



Java Foundations

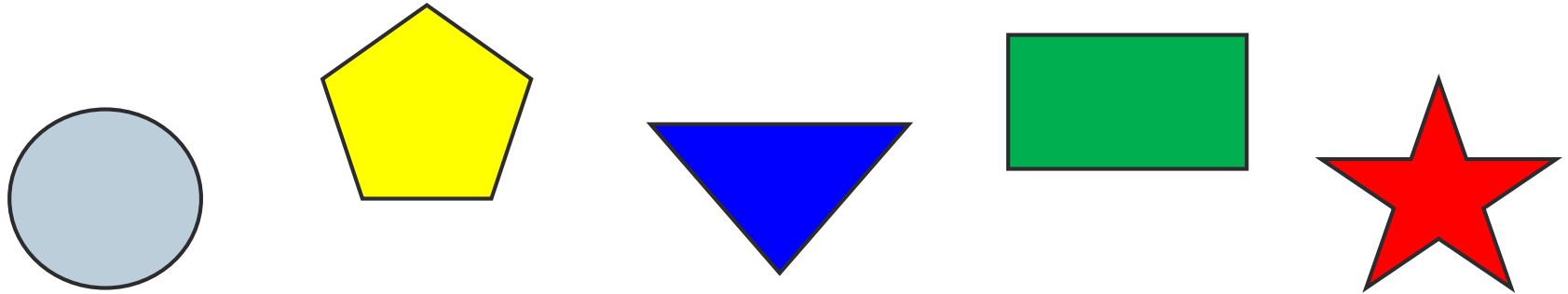
9-2

Colors and Shapes



Guess What, Boys and Girls?!

Today we're going to learn about colors and shapes!



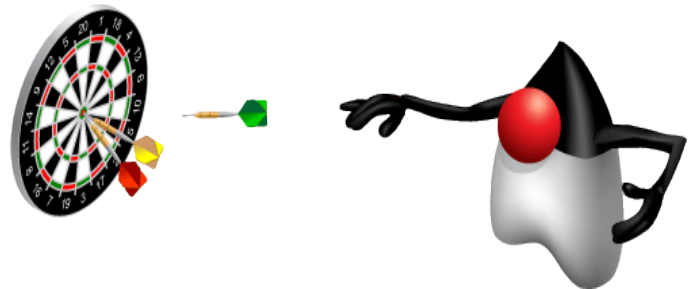
Yay!!!



Objectives

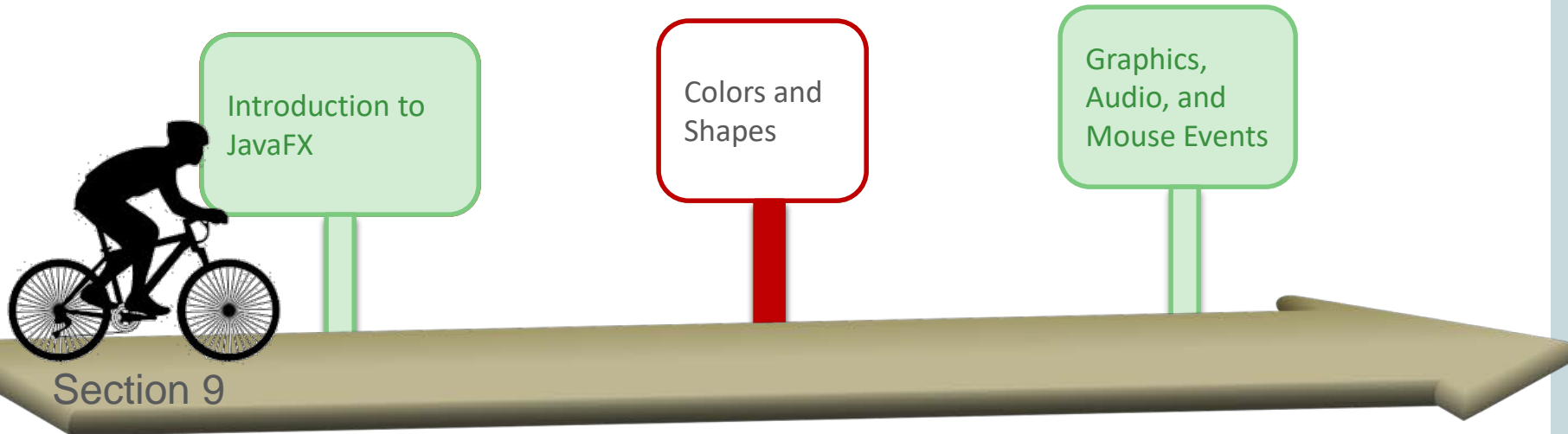
This lesson covers the following objectives:

- Create and use custom colors
- Create shapes and explain their properties and behaviors
- Reference the JavaFX Ensemble



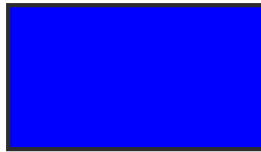
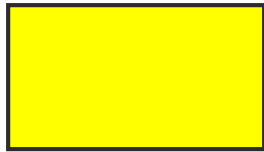
Topics

- Colors
- Shapes
- The JavaFX Ensemble



What Can I Do with Colors in JavaFX?

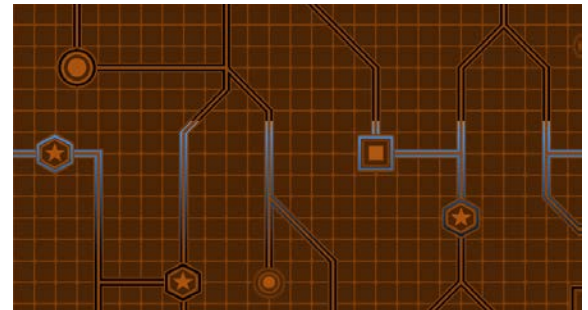
- Color shapes



- Create gradients



- Colorize images



JavaFX Contains a Color Class

- Colors can be stored as variables:

```
Color color = Color.BLUE;
```

- Colors can be passed in methods:

```
Scene scene = new Scene(root, 300, 250, Color.BLACK);
```

- This example makes the scene's background black.

- But before using any Color ...

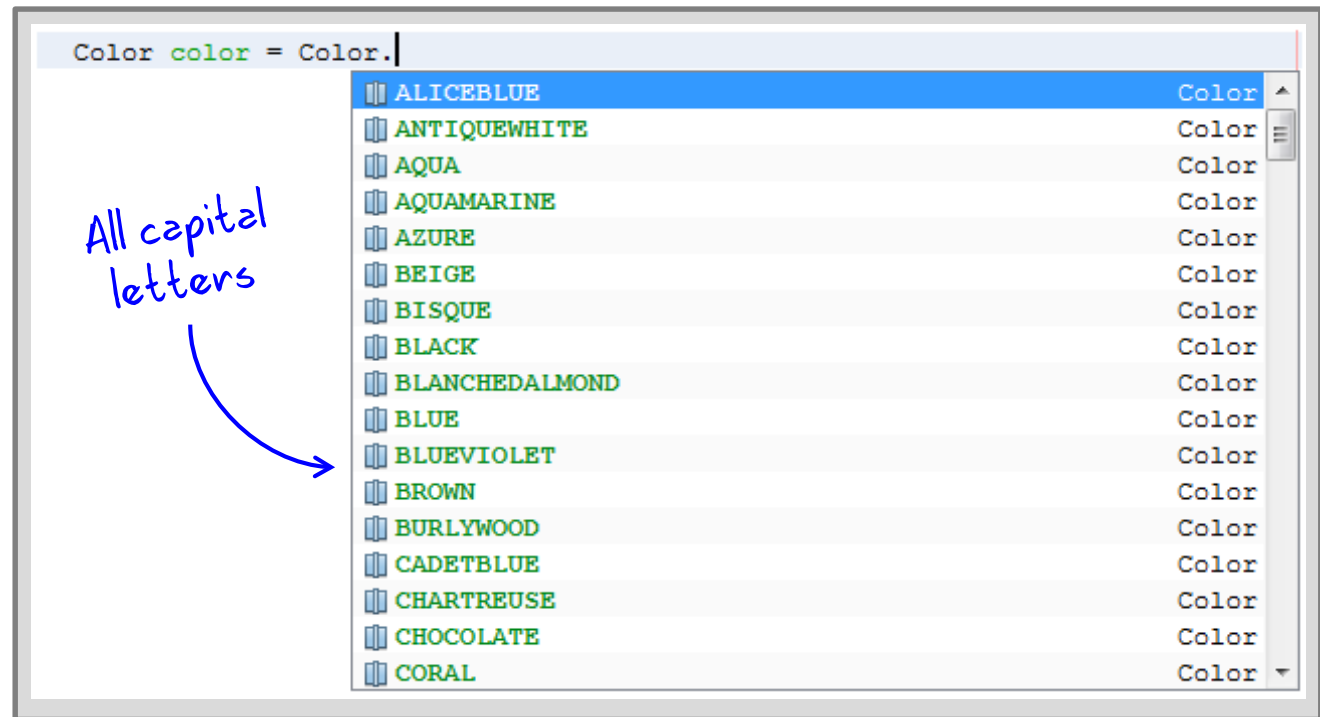
- You'll first need to make the following import:

```
import javafx.scene.paint.Color;
```

