

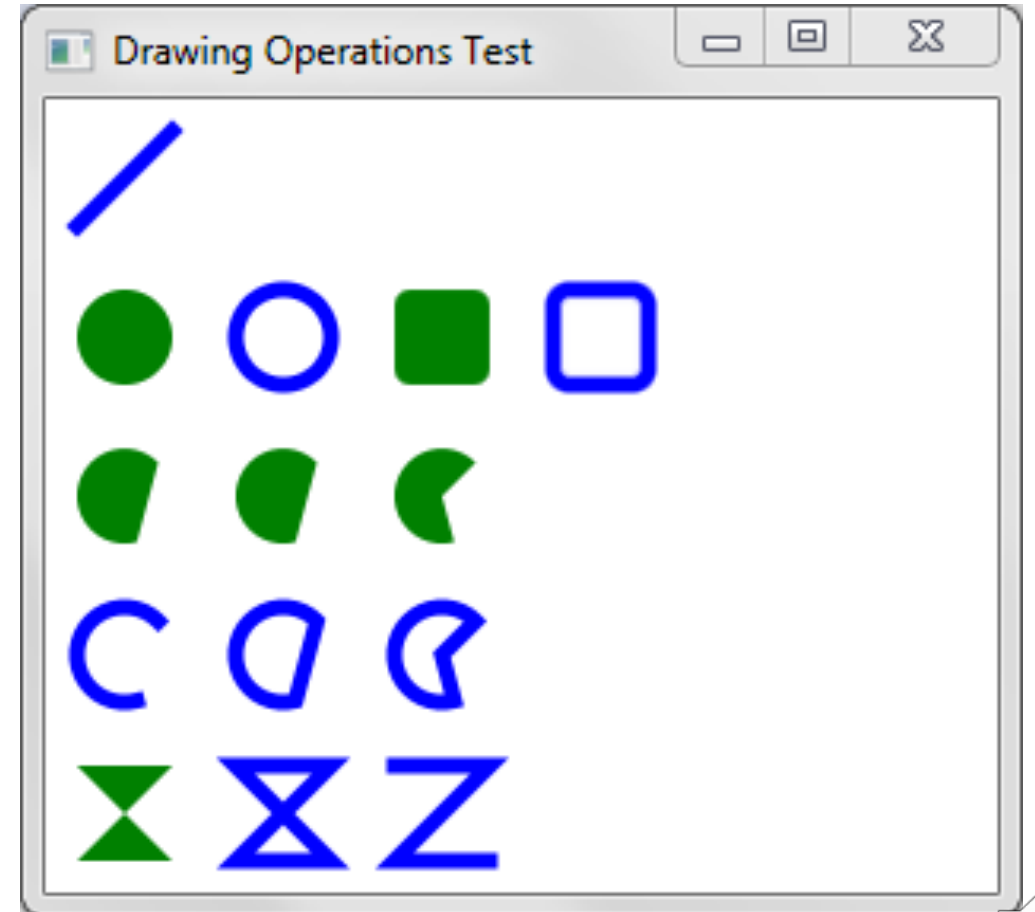
Graphics

Week 7 – Presentation 2



2D Graphics


Charts are 2D graphics (you also have 3D charts but if you ever read Tufte you'll never want to use them), but in charts you haven't full freedom to draw whatever you want on the screen. If you want to draw you should you a Canvas object. "Canvas" was the name of the cloth used in the old days for making ship sails. Put on a wooden frame, this is what western artists started to use around the 17th century for painting, hence the name in graphical interfaces.



Canvas Object

```
import javafx.scene.*;  
import javafx.scene.paint.*;  
import javafx.scene.canvas.*;
```

- Lines and Shapes

An easel with a blank canvas. The easel is made of wood and has a large, empty rectangular canvas attached to it. The canvas is white and has a thin black border. The easel has a horizontal bar across the middle and two legs at the bottom. A small wooden clip is attached to the top of the canvas.

