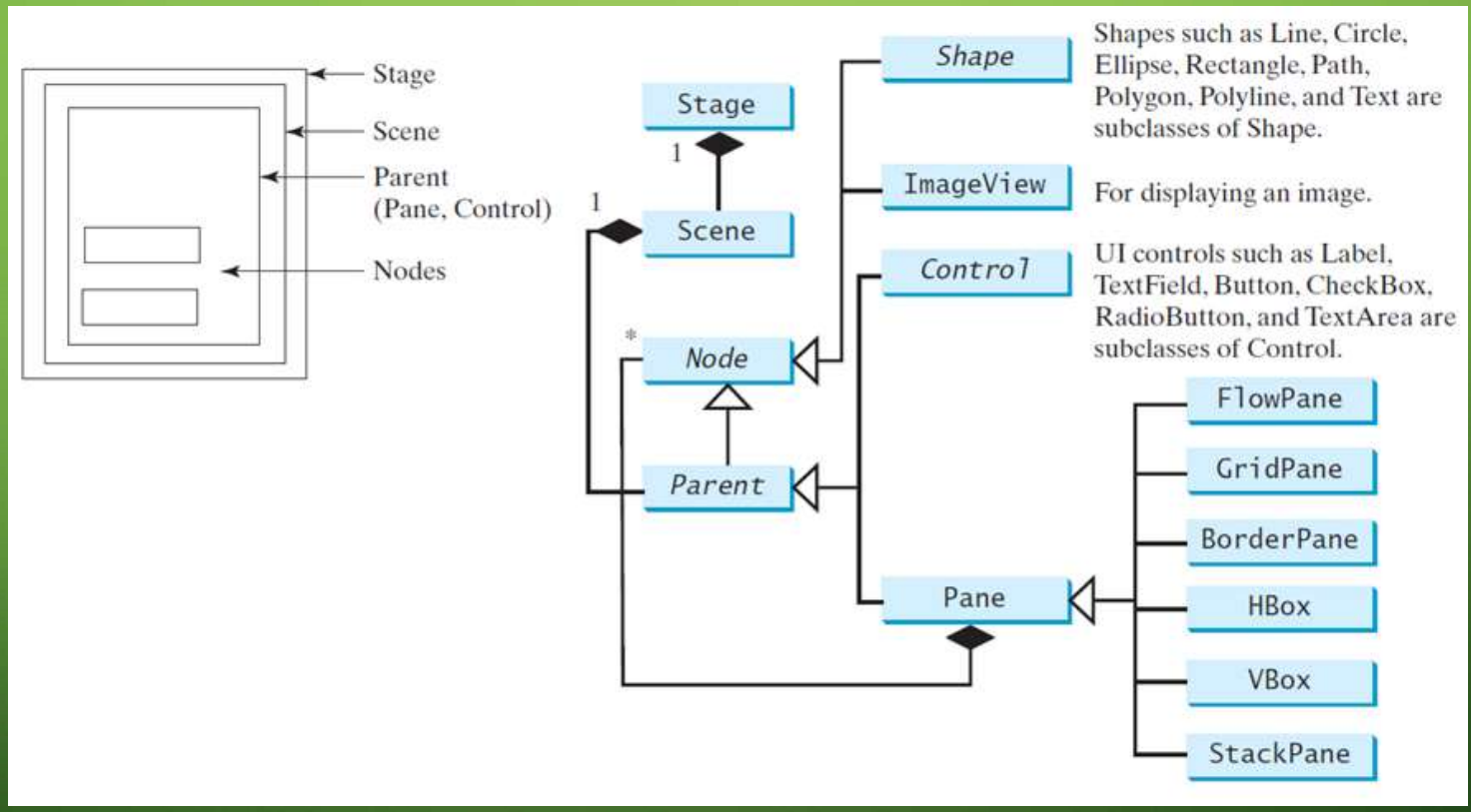
CSE 219
COMPUTER SCIENCE III

GRAPHICS2D WITH JAVAFX

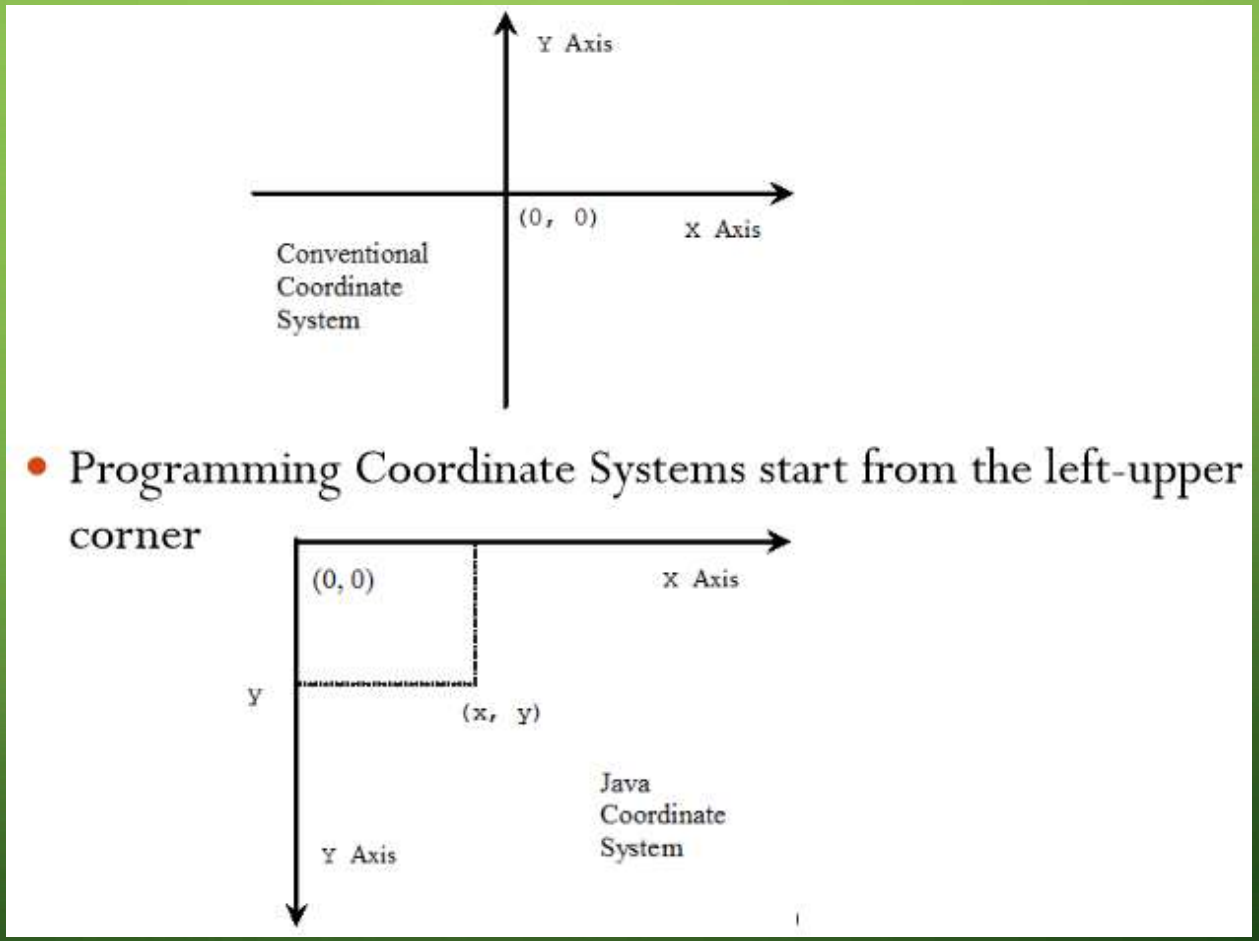
SLIDES COURTESY:

RICHARD MCKENNA, STONY BROOK UNIVERSITY.

PANES, UI CONTROLS, AND SHAPES



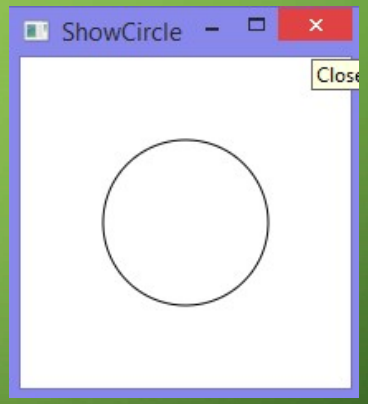
DISPLAY A SHAPE



- Programming Coordinate Systems start from the left-upper corner

A CIRCLE IS A SHAPE

```
public class ShowCircle extends Application {
    @Override
    public void start(Stage primaryStage) {
        Circle circle = new Circle();
        circle.setCenterX(100);
        circle.setCenterY(100);
        circle.setRadius(50);
        circle.setStroke(Color.BLACK);
        circle.setFill(null);
        Pane pane = new Pane();
        pane.getChildren().add(circle);
        Scene scene = new Scene(pane, 200, 200);
        primaryStage.setTitle("ShowCircle");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

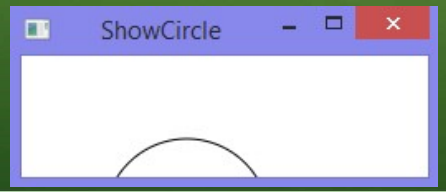


BINDING PROPERTIES

JavaFX introduces a new concept called a ***binding property*** that enables a target object to be bound to a source object

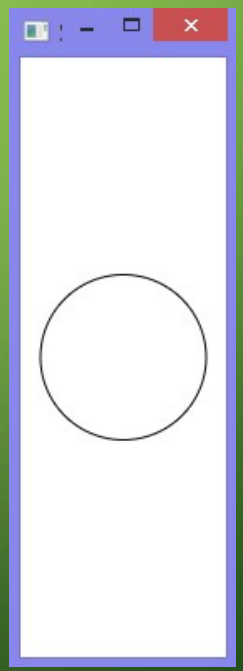
- If the value in the source object changes, the target property is also changed automatically
- The target object is simply called a binding object or a binding property

Resizing the window in the previous example would cover the shape:



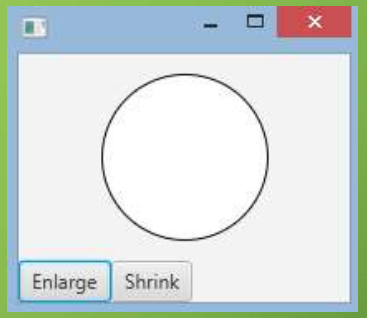
CIRCLE BOUND TO CENTER