- Ignore NetBeans' other Color import suggestions.

Referencing a Color

- There are many colors in JavaFX.
- Typing `Color.` in NetBeans reveals the entire list of possible colors.



Customizing a Color

- If you're unhappy with the colors that JavaFX provides, there are ways to customize your own color.
- The Color class contains methods to do this:

```
Color cusotmColor = Color.rgb|
```

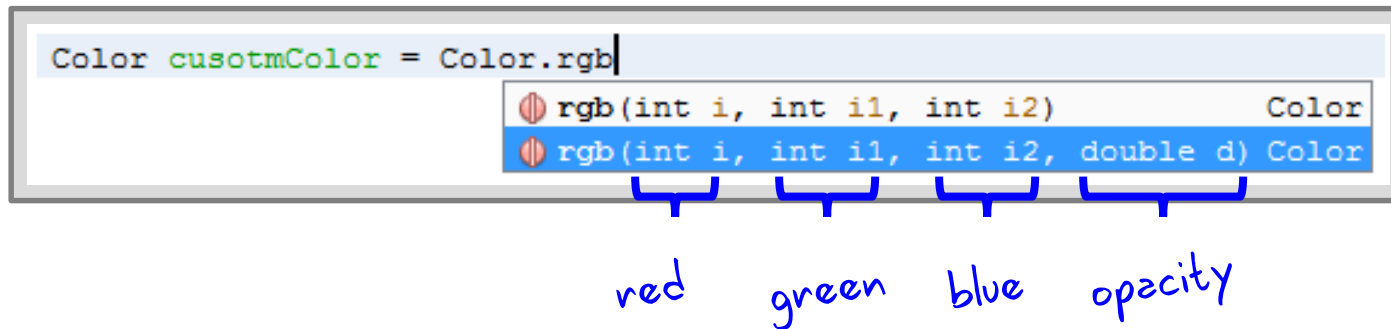
```
rgb(int i, int i1, int i2) Color
```

```
rgb(int i, int i1, int i2, double d) Color
```

red green blue opacity

- Customize a color by mixing red, green, and blue components.
- Opacity can also be controlled.

The Range of Color Components



Component	Range of values
Red	0–255
Green	0–255
Blue	0–255
Opacity	0.0–1.0

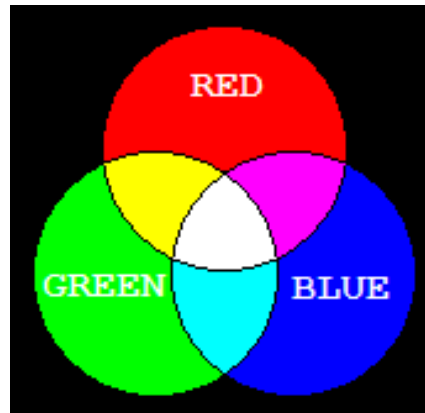
Color Example

- In this example, the resulting color contains ...

```
Color color = new Color.rgb(255, 255, 20);
```

- As much Red as possible
 - As much Green as possible
 - Only a little Blue
- The resulting color is very close to yellow.
 - But how do we know this?
 - For the most part, finding the perfect color is “guess and check,” but there are guiding principles.