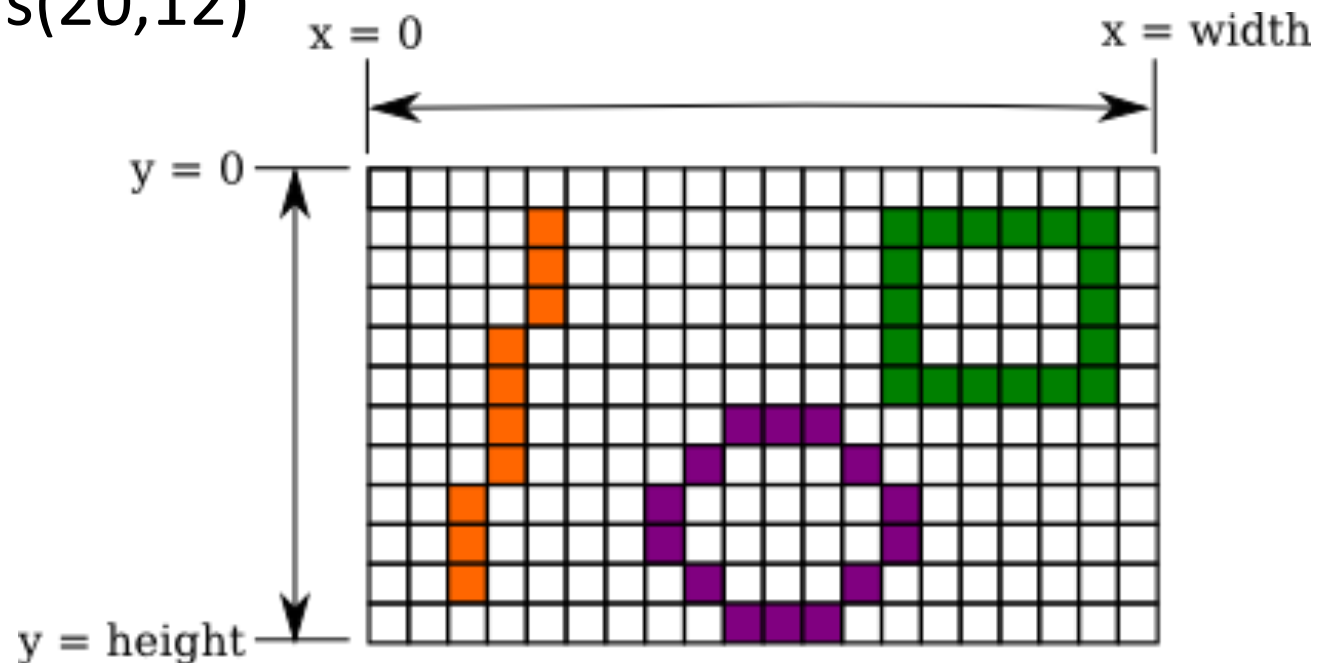
On a canvas, you mostly draw
lines and shapes.



