Week 9 Lab: JavaFX Graphics

In this lab, we will use JavaFX to build a program that displays shapes in a window. JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.

1 Objectives

1. Draw shapes using JavaFX.

2 Team Programming

Switch roles this Week. Make a note of each person's name, role, and email.

As a reminder the roles are:

Spokesperson: reads the questions aloud, makes sure everyone contributes appropriately.

Quality Control: records all answers & questions, and provides team reflection to team & instructor.

Process Analyst: considers how the team could work and learn more effectively.

If your team has four members, add a Facilitator:

Facilitator: keeps track of time, talks to the instructor and other teams and compiles and runs programs.

You must swap roles every Week. By the end of the course, you must have performed each role at least three times.

Note: **Everyone Codes!** You can divide up the work, but each member of the team must code some part of the overall lab solution. Remember, coding is a social activity — feel free to ask questions and help each other. Make sure that all questions get answered and everyone is able to contribute ideas toward the completion of the assignment. Sharing of ideas and code is encouraged among teammates within the lab. Pull the best code ideas together for your final submission.

Keep track of how much time it takes to complete each problem and log the times in your lab portfolio.

3 Background

A standard template for a JavaFX application follows in Listing 1. Use this as a template to get started on your program.

Drawing Geometric Shapes

JavaFX provides several methods for drawing 2-dimensional geometric shapes. The following code snippets provide examples of how to draw different shapes.

`Line`

`Line` represents a line segment in (x, y) coordinate space. See Listing 2.

`Circle`

`Circle` creates a new circle with the specified radius and center location measured in pixels. See Listing 3.

Listing 1: Week09Lab.java

```java
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class Week09Lab extends Application {
   /**
    * The start method. Required by Application
    * @param args
    */
   public void start(Stage stage) {
      Group root = new Group( );
      Scene scene = new Scene( root, 400, 300 );

      // Your code goes here.

      stage.setTitle("My JavaFX Application");
      stage.setScene(scene);
      stage.show();
   }
}
```

Listing 2: Line Example

```java
// import the Line class at the top of your source file.
import javafx.scene.shape.Line;

// Your start method should be enclosed in a class.
public void start(Stage stage) {
   // A group contains objects that are rendered in order.
   Group root = new Group( );
   Scene scene = new Scene( root, 400, 300 );

   // Here we create the Line and add it to the Group to be rendered later.
   // Arguments to the constructor are doubles: x1, y1, x2, y2
   Line line = new Line( 0, 0, 400, 300 );
   line.setStroke(Color.rgb(127, 244, 16));
   line.setStrokeWidth(10);
   root.getChildren( ).add( line );

   stage.setTitle("Shape Test");
   stage.setScene(scene);
   stage.show();
}
```

Listing 3: Circle Example

```java
// import the Line class at the top of your source file.
import javafx.scene.shape.Circle;

 // Here we create the Circle and add it to the Group to be rendered later.
 // Arguments to the constructor are doubles: x, y, radius
 // This code should be encapsulated within the start( ) method.
 Circle circle = new Circle( 200, 150, 125 );
 circle.setFill(Color.rgb(32, 244, 64));
 circle.setStroke(Color.rgb(127, 244, 16));
 circle.setStrokeWidth(10);
 root.getChildren( ).add( circle );
```

`Rectangle`

The `Rectangle` class defines a rectangle with the specified size and location. By default the rectangle has sharp corners. See Listing 4.

