

# JAVAFX - GRAPHIC

# CONTENT

2

- Graphics
  - Canvas, GraphicsContext
- AnimationTimer
- Design
  - Shared Object, Drawing Part, Logic Part
- Handling User Input
  - Mouse, Keyboard
- Audio
- Export Jar
- Conclusion

# Graphics

# Drawing

4

- Where to draw ?
- How to draw ?

# Canvas

5

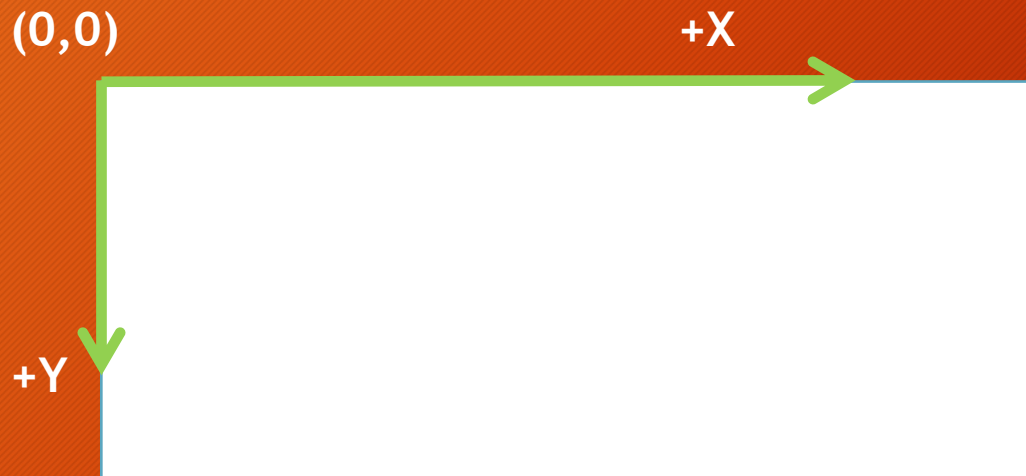
- Empty component
- See it as Paper (**Where to draw**)
- Can draw text, shapes, lines and images using a set of graphics commands provided by a GraphicsContext.
- Canvas has 2 constructors
  - `Canvas canvas = new Canvas();`
    - Create a Canvas of zero width and height
    - Can set width and height later
  - `Canvas canvas = new Canvas(double width, double height);`



# Canvas - Coordinate system

6

- Vertically flipped version of real world



# Example :

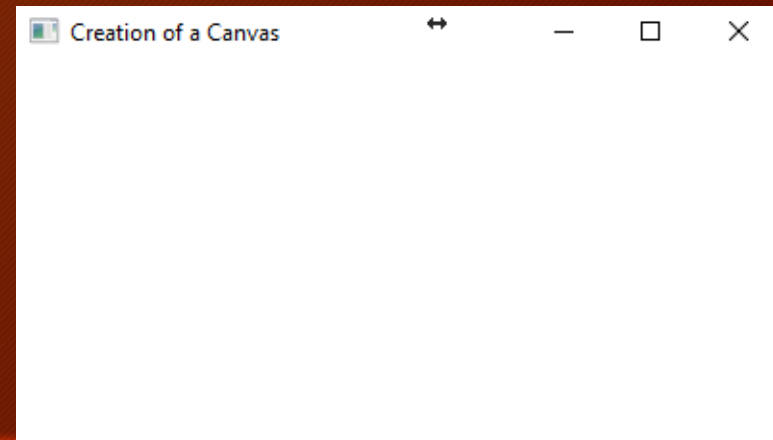
## *FxCanvasExample0.java*

7

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Creation of a Canvas");

    Canvas canvas = new Canvas(400, 200);
    root.getChildren().add(canvas);

    stage.show();
}
```



# GraphicsContext

8

- Drawing class
- See it as Pen, Pencil, Brush **(How to draw)**
- Contains a wealth of powerful customization abilities
- `GraphicsContext gc = canvas.getGraphicsContext2D()`
  - Get the graphics context of the canvas
- Drawings that fall outside the bounds of the Canvas are clipped



# GraphicsContext

9

- `setLineWidth(Double lw)`
- `setFill(Paint p)`
- `setStroke(Paint p)`
- `restore()`
  - Used to remove all properties from GraphicsContext



# Example : *FxCanvasExample1.java*

10

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Creation of a Canvas");

    Canvas canvas = new Canvas(400, 200);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);
    setBackground(gc);
    drawString(gc);

    stage.show();
}
```

```
public void setBackground(GraphicsContext gc) {
    gc.setFill(Color.BLACK);
    gc.fillRect(0, 0, gc.getCanvas().getWidth(), gc.getCanvas().getHeight());
}

public void drawString(GraphicsContext gc) {
    Font theFont = Font.font("Times New Roman", FontWeight.BOLD, 32);
    gc.setFont(theFont);

    gc.setFill(Color.RED);
    gc.fillText("Valor", 60, 50);

    gc.setFill(Color.BLUE);
    gc.fillText("Mystic", 60, 100);

    gc.setFill(Color.YELLOW);
    gc.fillText("Instinct", 60, 150);
}
```