Canvas Object

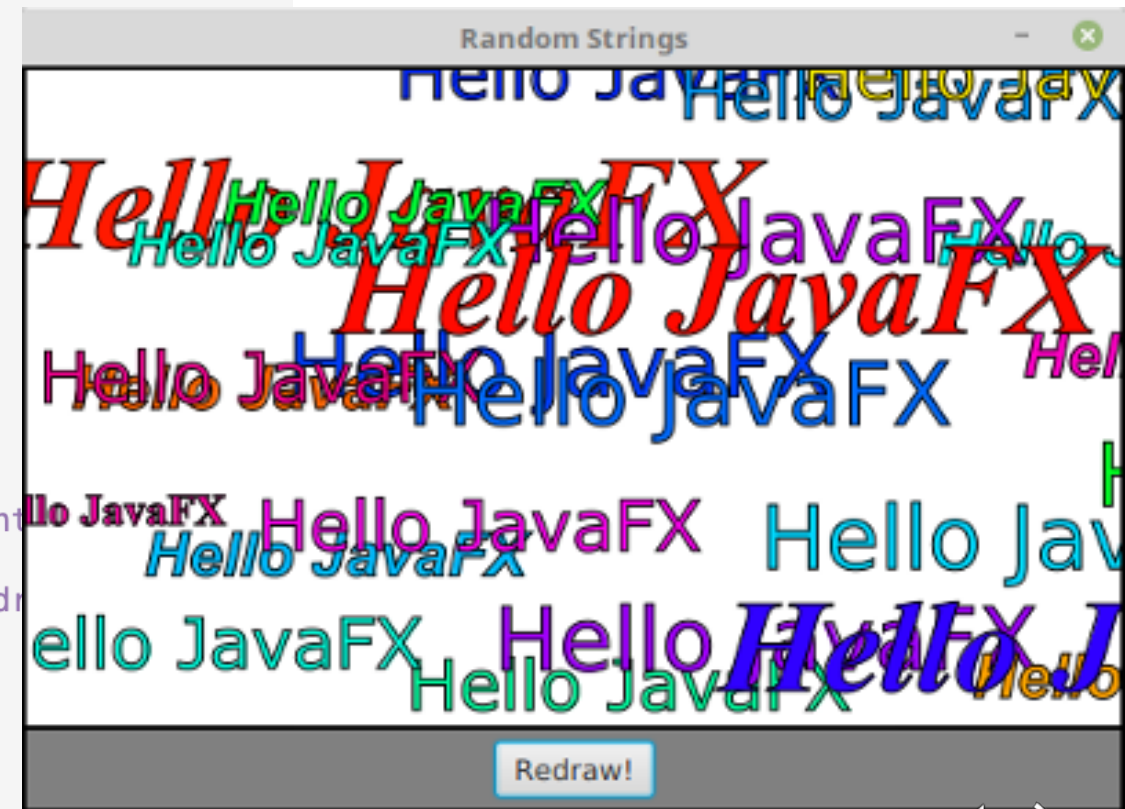
The width and height of a *Canvas* can be specified in the constructor that used to create the canvas object. For example, to create a tiny 20-by-12 canvas:

Canvas canvas = new Canvas(20,12)



RandomStrings.java

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.scene.layout.BorderPane;
5  import javafx.scene.layout.StackPane;
6  import javafx.stage.Stage;
7  import javafx.scene.text.*;
8  import javafx.scene.canvas.*;
9  import javafx.scene.paint.Color;
10
11  /**
12   * This program displays 25 copies of a message. The color and
13   * position of each message is selected at random. The font
14   * of each message is randomly chosen from among five possible
15   * fonts. The messages are displayed on a white background.
16   * There is a button that the user can click to redraw the
17   * image using new random values.
18   */
19  public class RandomStrings extends Application {
20
21      private final static String MESSAGE = "Hello JavaFX";
22
23      private Font font1, font2, font3, font4, font5; // The five fonts
24
25      private Canvas canvas; // The canvas on which the strings are drawn
26
27      public static void main(String[] args) {
28          launch(args);
29      }
30  }
```