```
public class ShowCircleCentered extends Application {
    @Override
    public void start(Stage primaryStage) {
        Pane pane = new Pane();
        Circle circle = new Circle();
        circle.centerXProperty().bind(pane.widthProperty().divide(2));
        circle.centerYProperty().bind(pane.heightProperty().divide(2));
        circle.setRadius(50);
        circle.setStroke(Color.BLACK);
        circle.setFill(Color.WHITE);
        pane.getChildren().add(circle);
        Scene scene = new Scene(pane, 200, 200);
        primaryStage.setTitle("ShowCircleCentered");
        primaryStage.setScene(scene);
        primaryStage.show();
```



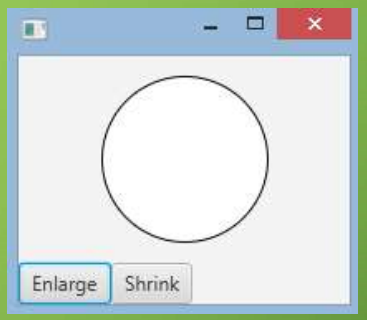
SHRINKING/ENLARGING CIRCLE

```
public class ControlCircle extends Application {
    private CirclePane circlePane = new CirclePane();
    @Override
    public void start(Stage primaryStage) {
        HBox hBox = new HBox();
        Button btEnlarge = new Button("Enlarge");
        Button btShrink = new Button("Shrink");
        hBox.getChildren().add(btEnlarge);
        hBox.getChildren().add(btShrink);
        btEnlarge.setOnAction(new EnlargeHandler());
        btShrink.setOnAction(new ShrinkHandler());
        BorderPane borderPane = new BorderPane();
        borderPane.setCenter(circlePane);
        borderPane.setBottom(hBox);
        BorderPane.setAlignment(hBox, Pos.CENTER);
        Scene scene = new Scene(borderPane, 200, 150);
        primaryStage.setScene(scene); primaryStage.show();
    } …
```



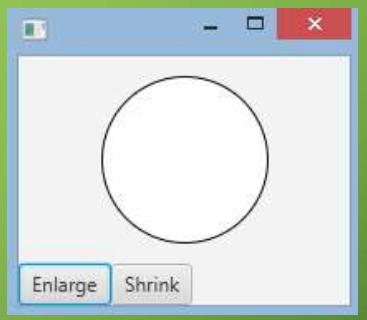
SHRINKING/ENLARGING CIRCLE

```
    // Inner Class
    class EnlargeHandler implements EventHandler<ActionEvent> {
        @Override
        public void handle(ActionEvent e) {
            circlePane.enlarge();
        }
    }

    class ShrinkHandler implements EventHandler<ActionEvent> {
        @Override
        public void handle(ActionEvent e) {
            circlePane.shrink();
        }
    }
}
```



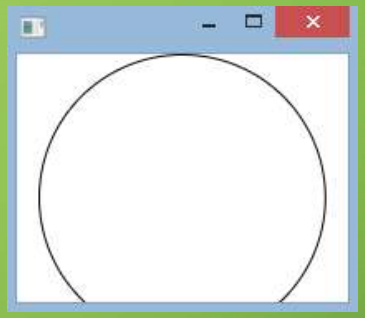
SHRINKING/ENLARGING CIRCLE

```
class CirclePane extends StackPane {
    private Circle circle = new Circle(50);
    public CirclePane() {
        getChildren().add(circle);
        circle.setStroke(Color.BLACK);
        circle.setFill(Color.WHITE);
    }
    public void enlarge() {
        circle.setRadius(circle.getRadius() * 1.2);
    }
    public void shrink() {
        circle.setRadius(circle.getRadius() * .8);
    }
}
```



USING THE MOUSE