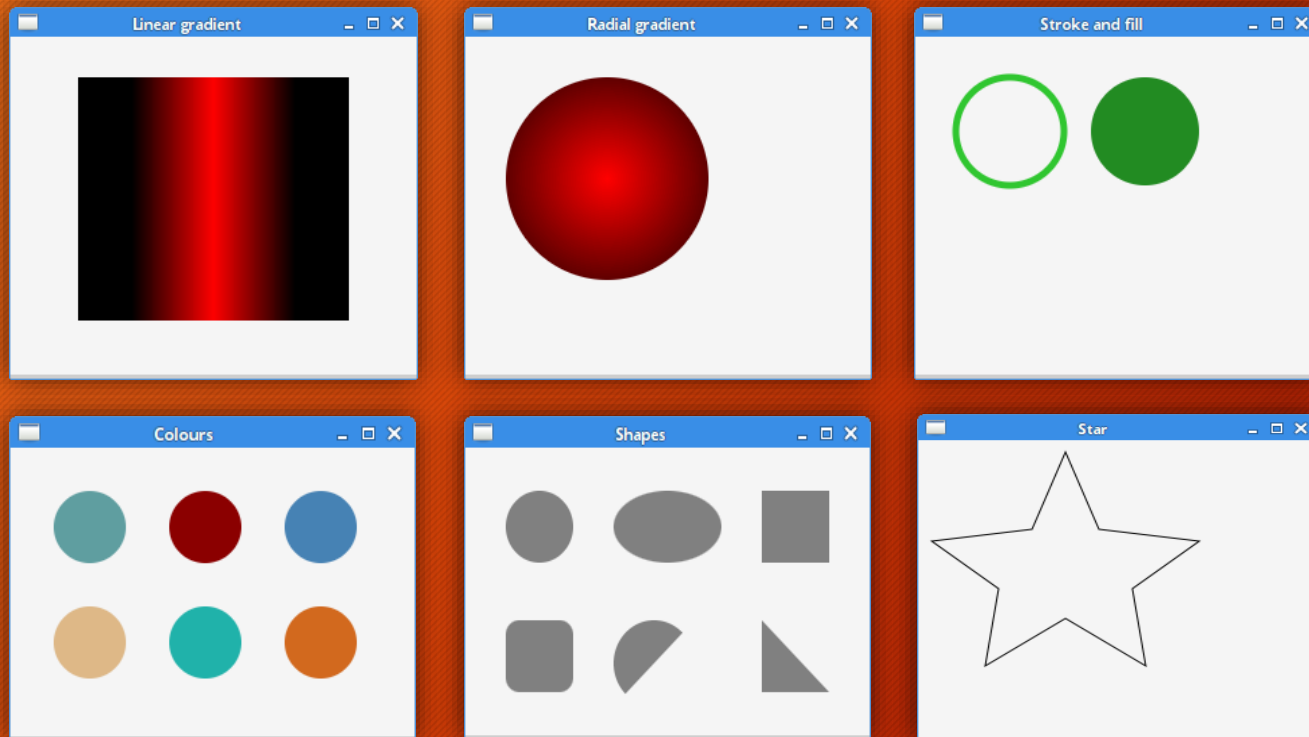
**Valor**

**Mystic**

**Instinct**

# Drawing Example

11



# Drawing

12

- Basic shapes
- Text
- Paths
- Images
- Rotate

# Drawing - Basic Shapes

13

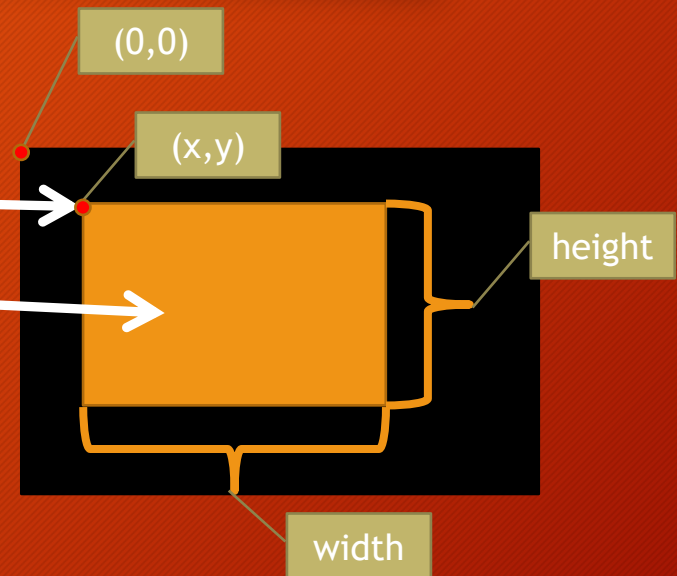
- Basic Shapes
  - fillRect(), fillRoundRect(), fillOval(), fillArc()
  - strokeLine(), strokeRect(), strokeRoundRect(), strokeOval(), strokeArc()
  - clearRect()
  - fillPolygon()
  - strokePolygon(), strokePolyline()



# Drawing 2D shapes

14

- Basic drawing methods
  - `setStroke` (the outline) // set property
  - `setFill` (the shape) // set property
  - `FillRect` // draw here
    - `gc.fillRect`(int x, int y, int width, int height)



# Example : *FxCanvasExample2.java*

15

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Basic Shapes");

    Canvas canvas = new Canvas(400, 200);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);
    drawRoundRect(gc);
    drawOval(gc);
    drawArc(gc);
    drawLine(gc);

    stage.show();
}
```

```
public void drawRoundRect(GraphicsContext gc) {
    gc.setLineWidth(2.0);
    gc.setFill(Color.RED);
    // Draw a rounded Rectangle
    // x,y,w,h,arcWidth,arcHeight
    gc.strokeRoundRect(10, 10, 50, 50, 10, 10);
    // Draw a filled rounded Rectangle
    gc.fillRoundRect(100, 10, 50, 50, 10, 10);
}
```

```
public void drawOval(GraphicsContext gc) {
    gc.setLineWidth(2.0);
    gc.setFill(Color.BLUE);
    // Draw an Oval
    // x,y,w,h
    gc.strokeOval(10, 70, 50, 30);
    // Draw a filled Oval
    gc.fillOval(100, 70, 50, 30);
}
```

```
public void drawLine(GraphicsContext gc) {
    gc.setLineWidth(2.0);
    gc.setFill(Color.BLACK);
    // Draw a Line
    // x1,y1,x2,y2
    gc.strokeLine(10, 190, 200, 190);
}
```

```
public void drawArc(GraphicsContext gc) {
    gc.setLineWidth(2.0);
    gc.setFill(Color.YELLOW);
    // Draw an Arc
    //x,y,w,h,startAngle,arcExtent,closure
    gc.strokeArc(10, 130, 50, 50, 40, 80, ArcType.ROUND);
    // Draw a filled Arc
    gc.fillArc(100, 130, 50, 50, 00, 120, ArcType.ROUND);
}
```

Basic Shapes



round  
closed  
open  
null

# Example : *FxCanvasExample2.java*

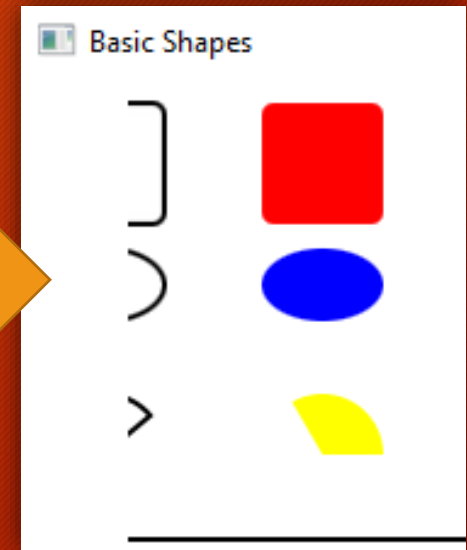
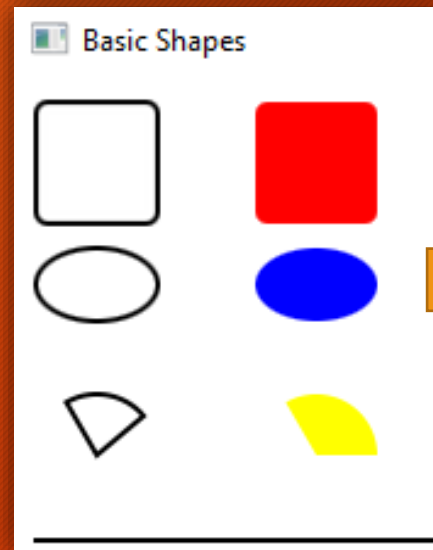
16

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Basic Shapes");

    Canvas canvas = new Canvas(400, 200);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);
    drawRoundRect(gc);
    drawOval(gc);
    drawArc(gc);
    drawLine(gc);

    clearRect(gc);

    stage.show();
}
```



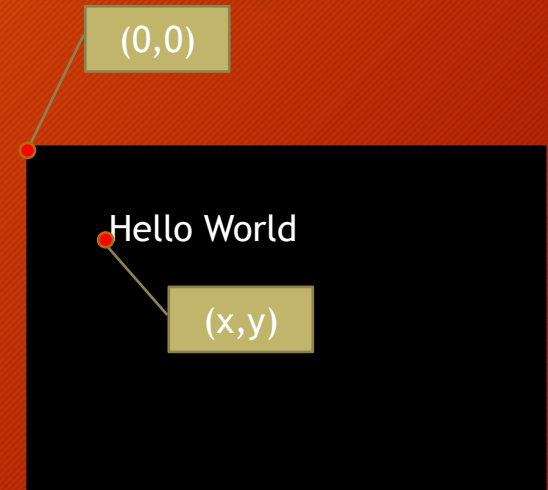
```
public void clearRect(GraphicsContext gc) {
    gc.clearRect(0, 0, gc.getCanvas().getWidth() / 9, gc.getCanvas().getHeight());
}
```

# Drawing - Text

17

- Text

- `setFont(Font f)`
- `getFont()`
- `fillText(Text t, int x, int y)`
- `fillText(Text t, int x, int y, double maxWidth)`
- `strokeText(Text t, int x, int y)`
- `strokeText(Text t, int x, int y, double maxWidth)`



- Font

- `Font.font(String family, FontWeight weight, FontPosture posture, int size)`
- More...