```

33 public void start( Stage stage ) {
34
35     font1 = Font.font("Times New Roman", FontWeight.BOLD, 20);
36     font2 = Font.font("Arial", FontWeight.BOLD, FontPosture.ITALIC, 28);
37     font3 = Font.font("Verdana", 32);
38     font4 = Font.font(40);
39     font5 = Font.font("Times New Roman", FontWeight.BOLD, FontPosture.ITALIC, 6
40
41     canvas = new Canvas(500,300);
42     draw(); // draw content of canvas the first time.
43
44     Button redraw = new Button("Redraw!");
45     redraw.setOnAction( e -> draw() );
46
47     StackPane bottom = new StackPane(redraw);
48     bottom.setStyle("-fx-background-color: gray; -fx-padding:5px;" +
49                    "-fx-border-color:black; -fx-border-width: 2px 0 0 0")
50     BorderPane root = new BorderPane(canvas);
51     root.setBottom(bottom);
52     root.setStyle("-fx-border-color:black; -fx-border-width: 2px");
53
54     stage.setScene( new Scene(root, Color.BLACK) );
55     stage.setTitle("Random Strings");
56     stage.setResizable(false);
57     stage.show();
58 }
59

```



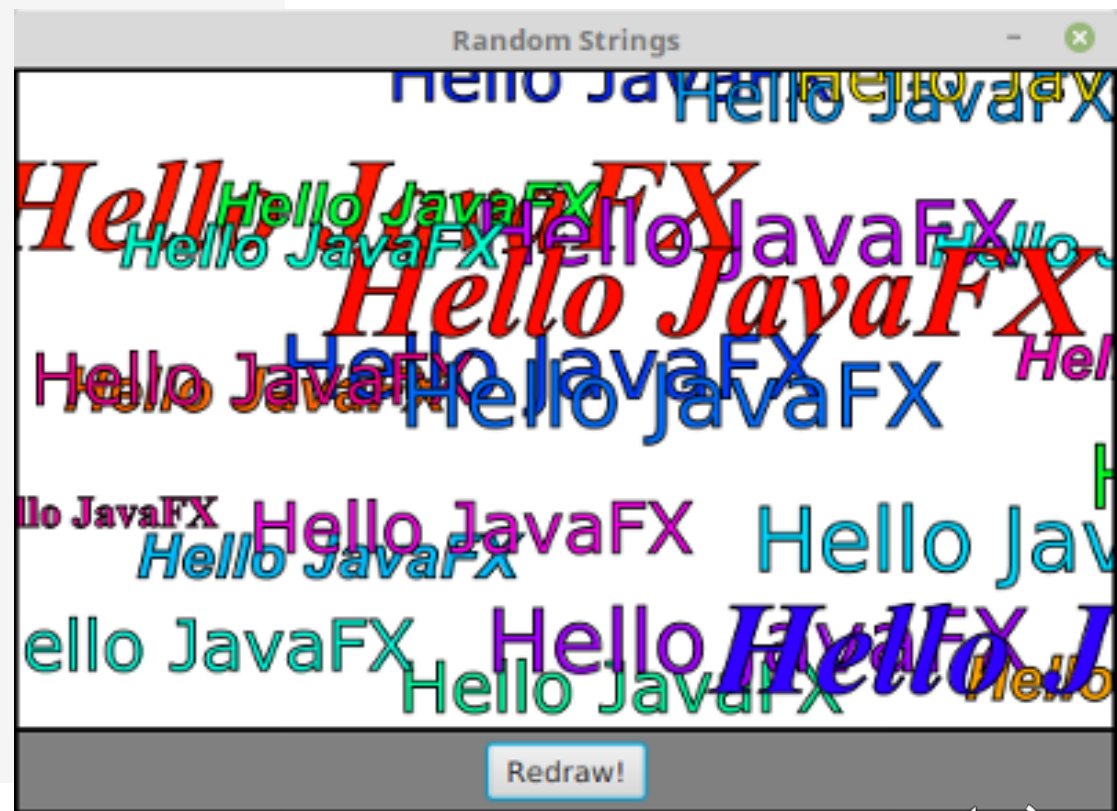
```