Rules of Additive Color Mixing



Examples:

Code	Color
<code>Color.rgb(255, 0, 0);</code>	red
<code>Color.rgb(0, 255, 0);</code>	green
<code>Color.rgb(0, 0, 255);</code>	blue
<code>Color.rgb(255, 255, 0);</code>	yellow
<code>Color.rgb(0, 0, 0);</code>	black
<code>Color.rgb(255, 255, 255);</code>	white

Pure red
Pure green
Pure blue
No blue
No color
All color



Exercise 1

- Create a new JavaFX project.
 - Change the Root Node to a `Group` type.
 - Remove the button and any other unnecessary code relating to the button.
- Experiment with customizing colors.
 - Create a few custom colors.
 - Admire your custom colors through the scene's background by providing a `Color` argument when the `Scene` is instantiated.

Topics

- Colors
- Shapes
- The JavaFX Ensemble

Introduction to
JavaFX

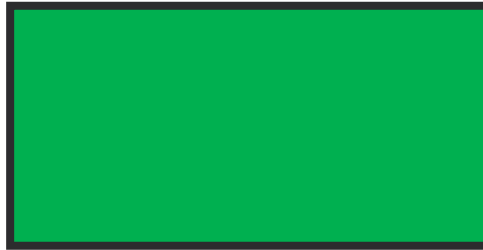
Colors and
Shapes

Graphics,
Audio, and
Mouse Events

Section 9

This Is a Rectangle

- This is how to instantiate a JavaFX Rectangle:



```
Rectangle rect = new Rectangle(20, 20, 100, 200);
```

x-position
y-position
width
height

- You'll first need to make the following import:

```
import javafx.scene.shape.Rectangle;
```

– Ignore NetBeans' other Rectangle import suggestions.

Important Methods for Rectangles

- We can get a sense of a Rectangle's properties from the constructor and the following methods:
 - **setX**(double d)
 - **setY**(double d)
 - **setWidth**(double d)
 - **setHeight**(double d)
 - **setFill**(Paint paint)
 - **setStroke**(Paint paint)
 - **setStrokeWidth**(double d)

(There are many more Rectangle methods besides these seven.)
- But what exactly will these methods do?

These can accept a color as an argument.

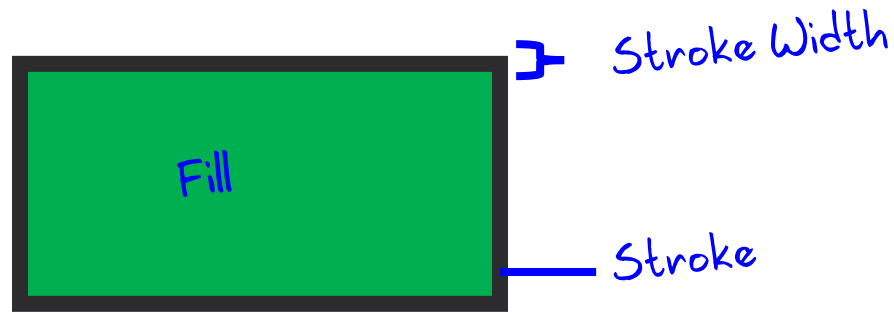


Exercise 2

- Continue editing the JavaFX project that you created in the previous exercise.
- Create a Rectangle and add it to the Root Node.
- Call each method outlined in the previous slide.
- Can you figure out what each method does?

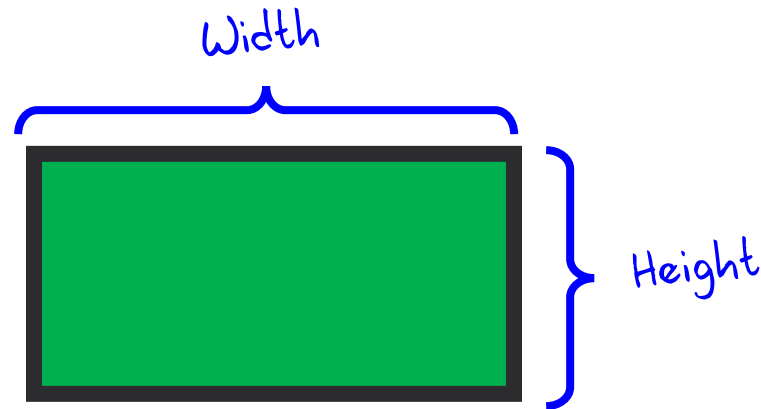
Method Descriptions, Part 1

- **setFill(Paint paint)**
 - Sets the color of the Rectangle
- **setStroke(Paint paint)**
 - Sets the color of the Rectangle's outline
- **setStrokeWidth(double d)**
 - Sets the width of the Rectangle's outline



