

Lab 12 – Drawing using Java

EECS 1560 – Week of November 28, 2006

Description

In this lab you will take a simple application and add to it to produce graphics. We are going to draw a holiday scene. Download the Lights.java file from the course website (<http://www.eecs.utoledo.edu/~jheuring/eecs1560.html>) as a starting point.

Adding a paintComponent Method

First we will add a *paintComponent* method to the class. To start out with it will be a minimal class that does nothing but call the parent *JPanel* class to update the component. There are some declarations here that we will use later:

```
public void paintComponent ( Graphics g ) {
    Graphics2D g2 = (Graphics2D)g;
    Rectangle r;
    int x, currentColor = 0;
    final int LIGHT_HEIGHT = 15, LIGHT_WIDTH=10,
            LIGHT_SPACING = 25, SOCKET_WIDTH = 10,
            SOCKET_HEIGHT = 10;
    Color multiColors[] = new Color[ ]{Color.BLUE, Color.RED, Color.YELLOW,
                                         Color.WHITE, Color.MAGENTA};

    super.paintComponent(g2);
}
```

Note: **final** means the value cannot be changed. It is sometimes much easier to use "named" constants to make the code easier to read and modify.

Need to call the parent object before adding our own info. Super says to use the class above this.

Run the program. You should have a window that comes up with 4 buttons used to set the color of the lights.

Adding Lights

We will deal with the lights first. I want to put a string of lights at the bottom of the window. However, I don't know where the bottom of the window is so we must first get the dimensions of the window. We are going to do that with the *getClipBounds()* method which returns a *Rectangle*.

After the *super.paintComponent(g2);* add:

```
r = g2.getClipBounds();
for ( x = (int) r.getMinX() + LIGHT_WIDTH;
      x < r.getMaxX();
      x = x + LIGHT_SPACING ) {
    g2.setColor(lightColor);
    g2.fillOval(x, r.getHeight() - (LIGHT_HEIGHT + SOCKET_HEIGHT),
               LIGHT_WIDTH, LIGHT_HEIGHT);
    g2.setColor(Color.GREEN);
    g2.fillRect(x, r.getHeight() - SOCKET_HEIGHT, SOCKET_WIDTH, SOCKET_HEIGHT);
}
```

Run the resulting program. It should put a row of lights along the bottom of the window. Try to resize the window – the reason we got the window size with the *getClipBounds* call was so that we would draw as many lights as were needed. If you use the buttons the color of the lights should change (except for multi – that's next).

We are going to change how the color for the light is determined in the *paintComponent* method. The line:

```
g2.setColor(lightColor)
```

needs to be *followed* with a block of code that will cycle through some light colors. We can do this by using the array *multiColors* and a counter (*currentColor*). The new code will become:

```
if (multiFlag) {
    g2.setColor(multiColors[currentColor++]);
    if (currentColor >= multiColors.length) {
        currentColor = 0;
    }
}
```

Type this in and run the program. The multicolor button should now work as you would expect it to. Next, we will deal with a snowman and trees.

Building a Tree

We will set up a draw tree method that will draw a tree with a given height and width and at a given location. This isn't as easy as simply drawing a tree in one location but it is a bit more useful. The main new item is the idea of filling a polygon. A polygon is given as two arrays of integers. The first array consists of the x coordinates for the vertices of the polygon and the second array is the y coordinates. The third parameter is the number of points contained in the two arrays.

```
public void drawTree(Graphics g, int xPosition, int yPosition,
                    int width, int height) {
    Graphics2D g2 = (Graphics2D) g;
    /*
     * These are ratios for drawing the tree. They are a
     * fraction of the height and width of the tree. We are
     * actually a bit big because I forgot about the trunk
     * height....
     */
    double xRatio[] = new double[] { 0.5, 0.0, 5.0/12.0,
                                      1.0/12.0, 5.0/12.0, 1.0/6.0, 0.5,
                                      5.0/6.0, 7.0/12.0, 11.0/12.0, 7.0/12.0,
                                      1.0, 0.5};
    double yRatio[] = new double[] { 1.0, 1.0, 0.67, 0.67,
                                      0.33, 0.33, 0.0, 0.33, 0.33, 0.67,
                                      0.67, 1.0, 1.0};
    int x[] = new int [xRatio.length];
    int y[] = new int [yRatio.length];
    int i;
    /*
     * Figure out the integer positions for the tree points.
     * Each vertex needs to be given for a polygon...
     */
    for (i = 0; i < x.length; i++) {
        x[i] = (int) Math.round(width * xRatio[i] + xPosition);
        y[i] = (int) Math.round(height * yRatio[i] + yPosition);
    }
    /*
     * Draw the tree and its trunk.
     */
    g2.setColor(Color.GREEN);
    g2.fillPolygon(x, y, x.length);
    g2.setColor(Color.DARK_GRAY);
    g2.fillRect(Math.round(0.4f*width + xPosition),
                Math.round(height+yPosition),
                Math.round(0.2f * width), Math.round(0.2f * height));
}
```

Note: Do you know what the 0.4f and the 0.2f's represent in the method above? This is one reason to use named constants!

Now, add the following three lines to the end of your *paintComponent* method:

```
drawTree(g2, 100, 100, 40, 100);
drawTree(g2, 50, 100, 50, 50);
drawTree(g2, 200, 75, 100, 100);
```

Compile and run your project. You should now have the same scene as before but with three “trees” added to it at different positions.

Adding a Snowman

We will now add a snowman to our scene. It will be less useful than the one for *drawTree* because we will set it up to draw the snowman in one spot and with one size. The *drawSnowman* method becomes:

```
public void drawSnowMan(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    int x[] = new int[]{327, 320, 327};
    int y[] = new int[]{86, 95, 90};
    /*
     * Draw the three white Snowballs
     */
    g2.setColor(Color.WHITE);
    g2.fillOval(300, 140, 60, 60); // Three snow balls for the body...
    g2.fillOval(310, 100, 40, 40);
    g2.fillOval(317, 75, 25, 25);
    g2.setColor(Color.BLACK);
    g2.fillOval(327, 105, 5, 5); // Three buttons down the middle of the
    g2.fillOval(327, 115, 5, 5); // center snowball for buttons.
    g2.fillOval(327, 125, 5, 5);
    g2.fillOval(322, 82, 5, 5); // Two buttons for eyes.
    g2.fillOval(332, 82, 5, 5);
    g2.setColor(Color.ORANGE); // and an orange triangle for the nose.
    g2.fillPolygon(x, y, x.length);
}
```

Add a line to the *paintComponent* routine to put in the snowman:

```
drawSnowMan(g2);
```

Now run your program. There should be a snowman on the right hand side.
E-mail me your *Lights.java* file (gheuring@eng.utoledo.edu).