Listing 4: Rectangle Example

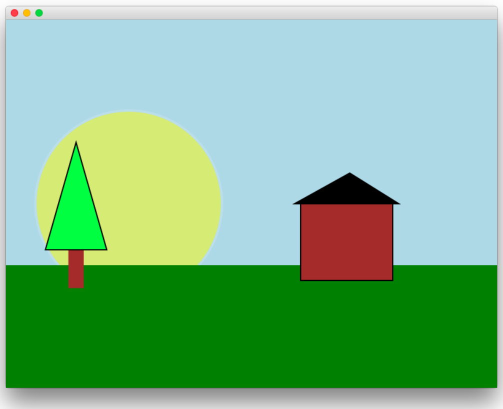
```java
// Here we create the Rectangle and add it to the Group to be rendered later.
 // Arguments to the constructor are doubles: x, y, width, height
 // This code should be encapsulated in the start( ) method.
 Rectangle rect = new Rectangle( 50, 50, 300, 200 );
 rect.setFill(Color.rgb(232, 44, 64));
 rect.setStroke(Color.rgb(0, 0, 0));
 rect.setStrokeWidth(10);
 root.getChildren( ).add( rect );
```

4 Problems

Problem 1: Render an original scene.

Use the JavaFX application template provided above to construct your program in a class named **Week09LabProblem1**. Open a window and draw an original scene using basic shapes.

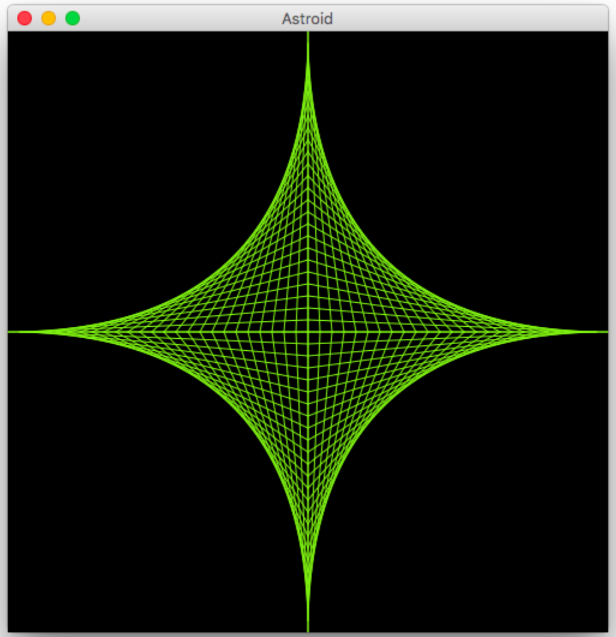
The image below is an example of a program that draws a scene.



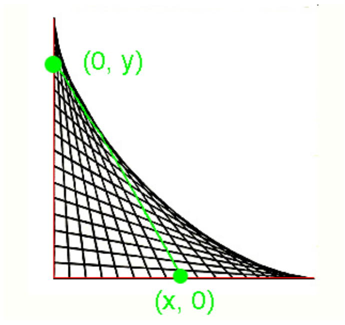
Problem 2: Render an astroid with lines.

Using JavaFX draw an astroid using straight lines. An astroid is a particular mathematical curve: a hypocycloid with four cusps. Its modern name comes from the Greek word for "star".

Develop a class named **Week09LabProblem2** that uses JavaFX to draw a 4-cusped astroid using lines as in the image below.