Method Descriptions, Part 2

- **setX(double d)**
- **setY(double d)**
 - Sets the x or y position of the Rectangle
- **setWidth(double d)**
- **setHeight(double d)**
 - Sets the width or height of the Rectangle



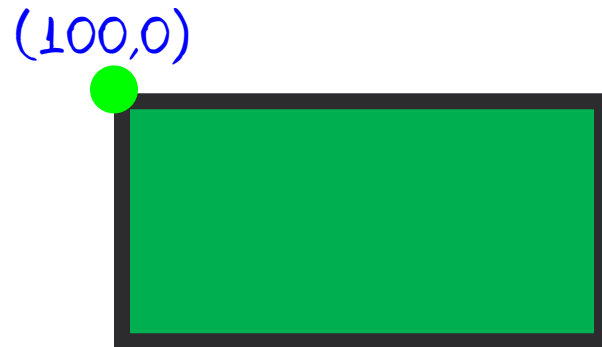
Changing a Node's Position

- We've seen a couple ways to change a node's position ... but which way is preferable?
- **setX(double d)**
- **setY(double d)**
 - These are preferable in most cases.
- **setLayoutX(double d)**
- **setLayoutY(double d)**
 - Use these if your Node is locked in a Layout pane, such as a FlowPane.
 - Or if `setX()` is unavailable, which is the case with UI elements, such as Buttons.

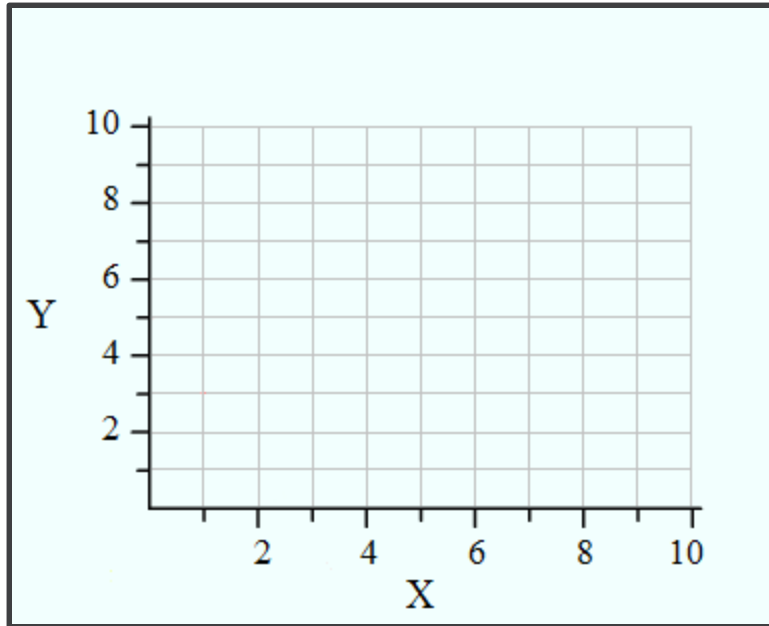
*setX() definitely won't
work in this case.*

Positioning a Node

- Most Nodes are positioned with respect to their top-left corner.
 - And not with respect to their geographic center.
- If you call `setX(100)` on a Node ...
 - The x-position of the Node's top-left corner is set to 100.