# Example : *FxCanvasExample3.java*

18

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Drawing - Text");

    Canvas canvas = new Canvas(800, 400);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);

    drawFilledText(gc);
    drawStrokedText(gc);
    drawText(gc);

    stage.show();
}
```

```
public void drawFilledText(GraphicsContext gc) {
    // Set line width
    gc.setLineWidth(2);
    // Set fill color
    gc.setFill(Color.RED);
    gc.setStroke(Color.BLACK);
    // set font
    Font theFont = Font.font("Times New Roman", FontWeight.LIGHT, 58);
    gc.setFont(theFont);

    // Draw a filled Text
    gc.fillText("This is a filled Text", 10, 75);

    //text,x,y,maxWidth - maximum width the text string can have.
    gc.fillText("This is a filled Text with Max Width 400 px", 10, 150, 400);
    gc.fillText("This is a filled Text with Max Width 400 pxxxxxxxxxxxxxxxxxxxxxxxxxxxx", 10, 225, 400);
}
```

```
public void drawText(GraphicsContext gc) {
    // Set line width
    gc.setLineWidth(2);
    // Set fill color
    gc.setFill(Color.RED);
    gc.setStroke(Color.BLUE);
    // set font
    Font theFont = Font.font("Times New Roman", FontWeight.LIGHT, 58);
    gc.setFont(theFont);

    // draw filled and stroked Text
    gc.fillText("This is a filled and stroked Text", 10, 375);
    gc.strokeText("This is a filled and stroked Text", 10, 375);
}
```

```
public void drawStrokedText(GraphicsContext gc) {
    // Set line width
    gc.setLineWidth(2);
    // Set fill color
    gc.setFill(Color.RED);
    gc.setStroke(Color.BLUE);
    // set font
    Font theFont = Font.font("Times New Roman", FontWeight.LIGHT, 58);
    gc.setFont(theFont);

    // Draw a Text
    gc.strokeText("This is a stroked Text", 10, 300);
}
```

This is a filled Text

This is a filled Text with Max Width 300 px

This is a filled Text with Max Width 300 pxxxxxxxxxxxxxxxxxxxxxxxxxxxx

This is a stroked Text

This is a filled and stroked Text

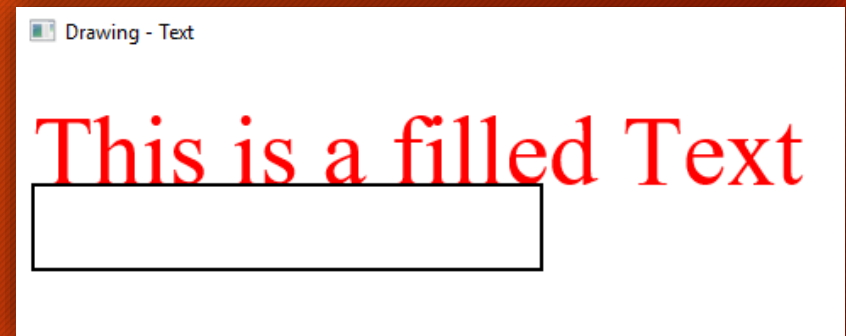


# Example :

## *FxCanvasExample3\_2.java*

19

```
public void drawFilledText(GraphicsContext gc) {  
    // Set line width  
    gc.setLineWidth(2);  
    // Set fill color  
    gc.setFill(Color.RED);  
    gc.setStroke(Color.BLACK);  
    // set font  
    Font theFont = Font.font("Times New Roman", FontWeight.LIGHT, 58);  
    gc.setFont(theFont);  
  
    // Draw a filled Text  
    gc.fillText("This is a filled Text", 10, 75);  
  
    gc.setLineWidth(2.0);  
    gc.setFill(Color.RED);  
    // Draw a rounded Rectangle  
    gc.strokeRect(10, 75, 100, 50);  
}
```



Look both have  
same position x,y

# Drawing - Text

20

- How to draw Text in Rect?
  - We need to get text width and height
  - -> use FontLoader



NO longer available!

# Drawing - Text

21

- `FontLoader fontLoader = Toolkit.getToolkit().getFontLoader();`
- Width
  - `double font_width = fontLoader.computeStringWidth("This is a test", fontLoader.getFont());`
- Height
  - `double font_height = fontLoader.getFontMetrics(fontLoader.getFont()).getHeight();`
  - For every text that use the same font

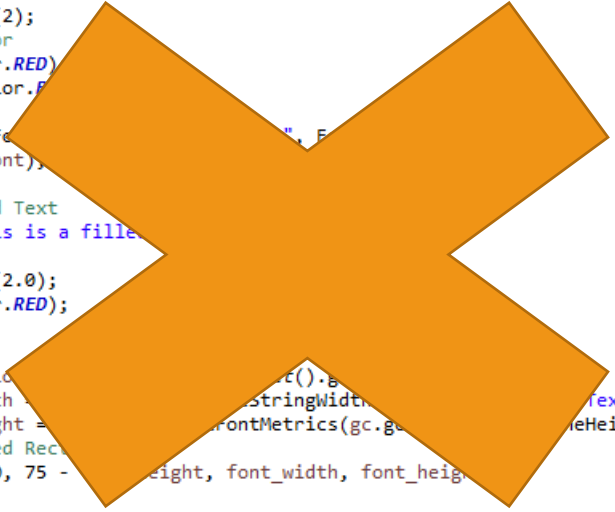
NO longer available!

# Example :

## *FxCanvasExample3\_3.java*

22

```
public void drawFilledText(GraphicsContext gc) {  
    // Set line width  
    gc.setLineWidth(2);  
    // Set fill color  
    gc.setFill(Color.RED);  
    gc.setStroke(Color.BLACK);  
    // set font  
    Font theFont = Font.getDefaultFont();  
    gc.setFont(theFont);  
  
    // Draw a filled Text  
    gc.fillText("This is a filled Text", 10, 75);  
  
    gc.setLineWidth(2.0);  
    gc.setFill(Color.RED);  
  
    FontLoader fontLoader = new FontLoader();  
    double font_width = fontLoader.getStringWidth("This is a filled Text", gc.getFont());  
    double font_height = fontLoader.getFontMetrics(gc.getFont()).getHeight();  
    // Draw a rounded Rectangle  
    gc.strokeRect(10, 75 - font_height, font_width, font_height);  
}
```




Drawing - Text

This is a filled Text

NO longer available!