60
61  /**
62   * The draw() method is responsible for drawing the content of the canvas.
63   * It draws 25 copies of the message string, using a random color, font, and
64   * position for each string.
65   */
66  private void draw() {
67
68      GraphicsContext g = canvas.getGraphicsContext2D();
69
70      double width = canvas.getWidth();
71      double height = canvas.getHeight();
72
73      g.setFill( Color.WHITE ); // fill with white background
74      g.fillRect(0, 0, width, height);
75
76      for (int i = 0; i < 25; i++) {
77
78          // Draw one string. First, set the font to be one of the five
79          // available fonts, at random.
80
81          int fontNum = (int)(5*Math.random()) + 1;
82          switch (fontNum) {
83              case 1:
84                  g.setFont(font1);
85                  break;
86              case 2:
87                  g.setFont(font2);
88                  break;
89              case 3:
90                  g.setFont(font3);
91                  break;
92              case 4:
93                  g.setFont(font4);
94                  break;
95              case 5:
96                  g.setFont(font5);
97                  break;
98          } // end switch
99
00          // Set the color to a bright, saturated color, with random hue.
01
02          double hue = 360*Math.random();
03          g.setFill( Color.hsb(hue, 1.0, 1.0) );
04

```

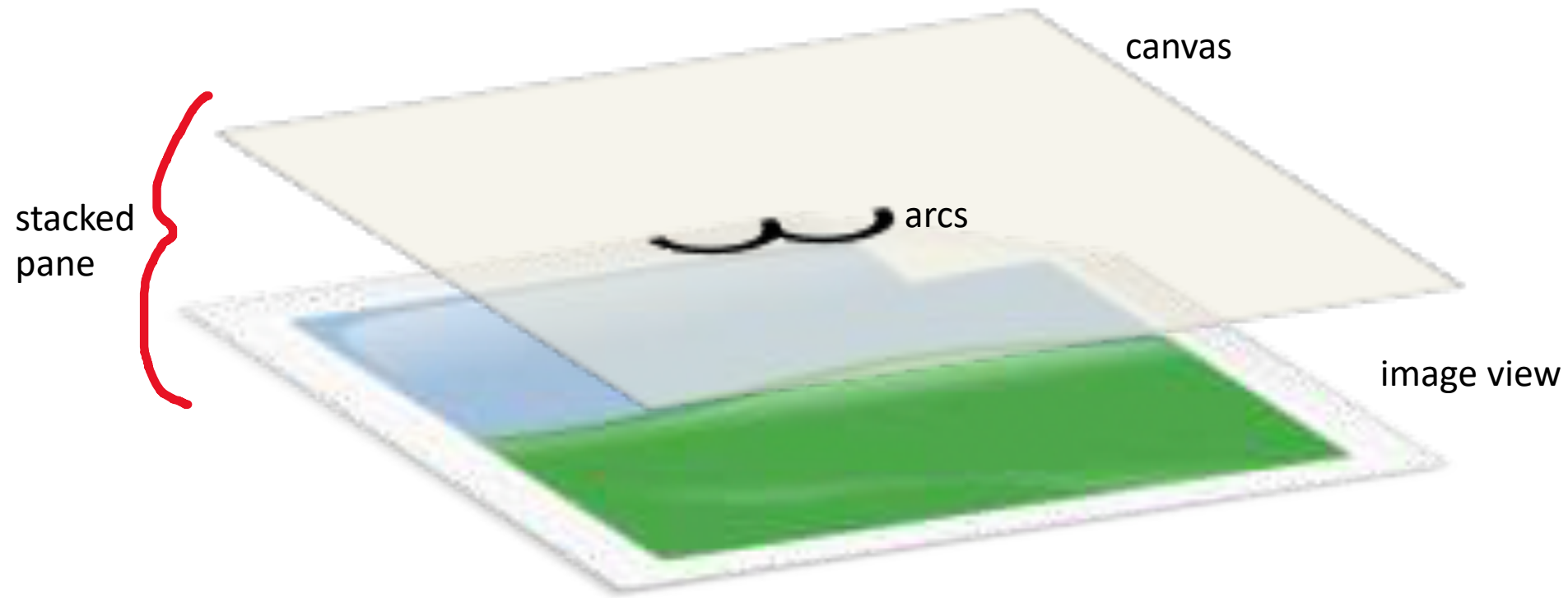


```
99 // Set the color to a bright, saturated color, with random hue.
100
101 double hue = 360*Math.random();
102 g.setFill( Color.hsb(hue, 1.0, 1.0) );
103
104 // Select the position of the string, at random.
105
106 double x,y;
107 x = -50 + Math.random()*(width+40);
108 y = Math.random()*(height+20);
109
110 // Draw the message.
111
112 g.fillText(MESSAGE,x,y);
113
114 // Also stroke the outline of the strings with black
115
116 g.setStroke(Color.BLACK);
117 g.strokeText(MESSAGE,x,y);
118
119 } // end for
120
121 } // end draw()
122
123
124
```



Another example:

Take a background image, and Canvas on which to draw two half circles close to each other (part-circles shapes are called "arc")



Import packages

```
1  import javafx.application.Application;
2  import javafx.event.ActionEvent;
3  import javafx.event.EventHandler;
4  import javafx.scene.*;
5  import javafx.scene.layout.*;
6  import javafx.scene.paint.*;
7  import javafx.scene.canvas.*;
8  import javafx.scene.shape.*;
9  import javafx.stage.Stage;
10 import javafx.stage.Screen;
11 import javafx.scene.image.*;
12 import java.net.URL;
13
14 public class StupidCanvasExample extends Application {
15     public static void main(String[] args) { launch(args);
16
17
18 }
```



```
19
20 public void start(Stage stage) {
21     double width;
22     double height;
23     double x;
24     double y;
25
26     [ stage.setTitle("StupidCanvasExample");
27       stage.setResizable(false);
28       Group root = new Group();
29       Scene scene = new Scene(root);
30       StackPane pane = new StackPane();
31
32       URL url = this.getClass().getClassLoader()
33                 .getResource("background.jpeg");|
```

StackPane to put image and Canvas (transparent) on top of it.



