unless you want an animation.
Calling repaint more than 60x/sec is a waste.
It is important to call repaint() after the last draw command.
Picture.repaint() Race Condition

- Java automatically calls repaint() whenever it detects the JFrame needs repainting. For example, when the JFrame is resized or when a window covering part of the JFrame is moved.

⚠️ **BUG:** If you only draw a few things and do not call repaint() your image may *usually* display correctly. When the frame first opens or a dialog covering part of the frame is closed, Java will call repaint. If your drawing is quick, it may *usually* be finished before Java calls repaint(). This type of bug is called a **race condition**.

- A race condition is when two or more events sometimes are scheduled in different orders AND when the desired results of those events are dependent on a particular order.

---

Quiz: drawOval from Textbook 1.4

```java
1) Picture myPic = new Picture(500,500);
2) Graphics canvas = myPic.getOffScreenGraphics();
3) // drawOval(x, y, width, height);
4) canvas.drawOval( 50,50,25,25);
5) canvas.drawOval(100,50,25,25);
6) canvas.drawOval(150,50,25,25);
7) canvas.drawOval(200,50,25,25);
8) myPic.repaint();
```

(a) (b) (c) (d)
**drawOval** \( (x, y, width, height) \)

1) \( \text{canvas.drawOval}(50, 50, 25, 25) \);

\[ \begin{array}{c}
\text{50} \\
\text{25} \\
\end{array} \]

**drawOval** is a **method** that requires **four arguments**.

Since \( width = height \), the "oval" is a circle.

2) \( \text{canvas.drawOval}(100, 50, 25, 25) \);

3) \( \text{canvas.drawOval}(150, 50, 25, 25) \);

4) \( \text{canvas.drawOval}(200, 50, 25, 25) \);

---

**Picture extends JFrame**

**Picture extends JFrame.**

Therefore all **JFrame** methods are available to myPic.

1) \( \text{Picture myPic = new Picture(250,200)} \);
2) \( \text{myPic.setTitle("Joel's Visual Masterwork")}; \)
3) \( \text{myPic.setLocation(100,0)} ; \)
4) \( \)
5) \( \text{Graphics canvas = myPic.getOffScreenGraphics()} ; \)
6) \( \)
7) \( \text{canvas.setColor(Color.WHITE)} ; \)
8) \( \text{canvas.fillRect(0, 0, 250, 200)} ; \)
9) \( \)
10) \( \text{canvas.setColor(Color.GREEN)} ; \)
11) \( \text{canvas.fillOval(100, 50, 150, 150)} ; \)
12) \( \)
13) \( \text{myPic.repaint();} \)
import java.awt.Color;
import java.awt.Graphics;
public class DrawCommands
{
    public static void main(String[] args)
    {
        Picture myPic = new Picture(500, 500);
        Graphics canvas = myPic.getOffScreenGraphics();
        Color periwinkle = new Color(204, 204, 255);
        Color pumpkin = new Color(255, 117, 24);
        canvas.setColor(periwinkle);
        canvas.fillOval(50, 50, 50, 50);
        canvas.setColor(pumpkin);
        canvas.fillOval(100, 50, 50, 100);
        myPic.repaint();
    }
}

drawOval(int left, int top, int width, int height)
fillOval(int left, int top, int width, int height)
drawArc(int left, int top, int width, int height, int startAngle, int arcAngle)
drawRect(int left, int top, int width, int height)
fillRect(int left, int top, int width, int height)
drawLine(int x1, int y1, int x2, int y2)
setColor(java.awt.Color myColor) ...and many more...
Using Picture to Display an Image

// Display a file chooser dialog for the user to select an image file. Then paints the image in a Picture.
Picture myPic = new Picture();

System.out.println("Outside Width = " + myPic.getWidth());

System.out.println("Inside Width = " + myPic.getImageWidth());

Output: Outside Width = 418
Inside Width = 400