```
// Draw a rounded Rectangle  
final Text text = new Text("This is a filled Text");  
text.setFont(theFont);  
double font_width = text.getLayoutBounds().getWidth();  
double font_height = text.getLayoutBounds().getHeight();  
  
gc.strokeRect(10, 75 - font_height, font_width, font_height);
```



# Drawing - Paths

23

- Paths
  - A path consists of multiple subpaths
  - **beginPath()**
    - Resets the current path to empty
  - **closePath()**
    - Closes the path
  - **moveTo()**, **lineTo()**, **quadraticCurveTo()**, **bezierCurveTo()**, **arc()**, **arcTo()**, **appendSVGPath()**, **rect()**
  - **stroke()**, **fill()**
    - draw an outline or fill the path



# Example : *FxCanvasExample4.java*

24

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Drawing - Paths");

    Canvas canvas = new Canvas(400, 400);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);

    drawLine(gc);
    drawCloseLine(gc);
    drawCurve(gc);

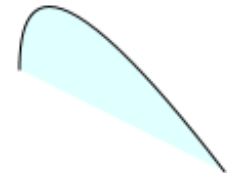
    stage.show();
}
```

```
public void drawLine(GraphicsContext gc) {
    // Start the Path
    gc.beginPath();
    // Make different Paths
    gc.moveTo(50, 50);
    gc.lineTo(100, 100);
    gc.lineTo(75, 100);
    // Draw the Path
    gc.stroke();
}
```

```
public void drawCurve(GraphicsContext gc) {
    // Start the Path
    gc.setFill(Color.LIGHTCYAN);
    gc.beginPath();
    // Make different Paths
    gc.moveTo(50, 300);
    gc.quadraticCurveTo(50, 220, 150, 350);
    gc.fill();
    // End the Path
    gc.stroke();
}
```

```
public void drawCloseLine(GraphicsContext gc) {
    // Start the Path
    gc.beginPath();
    // Make different Paths
    gc.moveTo(50, 150);
    gc.lineTo(100, 200);
    gc.lineTo(75, 200);
    // End the Path
    gc.closePath();
    // Draw the Path
    gc.stroke();
}
```

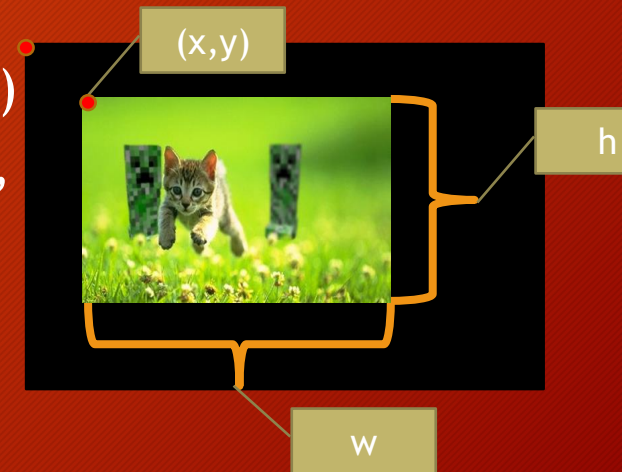
Basic Shapes



# Drawing - Images

25

- Can draw an Image on the Canvas using the drawImage() method
  - Can draw the whole or part of the Image
  - The drawn image can be stretched or shortened on the canvas
- 
- `void drawImage(Image img, double x, double y)`
  - `void drawImage(Image img, double x, double y, double w, double h)`



# Drawing - Images

26

- Image
  - new Image(InputStream is)
  - new Image(String url)
    - url = image path in pc or web url
  - new Image(String url,  
double requestedWidth,  
double requestedHeight,  
boolean preserveRatio,  
boolean smooth)

Use this to resize image

# Example : *FxCanvasExample5.java*

27

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Drawing - Images");

    Canvas canvas = new Canvas(800, 800);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);

    setBackground(gc);
    String image_path = "file:res/image/javafx_logo_color.jpg";
    drawImageFixSize(gc, image_path);
    drawImage(gc, image_path);

    stage.show();
}
```

```
public void drawImage(GraphicsContext gc, String image_path) {
    System.out.println(image_path);
    Image javafx_logo = new Image(image_path);
    gc.drawImage(javafx_logo, 40, 250);
}
```

```
public void drawImageFixSize(GraphicsContext gc, String image_path) {
    System.out.println(image_path);
    Image javafx_logo = new Image(image_path);
    gc.drawImage(javafx_logo, 40, 40, 600, 200);
}
```

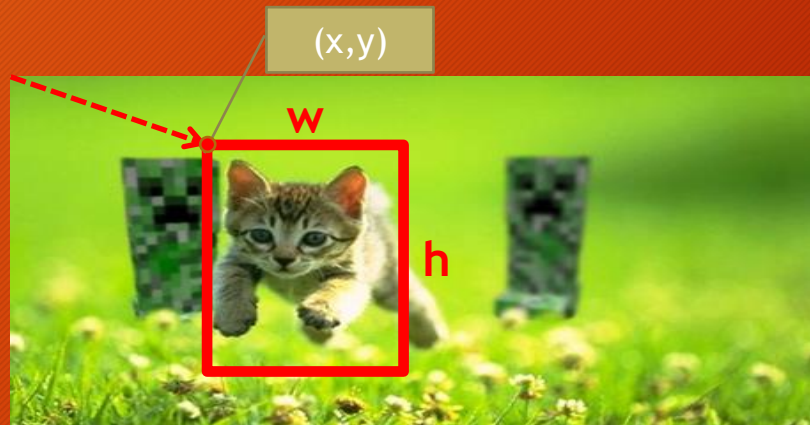




# Drawing - Images

28

- SubImage
  - WritableImage croppedImage = new WritableImage(image.getPixelReader(), x, y, h, w);
  - (x,y) is the rectangle's top-left related to image's top-left



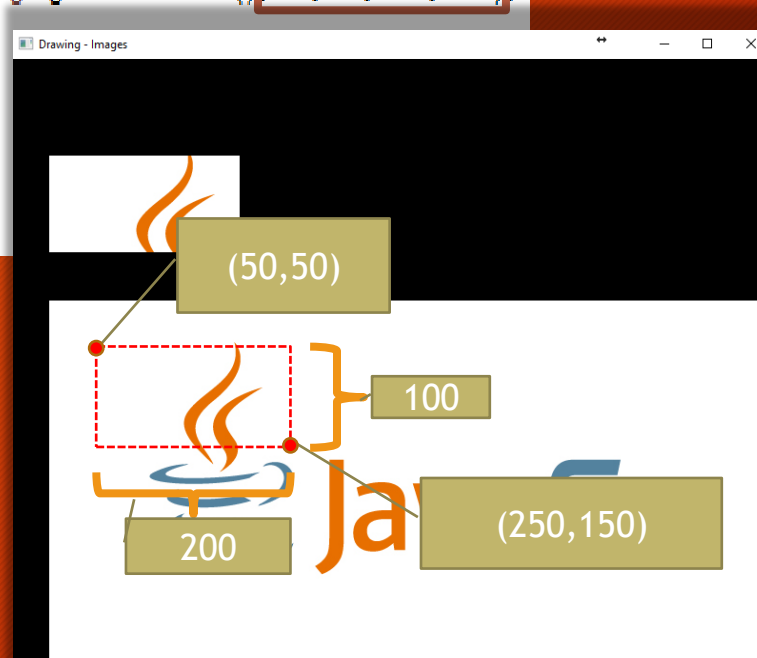


# Example :

## *FxCanvasExample5\_2.java*

29

