7.17 (Optional) GUI and Graphics Case Study: Drawing Arcs

In this section, we build a **Draw Rainbow** app that uses arrays and iteration statements to draw a rainbow with `GraphicsContext` method `fillArc`. Drawing arcs in JavaFX is similar to drawing ovals—an arc is simply a section of an oval.

Creating the `Draw Rainbow` App's GUI

The app's GUI is defined in `DrawRainbow.fxml`. We reused the `DrawLines.fxml` GUI shown in Fig. 4.17, but made the following changes to `DrawRainbow.fxml`:

- We specified `DrawRainbowController` as the app's **Controller class** in the Scene Builder **Document** window's **Controller** section.
- We set the `Button`'s **Text** property to `Draw Rainbow` (in the **Inspector**'s **Properties** section) and set the `Button`'s **On Action** event handler to `drawRainbowButtonPressed` (in the **Inspector**'s **Code** section).
- We changed the `Canvas`'s **Width** and **Height** properties (in the **Inspector**'s **Layout** section) to `400` and `200`, respectively.

Class `DrawSmileyController`

Again, we do not show the code for `DrawRainbow.java`, because the only changes from earlier examples are the name of the FXML file to load (`DrawRainbow.fxml`) and the text displayed in the stage's title bar (`Draw Rainbow`).

In Fig. 7.25, lines 16–17 use various predefined `Color`s to initialize an array with the colors of the rainbow, starting with the innermost arcs first. The array begins with two `Color.WHITE` elements, which, as you'll soon see, are for drawing the empty arcs at the center of the rainbow.

```
 1  // Fig. 7.25: DrawRainbowController.java
 2  // Drawing a rainbow using arcs.
 3  import javafx.event.ActionEvent;
 4  import javafx.fxml.FXML;
 5  import javafx.scene.canvas.Canvas;
 6  import javafx.scene.canvas.GraphicsContext;
 7  import javafx.scene.paint.Color;
 8  import javafx.scene.shape.ArcType;
 9
10  public class DrawRainbowController {
11     @FXML private Canvas canvas;
12
13     // colors to use in the rainbow, starting fro
14     // The two white entries result in an empty a
15     private final Color[] colors = {
16        Color.WHITE, Color.WHITE, Color.VIOLET, Co
17        Color.GREEN, Color.YELLOW, Color.ORANGE, C
18
19     // draws a rainbow using arcs
20     @FXML
```

```
21      void drawRainbowButtonPressed(ActionEvent eve
22         // get the GraphicsContext, which is used
23         GraphicsContext gc = canvas.getGraphicsCon
                      24
25         final int radius = 20; // radius of an arc
                      26
27         // draw the rainbow near the bottom-center
28         final double centerX = canvas.getWidth() /
29         final double maxY = canvas.getHeight() - 1
                      30
31         // draws filled arcs starting with the out
32         for (int counter = colors.length; counter
33            // set the color for the current arc
 34              gc.setFill(colors[counter - 1]);
                      35
36            // fill the arc from 0 to 180 degrees
37            gc.fillArc(centerX - counter * radius,
38               maxY - counter * radius, counter * r
39               counter * radius * 2, 0, 180, ArcTyp
                  40         }
                    41      }
                      42   }
```

Fig. 7.25

Drawing a rainbow using arcs.

Description

Line 25 in method `drawRainbowButtonPressed` declares local variable `radius`, which determines each arc's radius. Local variables `centerX` and `maxY` (lines 28–29) determine the location of the midpoint on the rainbow's base. Lines 32–40 use control variable `counter` to count backwards from the array's elements, drawing the largest arcs first and placing each successive smaller arc on top of the previous one. Line 34 uses an element from the `colors` array to set the fill color for the current arc. The two `Color.WHITE` entries at the beginning of the array create the empty white arc in the center. You can change the individual colors and the number of entries in the array to create new designs.

The `fillArc` method call at lines 37–39 draws a filled semicircle. Method `fillArc` requires six parameters. The first four represent the bounding rectangle in which the arc will be drawn. The first two of these specify the coordinates for the bounding rectangle's upper-left corner, and the next two specify its width and height. The fifth parameter is the starting angle on the oval, and the sixth specifies the **sweep**, or the amount of arc to cover. The starting angle and sweep are measured in degrees, with zero degrees pointing right. A *positive* sweep draws the arc *counterclockwise*, while a *negative* sweep draws the arc *clockwise*. The last argument

—`ArcType.OPEN`—specifies that the arc's endpoints should not be connected with a line. You can see the other `ArcType` options at

```
https://docs.oracle.com/javase/8/javafx/api/javafx/sc
```

Method `strokeArc` requires the same parameters as `fillArc`, but draws the edge of the arc rather than filling it.

GUI and Graphics Case Study Exercises

7.1 ***(Drawing a Spiral with Lines)*** Draw a square-shaped spiral (Fig. 7.26), centered on the `Canvas`, using method `strokeLine`. One technique is to use a loop that increases the line length after drawing every second line. The direction in which to draw the next line should follow a distinct pattern, such as down, left, up, right.

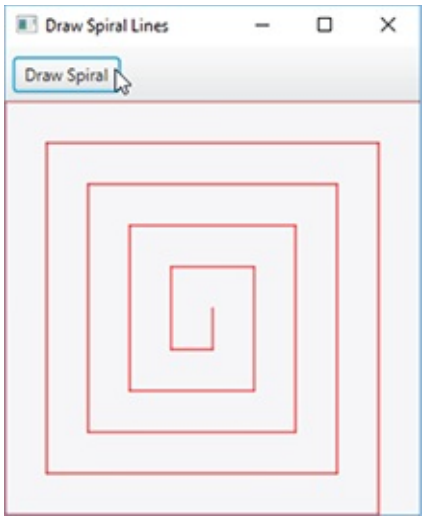

Fig. 7.26

Drawing a spiral using `strokeLine`.

7.2 ***(Drawing a Spiral with Arcs)*** Draw a circular spiral (Fig. 7.27), using method `strokeArc` to draw one semicircle at a time. Each successive semicircle should have a larger radius (as specified by the bounding rectangle's width) and should continue drawing where the previous semicircle finished.

Fig. 7.27

Drawing a spiral using `strokeArc`.

Description

7.3 ***(Rotating Spiral)*** Enhance Exercise 7.2 so that the spiral rotates slightly on each `Button` click.

7.4 ***(Color Changing Spiral)*** Enhance Exercise 7.2 so that each arc segment in the spiral uses a slightly brighter version of the previous `Color`. Start with a dark `Color`, then use the `Color` method `brighter` to create the next segment's `Color`.

7.5 ***(Enhanced Rainbow)*** Enhance the **Draw Rainbow** app to use thinner arcs and to transition through additional color variations to create a more realistic looking rainbow.