```
33         .getResource("background.jpeg");
34
35     if (url != null) {
36         Image image = new Image(url.toString());
37         width = image.getWidth();
38         height = image.getHeight();
39         ImageView iv = new ImageView(image); pane.getChildren().add(iv);
40         final Canvas canvas = new Canvas(width, height);
41         GraphicsContext gc = canvas.getGraphicsContext2D();
```

To draw on a Canvas, you need the associated "GraphicsContext". This is where you define, among other things, line thickness and colours.



```
40 final Canvas canvas = new Canvas(width, height);
41 GraphicsContext gc = canvas.getGraphicsContext2D();
42
43 gc.setStroke(Color.BLACK);
44 gc.setLineWidth(height * 0.01);
45 x = 0.42 * width - width / 36.0;
46 gc.strokeArc(x, y,
47             width / 18.0, height / 40.0,
48             180, 180, ArcType.OPEN);
49 pane.getChildren().add(canvas);
50 // Make canvas disappear when clicked
51 canvas.setOnMouseClicked((e)->{ canvas.setVisible(false);});|
52
```

There are multiple ways to define colors,
here we use an enum.

Make the canvas invisible when clicked



```
49     pane.getChildren().add(canvas);
50     // Make canvas disappear when clicked
51     canvas.setOnMouseClicked((e)->{ canvas.setVisible(false);});
52
53 }
54
55 root.getChildren().add(pane);
56 stage.setScene(scene); stage.show();
57 }
58 }
59
60
61
```

And there you go. All the art, of course, is in the choice of the suitable background image

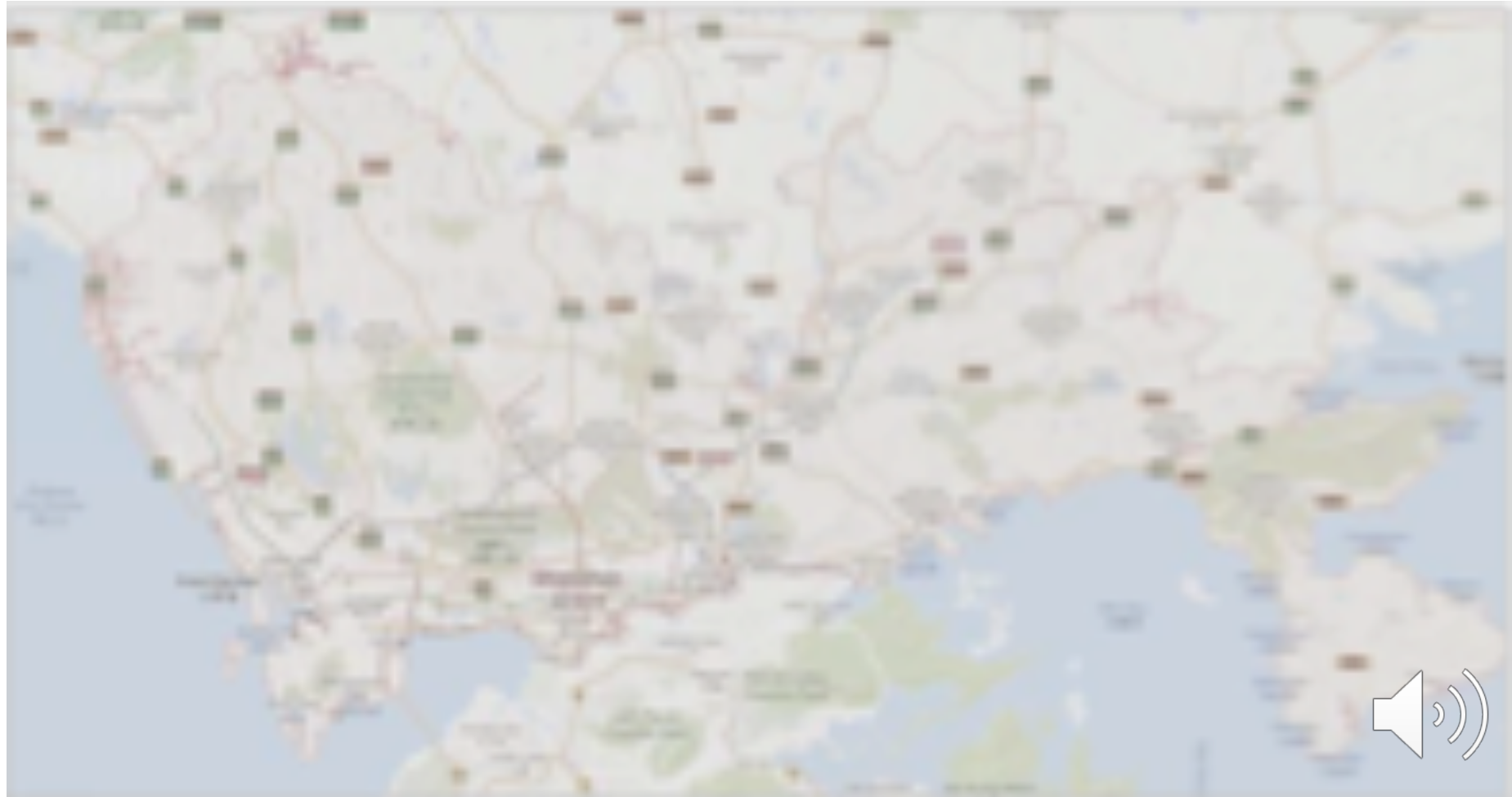