```
public class ControlCircleWithMouse extends Application {
  private CirclePane circlePane = new CirclePane();
  @Override
  public void start(Stage primaryStage) {
    HBox hBox = new HBox();
    hBox.setSpacing(10);
    hBox.setAlignment(Pos.CENTER);
    circlePane.setOnMouseClicked(e -> {
      if (e.getButton() == MouseButton.PRIMARY) {
        circlePane.enlarge();
      }
      else if (e.getButton() == MouseButton.SECONDARY) {
        circlePane.shrink();
      }
    });
    …
```

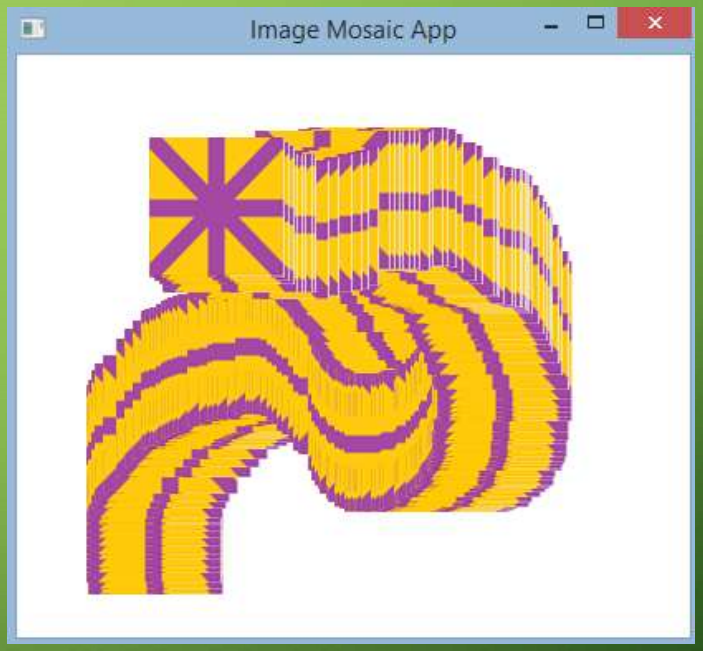


CANVAS

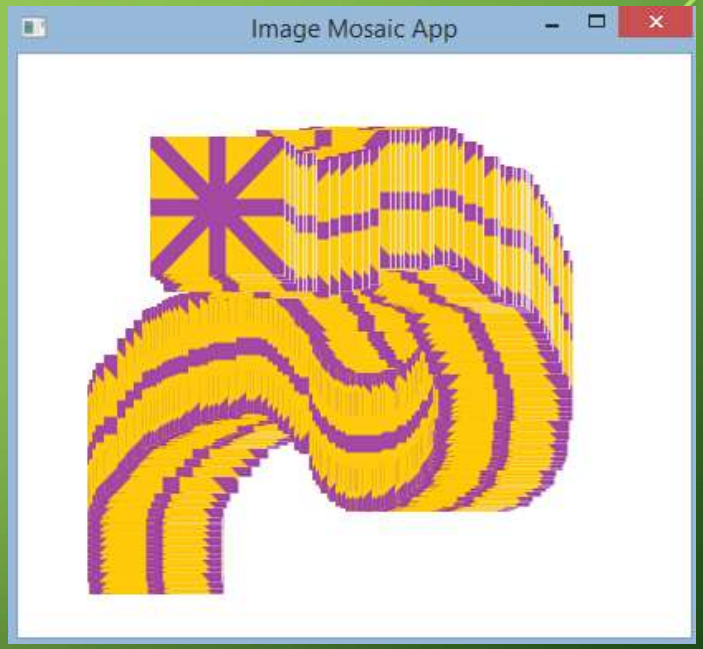
Rendering Alternative

Surface we can render to

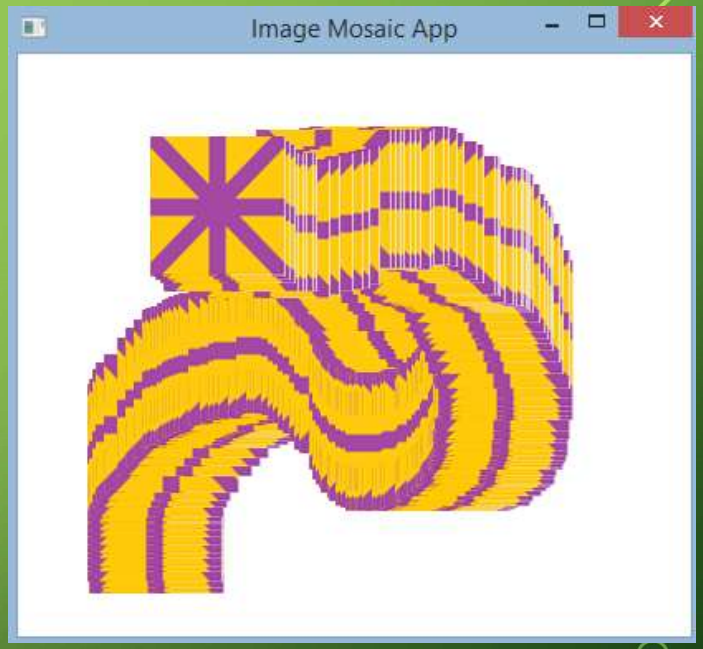
Mirrors HTML5 Canvas usage



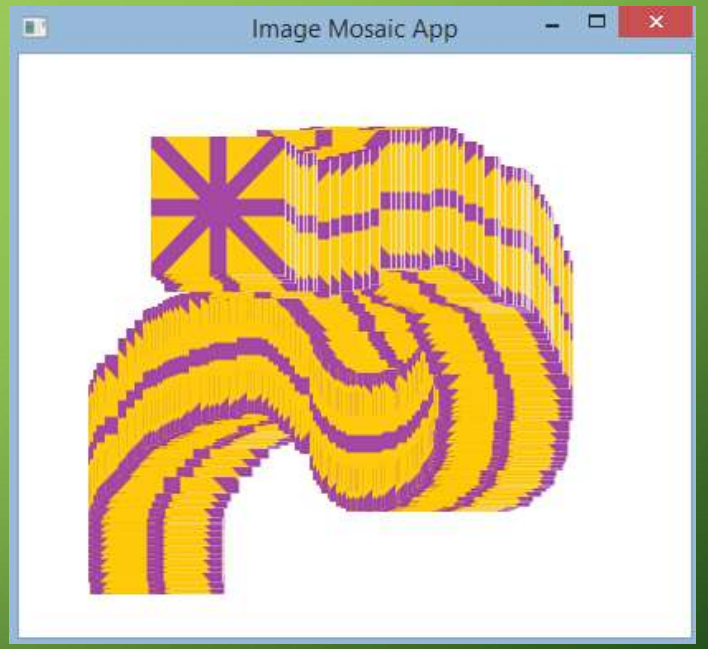
```
public class ImageMosaicApp extends Application {
    Stage primaryStage;
    Scene scene;
    Canvas canvas;
    GraphicsContext gc;
    Image logo1Image, logo2Image;
    ArrayList<Point2D> logo1Locations, logo2Locations;