```
public void drawCroppedImage(GraphicsContext gc, String image_path) {  
    System.out.println(image_path);  
    Image javafx_logo = new Image(image_path);  
    WritableImage croppedImage = new WritableImage(javafx_logo.getPixelReader(), 50, 50, 200, 100);  
    gc.drawImage(croppedImage, 40, 100);  
    gc.drawImage(javafx_logo, 40, 250);  
}
```

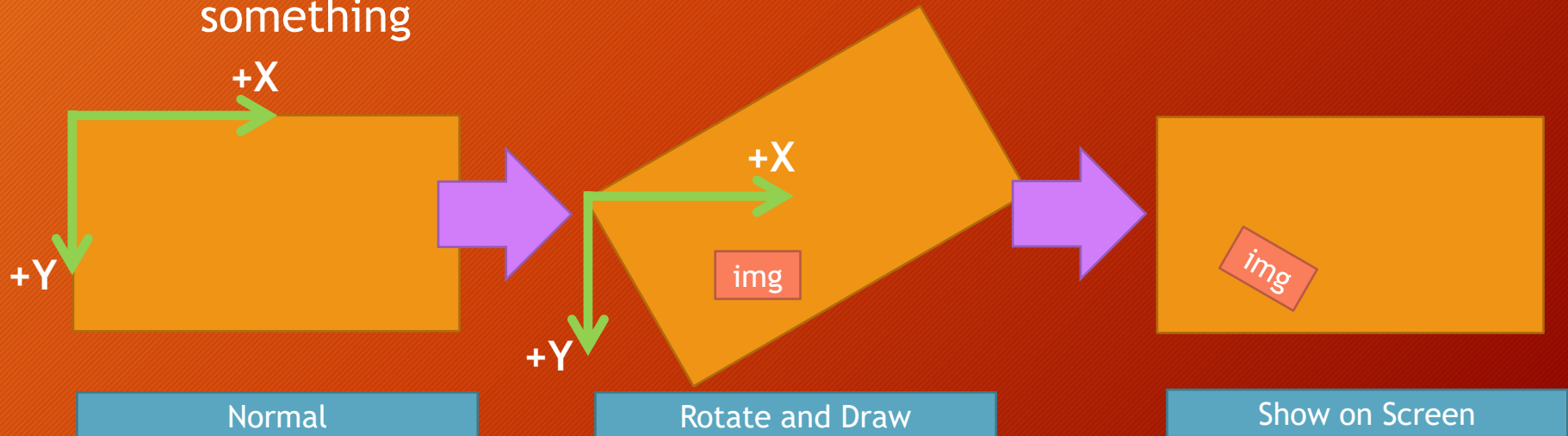


# Other Topic - Rotate

30

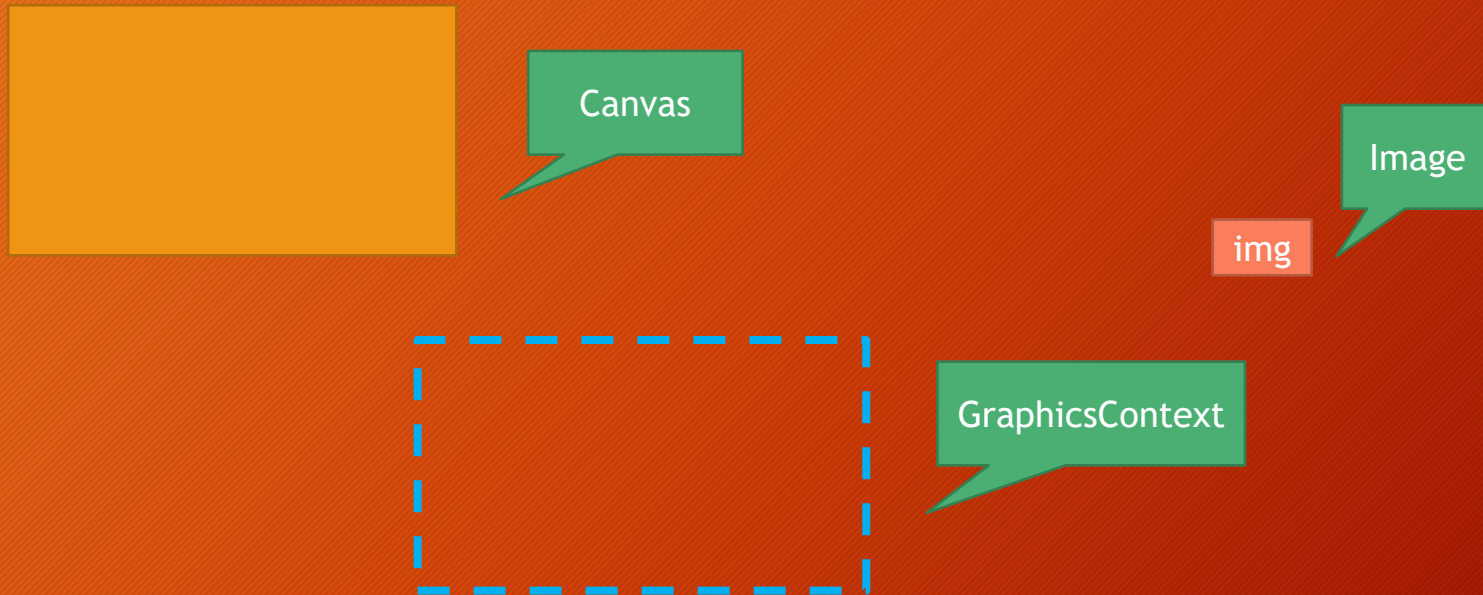
- Rotate

- `gc.rotate(double degrees)`
- `gc.drawImage(image, 100, 50)`
- It seem like rotate paper(Canvas) then draw something



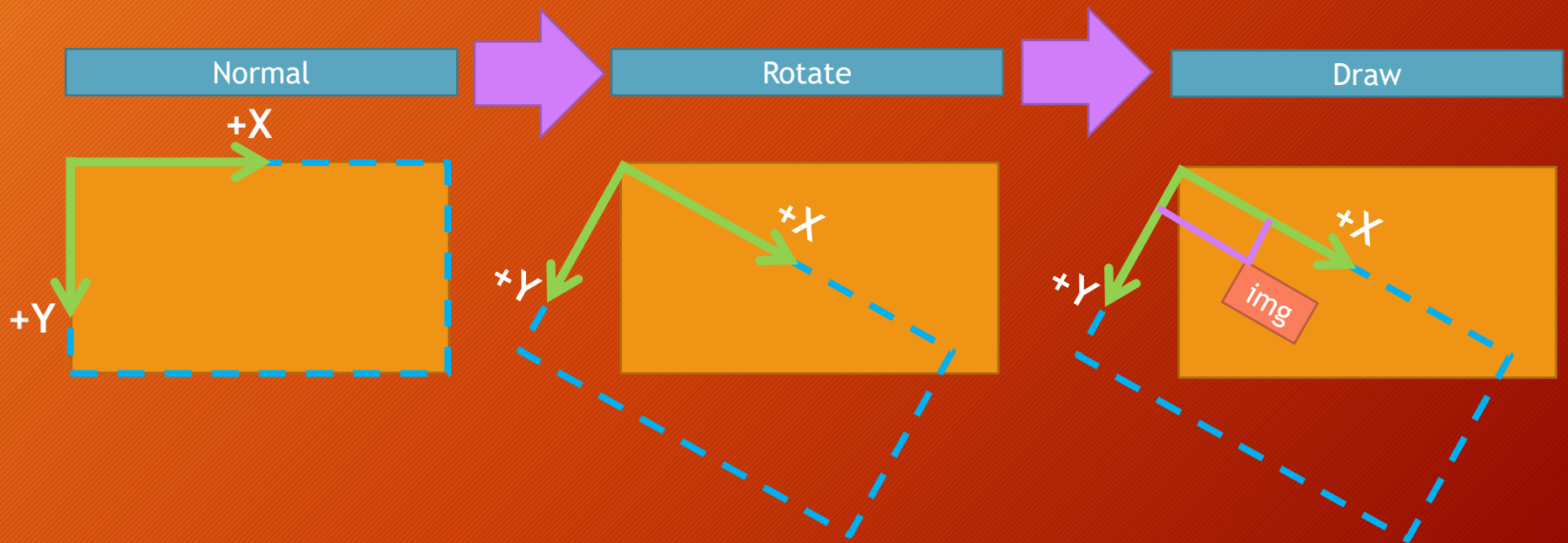
# Other Topic - Rotate

31



# Other Topic - Rotate

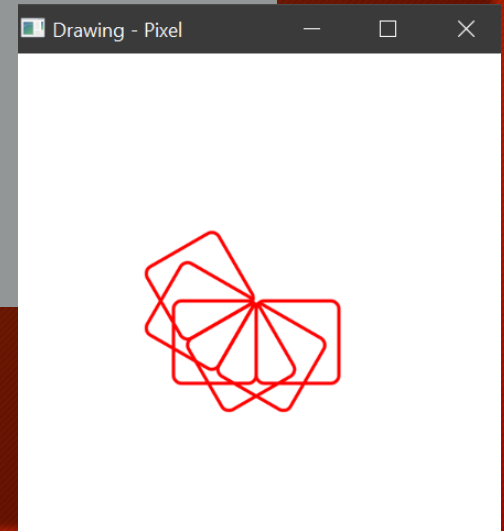
32



# Example: FxCanvasExample6.java

33

