Graphics in Java

Overview

This tutorial gives you the bare minimum you need to paint basic geometric shapes in Java using the `Graphics2D` functions.

Graphics Context and the Basic Shapes

Everything you will paint will be done through member functions of the class `Graphics2D`. The `BufferedPanel window` used in our applications has a member function `GetBufferGraphics` that returns its `Graphics2D` object - so called graphics context:

You see the following statement at the beginning of the functions that do the painting in the given `window`:

```
Graphics2D G = window.getBufferGraphics();
```

Next, you need to define at least one object for each of the three shapes (`Line`, `Ellipse`, `Rectangle`) you want to paint. You do this as follows:

```
Rectangle2D.Double R = new Rectangle2D.Double;
Ellipse2D.Double E = new Ellipse2D.Double;
Line2D.Double L = new Line2D.Double;
```

Each of these shapes also has a `Float` version, but we will not use them. The `.Double` indicates that the coordinates and sizes of these shapes are given as `double`.

To set the location and size of each object, we use the following statements:

```
R.setRect(left, top, width, height);
E.setFrame(left, top, width, height);
L.setLine(x1, y1, x2, y2);
```

Notice, that to set the location and size of an oval (ellipse), we specify the size of the rectangular frame around the ellipse.

Now we can actually paint these shapes as follows:

```
G.fill(R);
G.fill(E);
G.draw(L);
```

In addition, we can use the draw function with `Ellipse` or `Rectangle` to draw just the outlines of these shapes.

Colors

To set the colors, use the `Graphics2D` member function `setPaint()` in one of the following two forms:

```
// standard issue red
G.setPaint(Color.red);

// pale blue that you choose
G.SetPaint(new Color(100, 100, 255);
```

The `Color` class defines the following color constants:

Name	Equivalent
`Color.white`	`new Color(255, 255, 255)`
`Color.lightGray`	`new Color(192, 192, 192)`
`Color.gray`	`new Color(128, 128, 128)`
`Color.darkGray`	`new Color(64, 64, 64)`
`Color.black`	`new Color(0, 0, 0)`
`Color.red`	`new Color(255, 0, 0)`
`Color.pink`	`new Color(255, 175, 175)`
`Color.orange`	`new Color(255, 200, 0)`
`Color.yellow`	`new Color(255, 255, 0)`
`Color.green`	`new Color(0, 255, 0)`
`Color.magenta`	`new Color(255, 0, 255)`
`Color.cyan`	`new Color(0, 255, 255)`
`Color.blue`	`new Color(0, 0, 255)`

Paint modes

There are many tricks you can play with colors - we include only one that is used in the Target task of the Loops lab. In this lab, we are painting progressively larger circles, but want to reverse colors on all the existing circles. So we start with one red circle. The larger second circle is show in red, but the part that was the original first circle is converted to the original background color, which was white. When the third circle is painted, the inside turns red again, the middle band turns to white and the area outside of the previous two circles turns red.

To do this we started the task with the call

```
G.setXORMode(Color.white);
```

and reset to the original mode at the end by calling:

```
G.setPaintMode();
```

Fancy stuff

In addition there is a round rectangle and an arc.

The window in the Locomotive painting is a rounded rectangle, defined and set as follows:

```
RoundRectangle2D.Double Rd
            = new RoundRectangle2D.Double();
Rd.setRoundRect(x + 0.65 * width, y + 0.15 * height,
                0.2 * width,      0.3 * height,
                0.1 * width,      0.1 * height);
```

The last two values in the `setRoundRect` function specify the width and the height of the rounded arc. The coordinates still specify the whole rectangle - without the corners rounded off.

The `Arc` is a part of an ellipse. Again, you need to set the `Arc` by specifying the location and size of the frame enclosing the ellipse from which the arc is "cut out", then specify the starting and ending angle (in degrees, though this is a bit tricky and we will not explain it here). Finally, you need to select the method for cutting out the piece of the pie from the ellipse. To do so, use one of the following constants as the last argument for the `setArc` function: `OPEN`, `PIE`, `CHORD`. The first one will give you just a curved line, the second will give you a pie piece, and the third one will connect the two end points of the arc with a straight line:

```
Arc2D.Double A = new Arc2D.Double();
A.setArc(left, top, width, height, 0, 90, PIE);
G.fill(A);
```

The code shown above will give you the top right quarter of the ellipse.

Reference

If you want to learn more about the use of Graphics2D class and creating graphics, the best reference is:

Jonathan Knudsen, Java 2D Graphics, O'Reilly, 1999.