    public void start(Stage initPrimaryStage) {
        primaryStage = initPrimaryStage;
        initStage();
        initData();
        initGUI();
        initHandlers();
    }
```



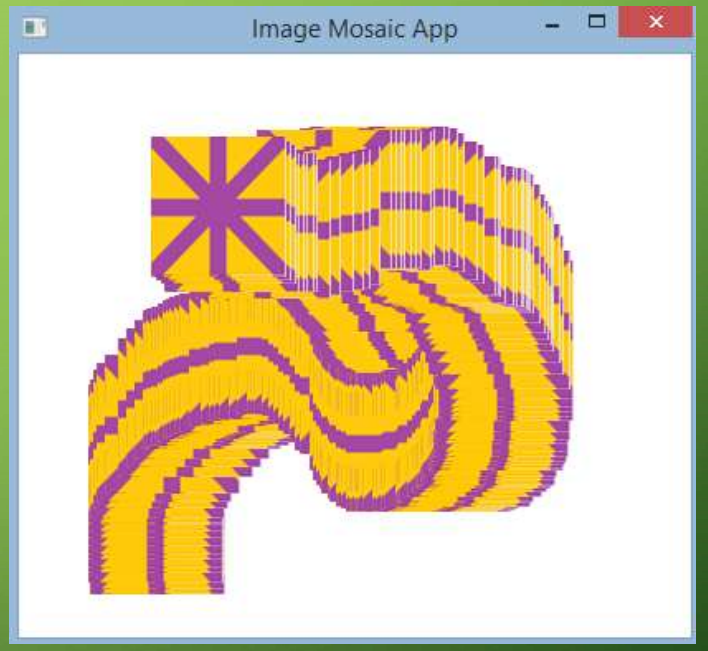
```
public void initStage() {
        primaryStage.setTitle("Image Mosaic App");
        Screen screen = Screen.getPrimary();
        Rectangle2D bounds = screen.getVisualBounds();
        primaryStage.setX(bounds.getMinX());
        primaryStage.setY(bounds.getMinY());
        primaryStage.setWidth(bounds.getWidth());
        primaryStage.setHeight(bounds.getHeight());
    }
```



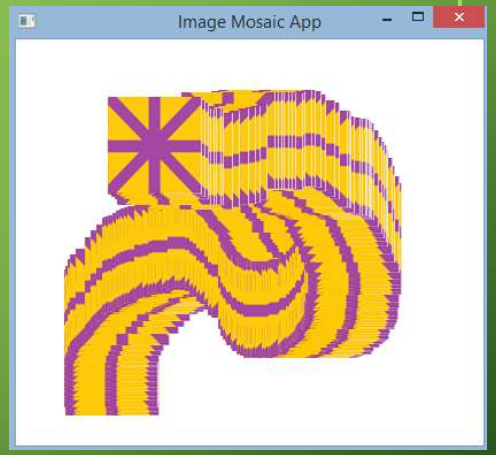
```
public void initData() {
    logo1Locations = new ArrayList();
    logo2Locations = new ArrayList();
    logo1Image = new Image("Logo1.png");
    logo2Image = new Image("Logo2.png");
}
```



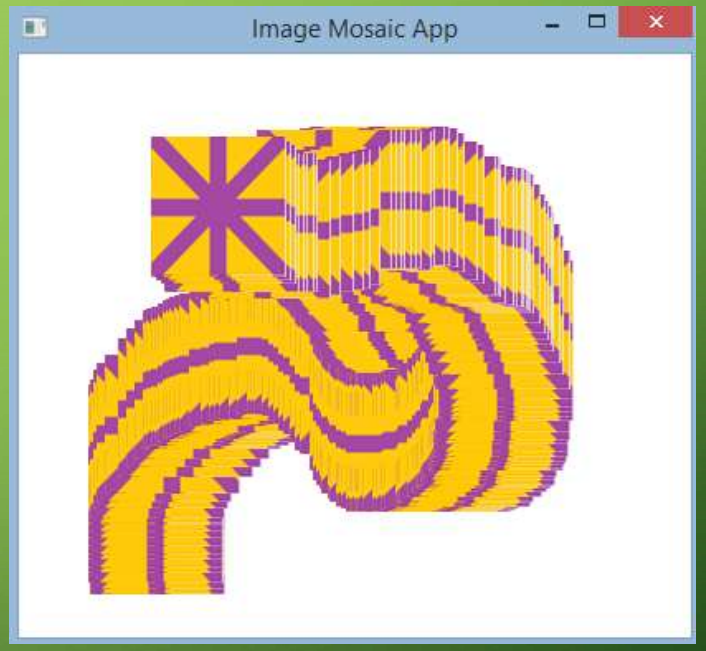
```
public void initGUI() {
    canvas = new Canvas();
    gc = canvas.getGraphicsContext2D();
    Group root = new Group();
    root.getChildren().add(canvas);
    scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.show();
    canvas.setWidth(scene.getWidth());
    canvas.setHeight(scene.getHeight());
}
```



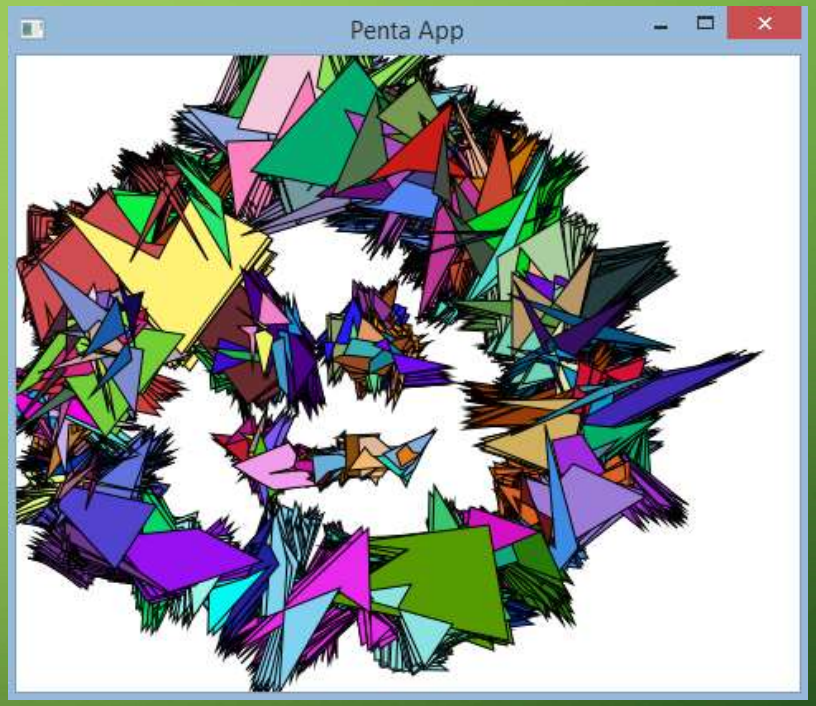
```
public void initHandlers() {