If instead of using Mona Lisa you use a map, you can create some really interesting applications with canvases (other than a drawing tool).

Use a map as a background

Draw routes

You could have for instance one Canvas per Metro line, stack all of them, and use buttons to make a line appear or disappear. That said, working with maps is not very easy because what you want to plot are usually places for which you know latitude and longitude.



Interaction?

Shape Objects:

- Arc
- Circle
- Line
- Polygon
- Rectangle
- Text
- ...

You can only interact with a "Node". A Canvas is a node, and you can interact with it (I was able to make the Canvas over Mona Lisa invisible by clicking on it). However, you cannot directly interact with the shapes drawn over the canvas using graphics context. If you want to do this you need shape objects.



Using shape objects...

Stroke

Color `.setStroke(col)`

Width `.setStrokeWidth(w)`

+ other properties



You can set for them what you can set in the GraphicsContext of a Canvas (note that the Width of a Stroke is the line thickness)



Stroke

Color `.setStroke(col)`

Width `.setStrokeWidth(w)`

+ other properties

Fill:

color,
gradient,
image,
pattern



You can fill shapes too,
colors can also be specified by
an intensity of red, green, and
blue, as well as a parameter
that specifies transparency.

`Color.rgb(255, 255, 0, 1.0)`



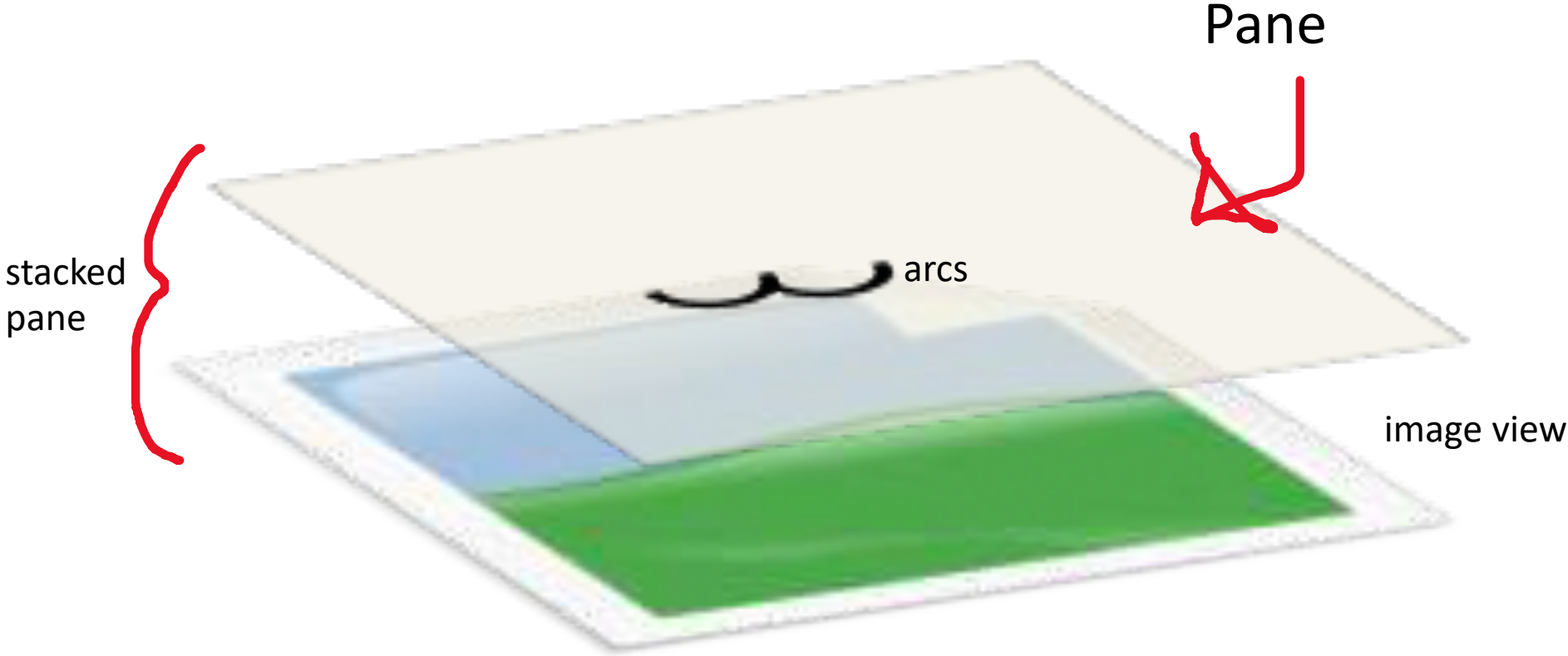
Shapes are nodes

CLICKABLE

With shapes you can click on any individual shape.



We can redo the Monalisa with shapes instead of a Canvas...