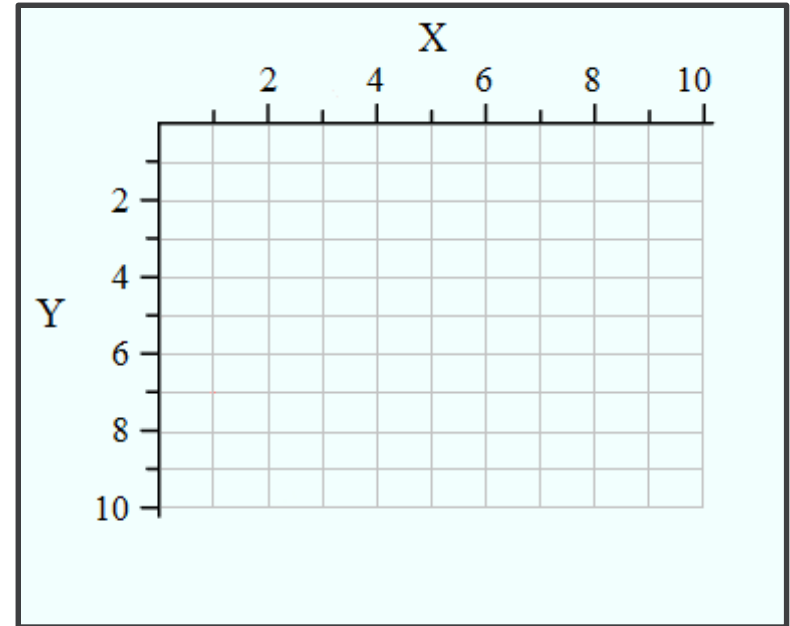


Coordinate Systems



Mathematical Coordinate System

- The origin is located at the bottom-left corner.

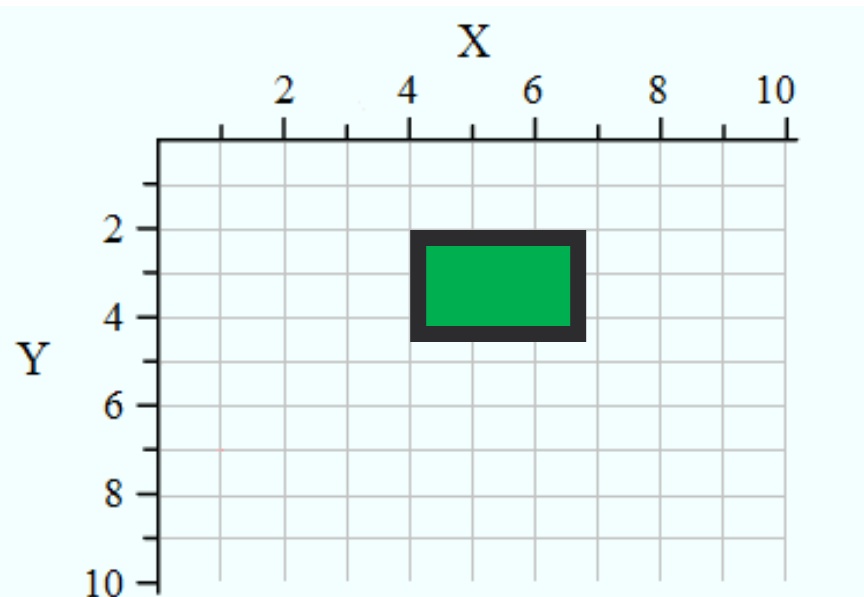


JavaFX Coordinate System

- The origin is located at the top-left corner.
- The y-axis is backward.