```
public void drawPixel(GraphicsContext gc) {  
    gc.setStroke(Color.RED);  
    gc.setLineWidth(2.0);  
    gc.translate(gc.getCanvas().getWidth() / 2, gc.getCanvas().getHeight() / 2);  
  
    for(int i = 0; i < 6; i++){  
        //gc.fillRoundRect(500, 100, 50, 50, 10, 10);  
        //gc.strokeRoundRect(400 + i * 50, 100, 50, 50, 10, 10);  
        gc.strokeRoundRect(0, 0, 50, 50, 10, 10);  
        gc.rotate(30);  
    }  
}
```





# Example : FxCanvasExample8.java

34

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("Drawing - Images");

    Canvas canvas = new Canvas(800, 800);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);

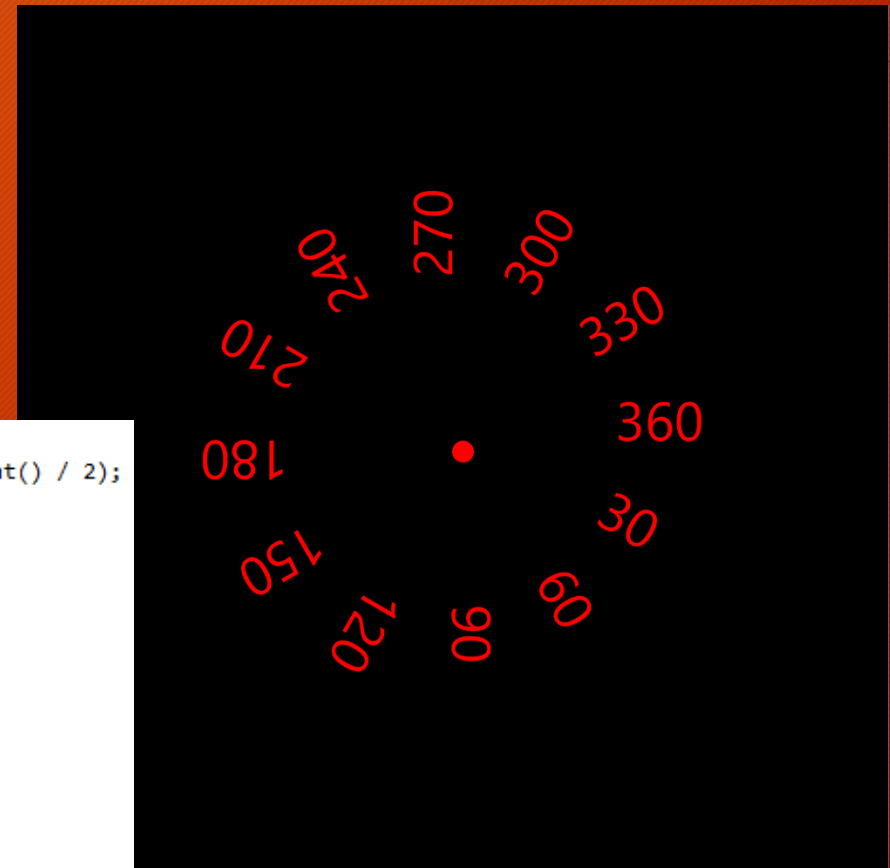
    setBackground(gc);
    drawRotatedText(gc);

    stage.show();
}
```

```
public void drawRotatedText(GraphicsContext gc) {
    gc.translate(gc.getCanvas().getWidth() / 2, gc.getCanvas().getHeight() / 2);
    gc.setFont(Font.font(50));
    gc.setFill(Color.RED);

    gc.fillOval(0, 0, 20, 20);

    int total_angle = 0;
    int angle = 30;
    while (total_angle < 360) {
        total_angle += angle;
        gc.rotate(angle);
        gc.fillText("" + total_angle, 150, 0);
    }
    gc.restore();
}
```



# Example : JAVA\_FX\_TankGame - Tank

35

```
public void update() {
    if (flashing) {
        if (flashCounter == 0) {
            this.visible = true;
            flashing = false;
        } else {
            if (flashDurationCounter > 0) {
                this.visible = flashCounter <= 5;
                flashDurationCounter--;
            } else {
                this.visible = true;
                flashDurationCounter = 10;
                flashCounter--;
            }
        }
    } else {
        this.visible = !InputUtility.getKeyPressed(KeyCode.SHIFT);
    }
    if (InputUtility.getKeyPressed(KeyCode.W)) {
        forward();
    }
    if (InputUtility.getKeyPressed(KeyCode.A)) {
        turn(true);
    }
    else if (InputUtility.getKeyPressed(KeyCode.D)) {
        turn(false);
    }
    if (InputUtility.isLeftClickTriggered()) {
        this.x = InputUtility.mouseX;
        this.y = InputUtility.mouseY;
    }
}
```

- This code receive input from user
- W -> go forward
- A -> turn left x degree
- D -> turn right x degree

```
@Override
public void draw(GraphicsContext gc) {
    gc.setFill(Color.BLUE);
    gc.fillArc(x - radius, y - radius, radius * 2, radius * 2, 0, 360, ArcType.OPEN);
    gc.translate(x, y);
    gc.rotate(angle);
    gc.setFill(Color.YELLOW);

    int gunSize = radius / 5;
    gc.fillRect(0, -gunSize, radius * 3 / 2, gunSize * 2);
    gc.rotate(-angle);
    gc.translate(-x, -y);
}
```

# AnimationTimer

# AnimationTimer

37

- make our programs *dynamic*, meaning that the game state changes over time
- implement a game loop: an infinite loop that updates the game objects and renders the scene to the screen
- AnimationTimer will be called at a rate of 60 times per second or as close to that rate as is possible