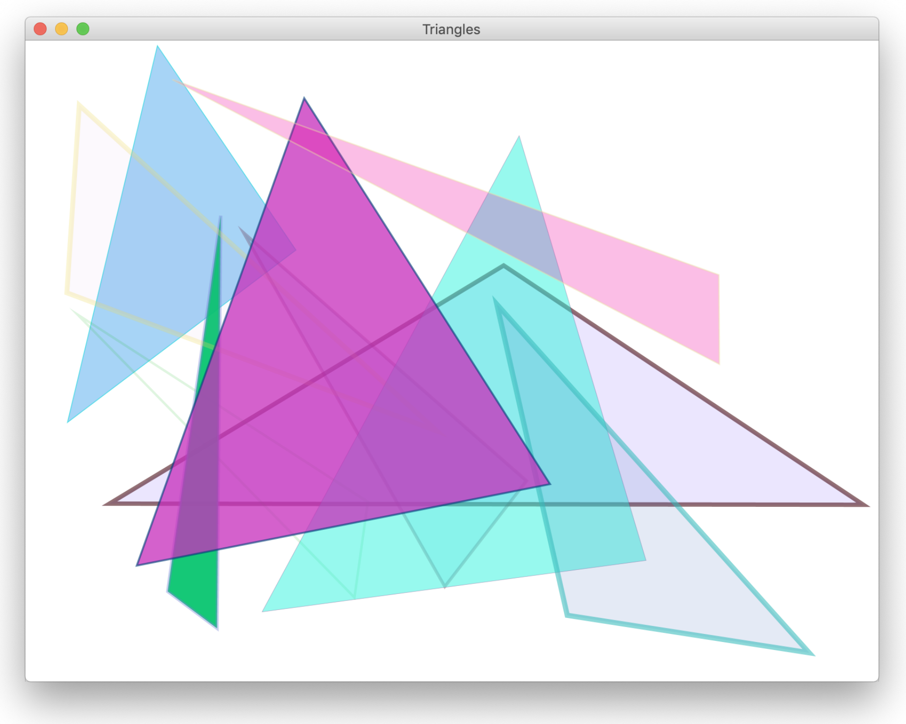


The astroid can be formed as the envelope produced when a line segment is moved with each end on one of a pair of perpendicular axes. It is the curve enveloped by a ladder sliding against a wall with the top corner moving along a vertical track.



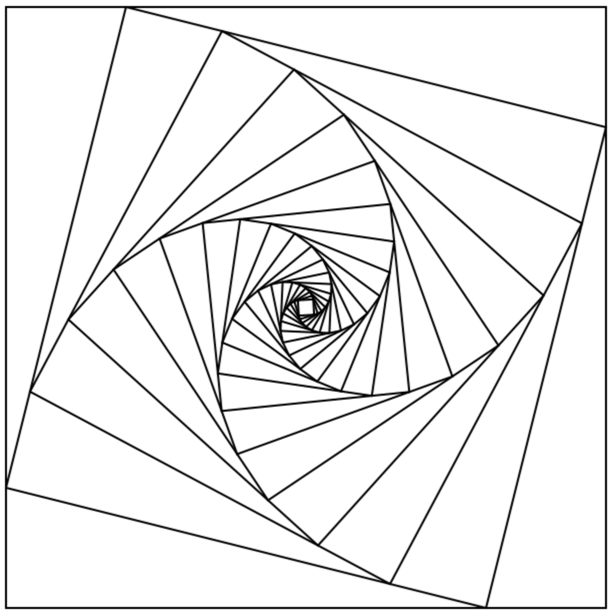
Problem 3: Render Random Triangles

Develop a class named **Week09LabProblem3**. Draw 25 triangles at random locations on the stage with random size, colors, line thickness, fill, and opacity.



Problem 4: Nested Squares

Develop a class named **Week09LabProblem4**. Draw a set of nested squares using iteration. Each square slightly rotated from the previous square.



5 Reflection

1. What challenges did you encounter while writing the lab programs?
2. How did you go about deciding which shapes to use for drawing the freeform scene?
3. What mathematical considerations did you have to make while working on the lab?
4. How can understanding JavaFX and graphics programming be beneficial in other areas of computer science?

6 Completing the Lab

Before you leave the lab, show your Lab Instructor your work and ask for it to be graded and ask for feedback.

1. You will receive full credit if you complete all of the assigned tasks.
2. You will receive partial credit if you complete only some of the tasks.
3. You will receive no credit if you do not complete any of the tasks.

6.1 Submitting your Work

Each student must submit work for each problem, whether it is complete or not. This provides a record of your effort in the lab. If grade questions arise in the future, we will rely on this record to show your work.

1. Make sure you have saved everything for each problem in the lab an combined them into a single zip file called *Week09Lab.zip*.
2. Upload the `.zip` file to Canvas.
   a. Click the **Submit Assignment** button at the top of the lab.
   b. Add your files for uploading.
   c. Enter any comments about the assignment. Then click on the ***Submit Assignment*** button.
3. Make sure that all lab partners have a copy of your `.zip` file.