Positioning Example

- This Rectangle is positioned at (4,2) by calling:
 - `setX(4);`
 - `setY(2);`



Topics

- Colors
- Shapes
- The JavaFX Ensemble

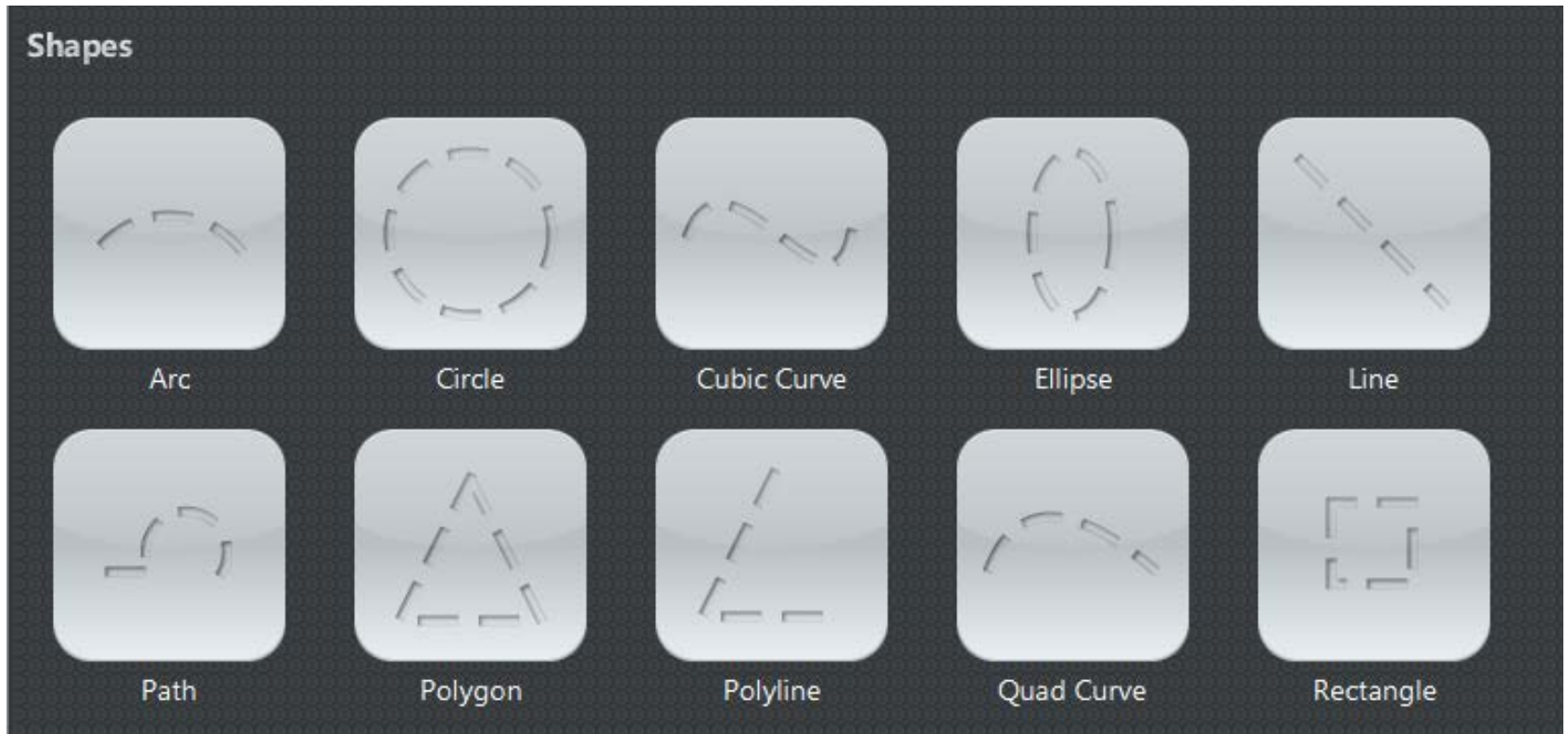
Introduction to
JavaFX

Colors and
Shapes

Graphics,
Audio, and
Mouse Events

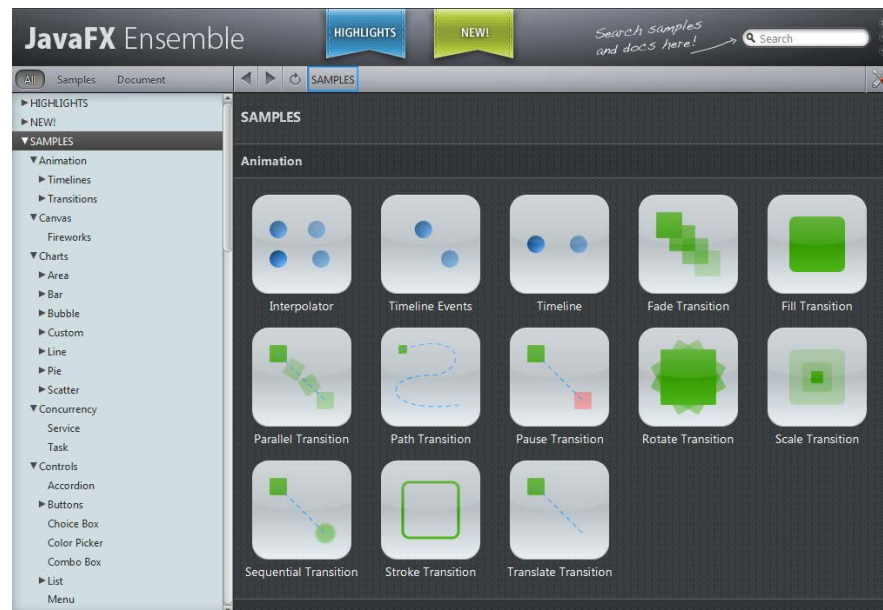
Section 9

Many Shapes Are Available in JavaFX



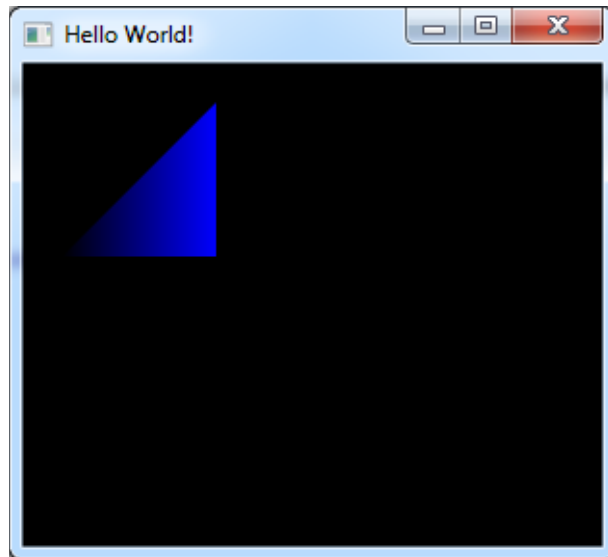
The JavaFX Ensemble

- This contains code examples of JavaFX features.
- We often consulted the Ensemble while developing Java Puzzle Ball.
- It's a helpful tool to explore and troubleshoot JavaFX.



Exercise 3

- Explore the JavaFX Ensemble.
- Can you figure out how to create a right triangle with a gradient coloring?



Exploring the Ensemble: Linear Gradient Example

- The Linear Gradient example shows us ...
 - How to create a gradient:



```
//create simple linear gradient
LinearGradient gradient1 = new LinearGradient(0, 0, 1, 0, true,
CycleMethod.NO_CYCLE, new Stop[] {
    new Stop(0, Color.DODGERBLUE),
    new Stop(1, Color.BLACK)
});
```

- How to color a shape with a gradient:

```
//First rectangle
Rectangle rect1 = new Rectangle(0,0,80,80);

//set rectangle fill
rect1.setFill(gradient1);
```

- Remember to make the proper imports.

Exploring the Ensemble: Polygon Example