# Example : *FxCanvasExample7.java*

38

```
@Override
public void start(Stage stage) {
    StackPane root = new StackPane();
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("AnimationTimer");

    Canvas canvas = new Canvas(800, 400);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    root.getChildren().add(canvas);

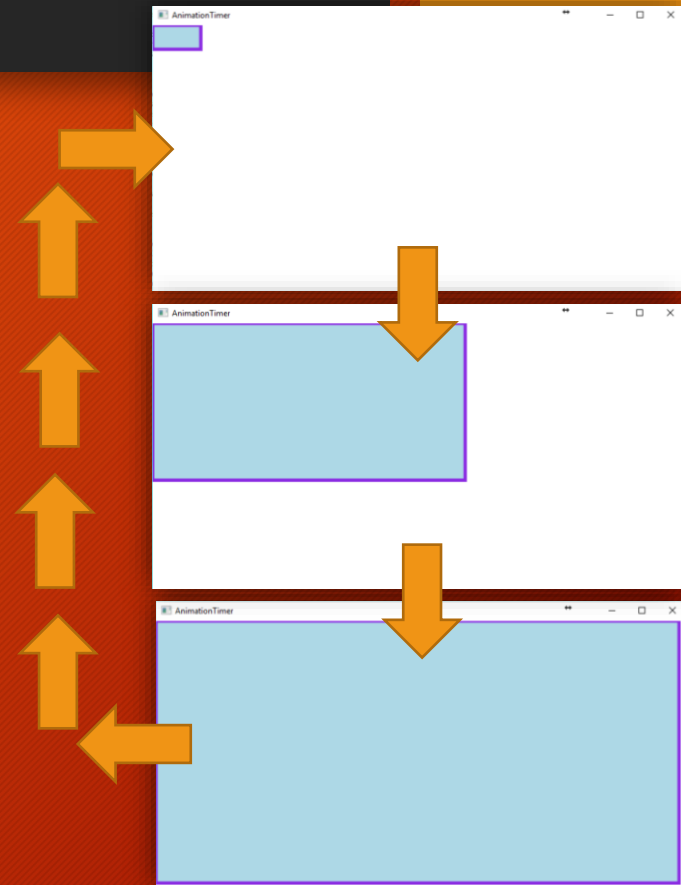
    drawScalableRectAnimation(gc);

    stage.show();
}
```

```
public void drawScalableRectAnimation(GraphicsContext gc) {
    final long startNanoTime = System.nanoTime();
    new AnimationTimer() {
        double width = 0;
        double height = 0;

        public void handle(long currentNanoTime) {
            double t = ((currentNanoTime - startNanoTime) / 1000000000.0) % 3;
            width = gc.getCanvas().getWidth() * t / 3;
            height = gc.getCanvas().getHeight() * t / 3;

            gc.setFill(Color.LIGHTBLUE);
            gc.setStroke(Color.BLUEVIOLET);
            gc.setLineWidth(5);
            gc.clearRect(0, 0, gc.getCanvas().getWidth(), gc.getCanvas().getHeight());
            gc.fillRect(0, 0, width, height);
            gc.strokeRect(0, 0, width, height);
        }
    }.start();
}
```





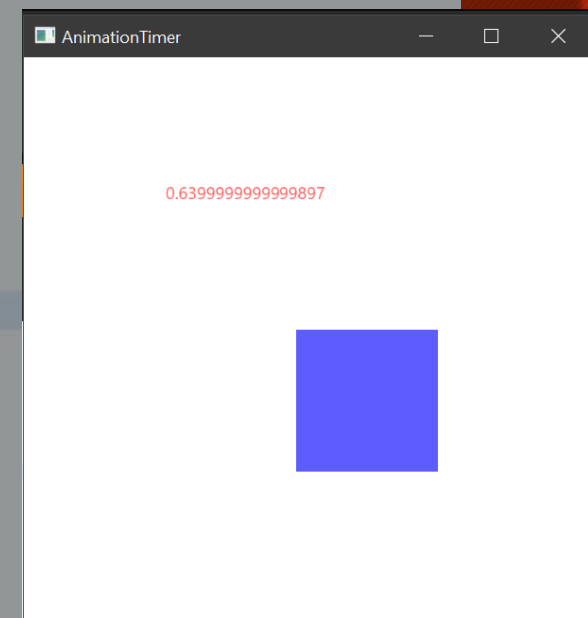
# Example: FxCanvasExample7-2.java

## AnimationTimer example 2

39

```
public void drawRectanglesAnimation(GraphicsContext gc) {
    new AnimationTimer() {
        double alpha = 0.0;
        boolean increase = true;

        public void handle(long currentNanoTime) {
            double t = 200;
            if (increase)
                alpha += 1.0 / t;
            else
                alpha -= 1.0 / t;
            if (alpha > 1.0) {
                alpha = 2 - alpha;
                increase = false;
            } else if (alpha < 0.0) {
                alpha = Math.abs(alpha);
                increase = true;
            }
            gc.setFill(Color.BLUE);
            gc.setGlobalAlpha(alpha);
            gc.clearRect(0, 0, gc.getCanvas().getWidth(), gc.getCanvas().getHeight());
            gc.fillRect(0 + alpha * 300, 0 + alpha * 300, 100, 100);
            gc.setFill(Color.RED);
            gc.fillText(alpha + "", 100, 100);
        }
    }.start();
}
```





Design

# Design

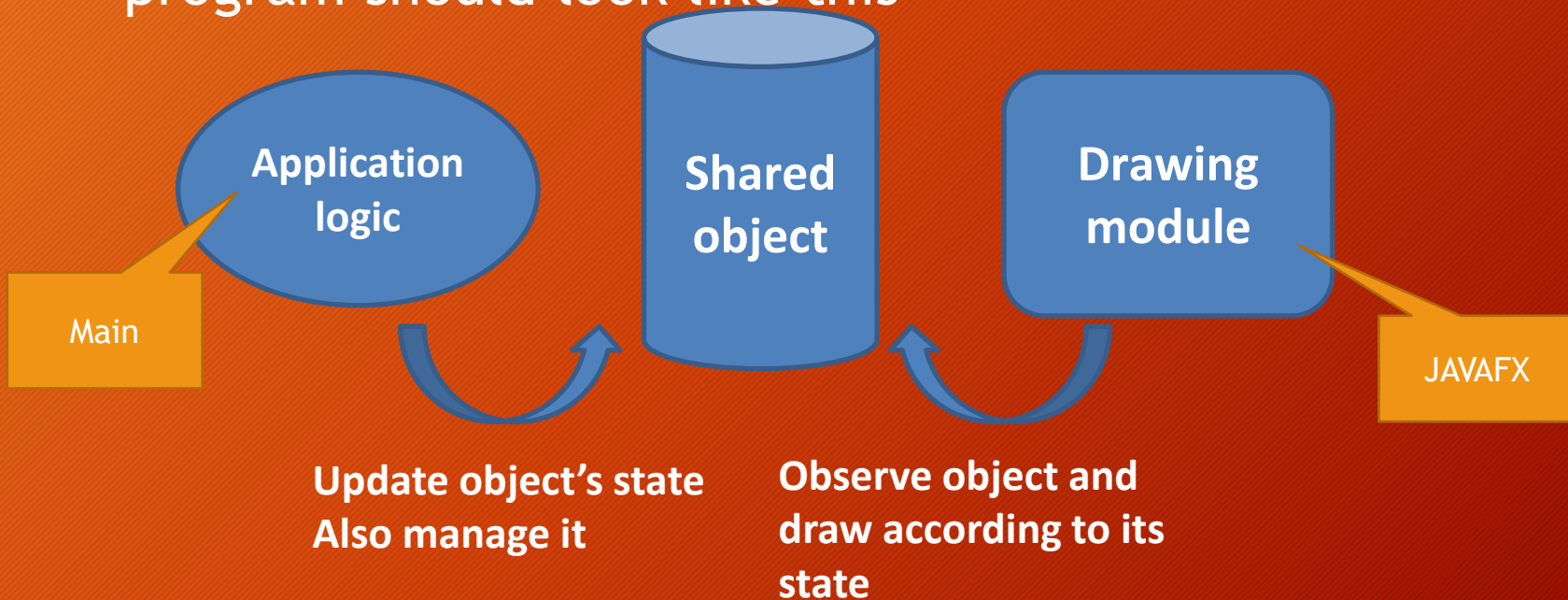
41

- Normally, we want to **draw** many **objects** on our screen
- Objects with different shapes should have different drawing methods
- At the same time, aside from drawing, we might want to **update** those objects' state