    canvas.setOnMouseClicked(mouseEvent -> {
        Point2D point = new Point2D(mouseEvent.getX(), mouseEvent.getY());
        if (!logo1Locations.contains(point))
            logo1Locations.add(point);
        draw();
    });
    canvas.setOnMouseDragged(mouseEvent -> {
       Point2D point = new Point2D(mouseEvent.getX(), mouseEvent.getY());
        if (!logo2Locations.contains(point))
            logo2Locations.add(point);
        draw();
    });
}
```



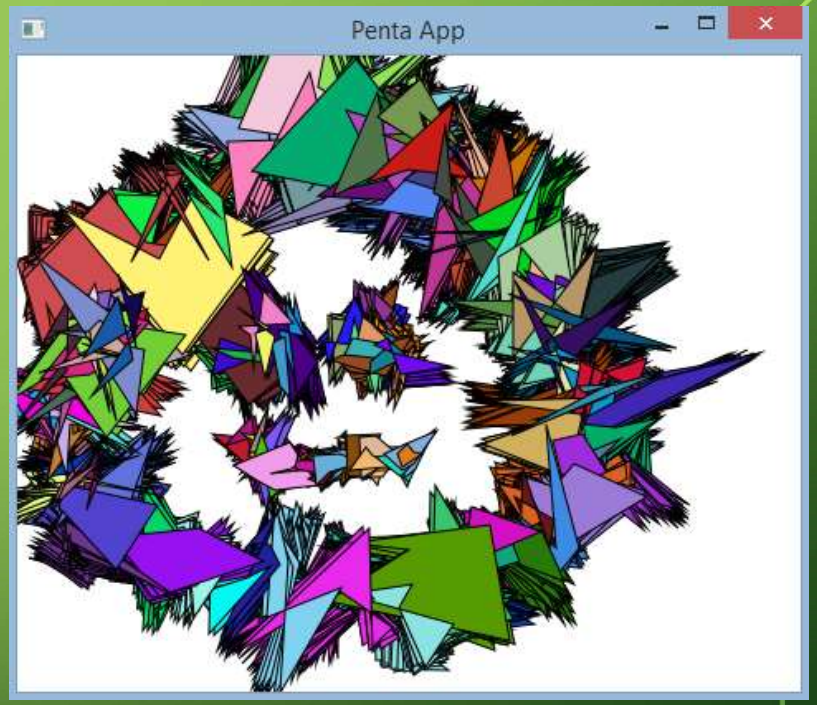
```
public void draw() {
        Iterator<Point2D> it = logo1Locations.iterator();
        while (it.hasNext()) {
            Point2D p = it.next();
            gc.drawImage(logo1Image, p.getX(), p.getY());
        }
        it = logo2Locations.iterator();
        while (it.hasNext()) {
            Point2D p = it.next();
            gc.drawImage(logo2Image, p.getX(), p.getY());
        }
    }
    public static void main(String[] args) {
        launch();
    }
}
```



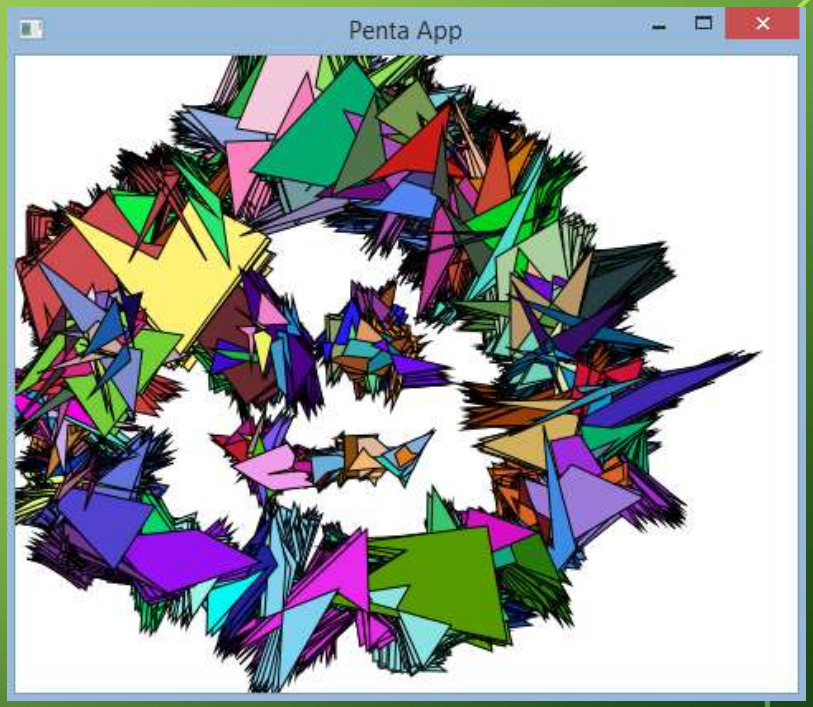
```
public class PentaApp extends Application {
    private Stage primaryStage;
    private Scene scene;
    private Canvas canvas;
    private GraphicsContext gc;
    private ArrayList<double[]> xPoints;
    private ArrayList<double[]> yPoints;
    private ArrayList<Color> colors;