- The Polygon example shows us ...
 - How to create a polygon from an array of points:

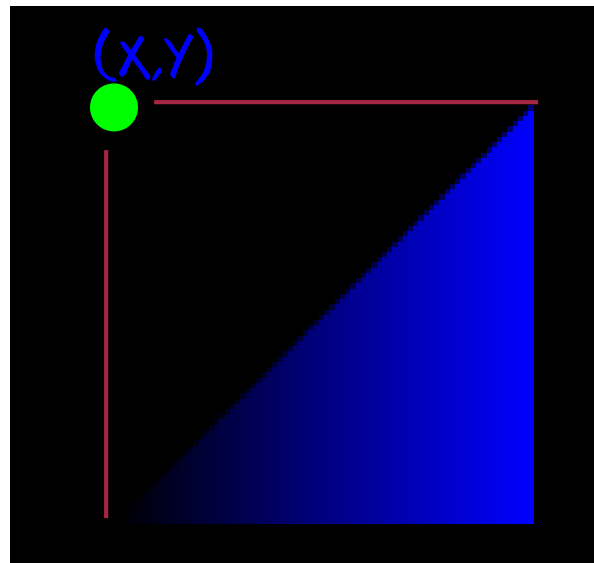
```
// Simple triangle
Polygon polygon1 = new Polygon(new double[] {
    45 , 10 ,
    10 , 80 ,
    80 , 80 ,
});
```

- Combine this with the gradient example, and you'll have your solution.
 - But even better, you'll understand how the Ensemble is a valuable resource.
 - This could prove very useful when you do the problem set.

The Polygon



- The Polygon has similar methods as a Rectangle.
 - Nodes share the same methods.
- If you experiment with `setLayoutX() ...`
 - You'll notice that the Polygon is positioned with respect to where its top-left corner would be.



Secrets about Java Puzzle Ball

- We drew lines and polygons for collision detection
 - But these lines are hidden in the latest version.



- We also drew two octagons around each bumper.
 - An inner octagon handles collision detection.
 - An outer octagon detects if the ball is far enough away for the bumper to rotate.
- We had to do extra work to position and rotate Nodes the way we wanted.

Summary

In this lesson, you should have learned how to:

- Create and use custom colors
- Create shapes and explain their properties and behaviors
- Reference the JavaFX Ensemble