```
43 Arc arc = new Arc();
44 arc.setCenterX(0.42 * width);
45 arc.setCenterY(0.288 * height);
46 arc.setRadiusX(width / 36.0);
47 arc.setRadiusY(width / 36.0);
48 arc.setStartAngle(180.0);
49 arc.setLength(180.0);
50 arc.setType(ArcType.OPEN);
51 arc.setStroke(Color.BLACK);
52 arc.setStrokeWidth(height * 0.01);
53 arc.setFill(Color.rgb(255, 255, 255, 0.0));
54 shapePane.getChildren().add(arc);
55
```

Same for arc2...



```
57  ▼ arc.setOnMouseClicked((e)->{  
58      Random rand = new Random();  
59      int r = rand.nextInt(256);  
60      int g = rand.nextInt(256);  
61      int b = rand.nextInt(256);  
62      Color col = Color.rgb(r, g, b);  
63      arc.setStroke(col);  
64      arc2.setStroke(col);  
65
```

Here the same action is associated with arc and arc2



Canvas or Shapes

- The choice depends how many shapes there are
- If too many shapes will become slow and difficult to check if you clicked in the right place...
- Note: its also possible to check where a canvas was clicked:
 mouseEvent .getX() / .getY()



3D Graphics

- JavaFX can also render 3D graphics
e.g. `javafx.scene.shape.Shape3D`

I won't talk about 3D graphics because it's beginning to become very specific to advanced applications. Let's just say that you have packages in JavaFX for 3D graphics as well.