    @Override
    public void start(Stage initPrimaryStage) {
        primaryStage = initPrimaryStage;
        initStage();
        initData();
        initGUI();
        initHandlers();
    }
```



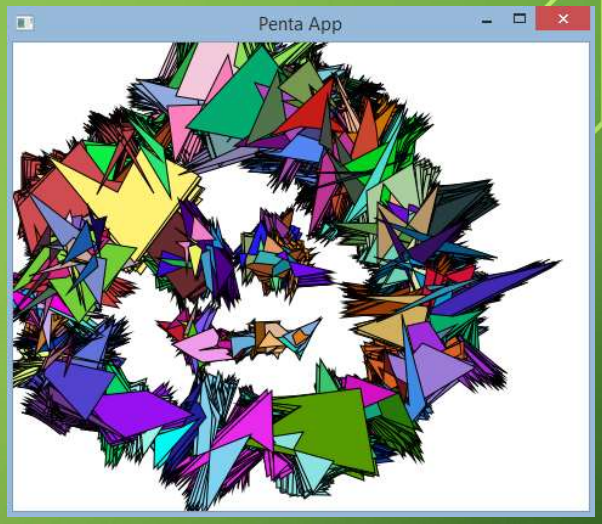
```
public void initStage() {
    primaryStage.setTitle("Penta App");
    Screen screen = Screen.getPrimary();
    Rectangle2D bounds = screen.getVisualBounds();
    primaryStage.setX(bounds.getMinX());
    primaryStage.setY(bounds.getMinY());
    primaryStage.setWidth(bounds.getWidth());
    primaryStage.setHeight(bounds.getHeight());
}

public void initData() {
    xPoints = new ArrayList();
    yPoints = new ArrayList();
    colors = new ArrayList();
}
```



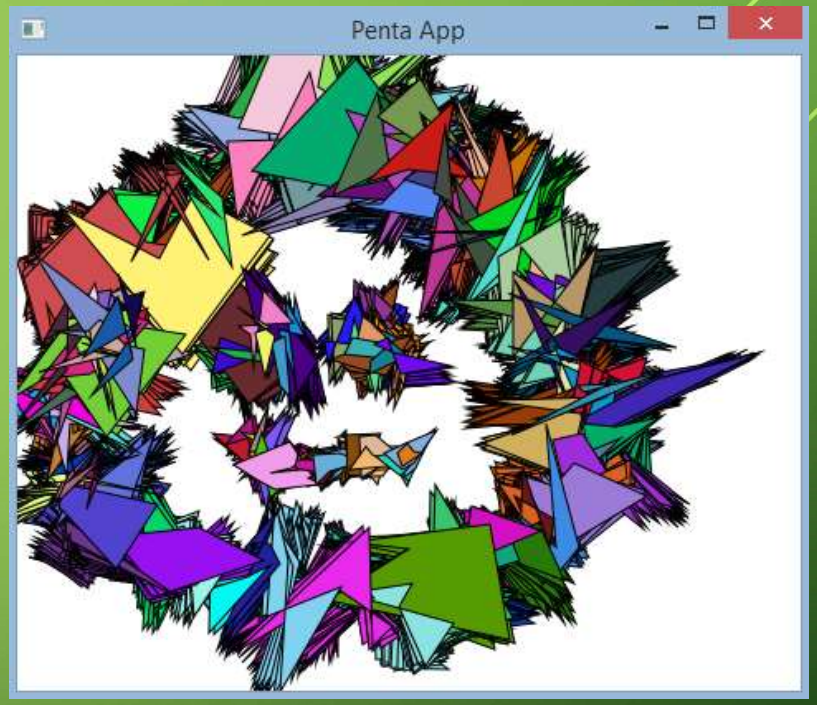
```
public void initGUI() {
    canvas = new Canvas();
    gc = canvas.getGraphicsContext2D();
    Group root = new Group();
    root.getChildren().add(canvas);
    scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.show();
    canvas.setWidth(scene.getWidth());
    canvas.setHeight(scene.getHeight());
}
```



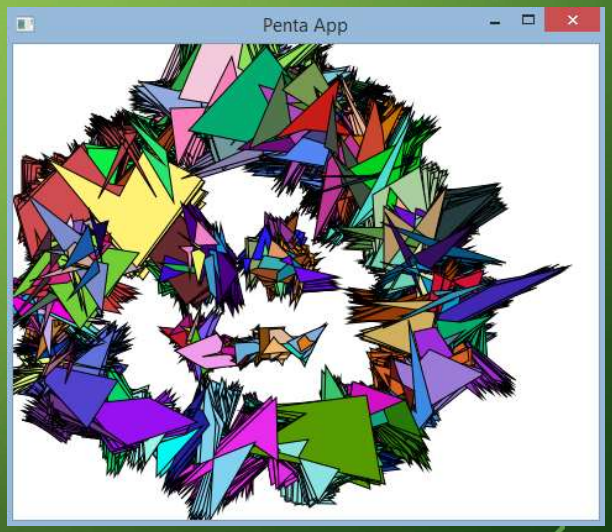
```
public void initHandlers() {
    canvas.setOnMouseClicked(mouseEvent -> {
        if (mouseEvent.getClickCount() == 2) {
            xPoints.clear();
            yPoints.clear();
            colors.clear();
            gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
        }
    });
```



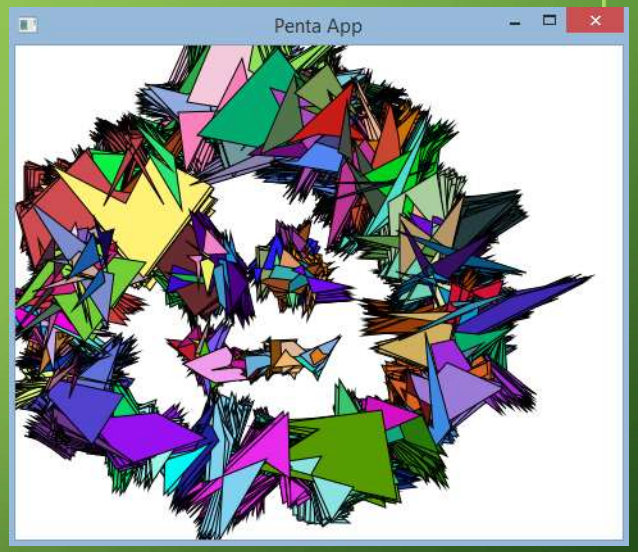
```
canvas.setOnMouseDragged(mouseEvent -> {
        double x = mouseEvent.getX();
        double y = mouseEvent.getY();
        double[] xs = new double[5];
        double[] ys = new double[5];

        // CENTER
        xs[0] = x;
        ys[0] = y - (int)(Math.random() * 20) - 1;

        // TOP-RIGHT POINT
        xs[1] = x + (int)(Math.random() * 15) + 1;
        ys[1] = y - (int)(Math.random() * 10) - 1;
```



```
// BOTTOM-RIGHT POINT
xs[2] = x + (int)(Math.random() * 10) + 1;
ys[2] = y + (int)(Math.random() * 15) + 1;

// BOTTOM-LEFT POINT
xs[3] = x - (int)(Math.random() * 10) - 1;
ys[3] = y + (int)(Math.random() * 15) + 1;

// TOP-LEFT POINT
xs[4] = x - (int)(Math.random() * 15) - 1;
ys[4] = y - (int)(Math.random() * 10) - 1;
```



```
...
            xPoints.add(xs);
            yPoints.add(ys);
            int r = (int)(Math.random() * 256);
            int g = (int)(Math.random() * 256);
            int b = (int)(Math.random() * 256);
            colors.add(Color.rgb(r,g,b));
            PentaApp.this.draw();
        });
    }
```



```
public void draw() {
        for (int i = 0; i < xPoints.size(); i++) {
            double[] xVertices = xPoints.get(i);
            double[] yVertices = yPoints.get(i);
            for (int j = 0; j < 5; j++) {
                xVertices[j] += (int)(Math.random()*9) - 4;
                yVertices[j] += (int)(Math.random()*9) - 4;
            }
            Color color = colors.get(i);
            gc.setFill(color);
            gc.fillPolygon(xVertices, yVertices, 5);
            gc.setStroke(Color.BLACK);
            gc.strokePolygon(xVertices, yVertices, 5);
        }
    }
```