# Design Component

42

- With requirements in the previous slide, the program should look like this



# Design Component

43

- Shared Object
- Drawing Part
- Logic Part



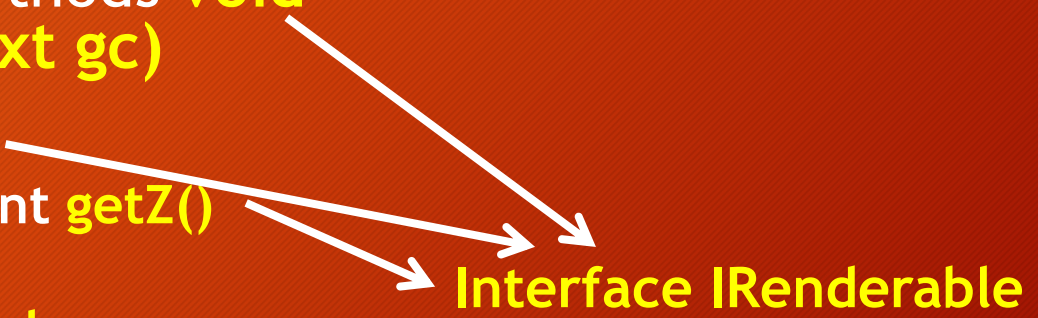
# Design Component

44

- **Shared Object**
- Drawing Part
- Logic Part

# Shared Object - Requirement

45

- Back to our requirements
  - Objects with different shapes should have different drawing methods **void draw(GraphicsContext gc)**
  - Objects may overlap
    - We need “ordering” int **getZ()**
    - **Z=9 -> Foreground**
    - **Z=-999 -> Background**
  - Add more capabilities to the object
    - Able to hide/show **boolean isVisible()**
- 
- The diagram shows three arrows pointing from specific requirements to the text 'Interface IRenderable'. The first arrow originates from the text 'void draw(GraphicsContext gc)' in the second bullet point. The second arrow originates from the text 'getZ()' in the sub-bullet of the third bullet point. The third arrow originates from the text 'Z=-999 -> Background' in the sub-bullet of the third bullet point.

# Shared Object - IRenderable

46

```
public interface IRenderable {  
    public int getZ();  
    public void draw(GraphicsContext gc);  
    public boolean isVisible();  
}
```

Polymorphism

Ordering

Draw

Hide/Show

# Shared Object - RenderableHolder

47

- We want to draw many objects on our screen
  - The application logic also access these objects
- A *shared* Collection of generic interface (IRenderable)

For example:

- ArrayList<IRenderable>
- LinkedList<IRenderable>

Anything to keep track of “all” IRenderable

# Shared Object - RenderableHolder

48

Collection

```
public class RenderableHolder_Basic {  
    private List<IRenderable> entities;  
  
    public RenderableHolder_Basic() {  
        entities = new ArrayList<IRenderable>();  
    }  
}
```

Constructor



# Shared Object - RenderableHolder - Sort

49

- RenderableHolder class
  - Any collection with <IRenderable> ArrayList<IRenderable>
  - Methods for accessing the collection
    - The collection is used to keep track of objects that have to be drawn on the screen
    - IRenderable in the collection **must be sorted by Z** (So you can just loop through the collection and draw the deepest one first)

# Shared Object - RenderableHolder - Sort

50

- RenderableHolder class
  - Any collection with <IRenderable>  
ArrayList<IRenderable>
  - Methods for accessing the collection ->  
add(IRenderable)
  - - add(IRenderable)  
add new IRenderable object to the list and sort the list  
(according to Z)

# Shared Object - RenderableHolder - Sort

51

```
public class RenderableHolder {  
    private List<IRenderable> entities;  
    private Comparator<IRenderable> comparator;  
  
    public RenderableHolder() {  
        entities = new ArrayList<IRenderable>();  
        comparator = (IRenderable o1, IRenderable o2) -> {  
            if (o1.getZ() > o2.getZ())  
                return 1;  
            return -1;  
        };  
    }  
  
    public void add(IRenderable entity) {  
        entities.add(entity);  
        // Sort our list by Z  
        Collections.sort(entities, comparator);  
    }  
}
```

Use to sort IRenderable order by Z

```
public void update() {  
    for (int i = entities.size() - 1; i >= 0; i--) {  
        if (entities.get(i).isDestroyed())  
            entities.remove(i);  
    }  
}
```

Add Irenderable to ArrayList  
then sort using comparator

# Shared Object - RenderableHolder - Singleton

52

- We can declare our shared collection as a field
- BUT:
  - According to our design, the collection does not belong to either the logic part (application logic) or the drawing part (drawing module)
  - We only need a single collection to store everything that must be drawn
- We encapsulate our collection in RenderableHolder class and use a **singleton pattern** to ensure that there is **only one instance of RenderableHolder**

# Shared Object - RenderableHolder - Singleton

53

- Singleton pattern
  - A design pattern for a class that can be instantiated to only one object
  - Good when the program only need a single object shared across every part

```
private static final RenderableHolder instance = new RenderableHolder();  
public static RenderableHolder getInstance(){  
    return instance;  
}
```



# Shared Object - RenderableHolder - Singleton

54

```
public class RenderableHolder_Singleton {  
    private List<IRenderable> entities;  
    private Comparator<IRenderable> comparator;  
    private static final RenderableHolder instance = new RenderableHolder();  
  
    public RenderableHolder_Singleton() {  
        entities = new ArrayList<IRenderable>();  
        comparator = (IRenderable o1, IRenderable o2) -> {  
            if (o1.getZ() > o2.getZ())  
                return 1;  
            return -1;  
        };  
    }  
  
    public void add(IRenderable entity) {  
        entities.add(entity);  
        // Sort our list by Z  
        Collections.sort(entities, comparator);  
    }  
  
    public static RenderableHolder getInstance(){  
        return instance;  
    }  
}
```

# Example : JAVA\_FX\_TankGame - GameScreen

55

This is how to use RenderableHolder from other Classes.

```
public class GameScreen extends Canvas {

    public GameScreen(double width, double height) {
        super(width, height);
        this.setVisible(true);
        addListener();
    }

    public void addListener() {}

    public void paintComponent() {
        GraphicsContext gc = this.getGraphicsContext2D();
        gc.setFill(Color.BLACK);
        for (IRenderable entity : RenderableHolder.getInstance().getEntities()) {
            if (entity.isVisible() && !entity.isDestroyed()) {
                entity.draw(gc);
            }
        }
    }
}
```

This class wants to iterate all entities

# Example : JAVA\_FX\_TankGame - GameLogic

56

This is how to use  
RenderableHolder  
from other Classes.

```
public class GameLogic {  
    private List<Entity> gameObjectContainer;  
  
    private Tank tank;  
    private Mine mine;  
  
    public GameLogic(){  
        this.gameObjectContainer = new ArrayList<Entity>();  
  
        Field field = new Field();  
        RenderableHolder.getInstance().add(field);  
        tank = new Tank(520,240);  
        mine = new Mine(100,100);  
        addNewObject(tank);  
        addNewObject(mine);  
    }  
  
    protected void addNewObject(Entity entity){  
        gameObjectContainer.add(entity);  
        RenderableHolder.getInstance().add(entity);  
    }  
  
    public void logicUpdate(){  
        tank.update();  
        if(!mine.isDestroyed() && tank.collideWith(mine)){  
            mine.onCollision(tank);  
        }  
    }  
}
```

This class wants to add entities

# Shared Object - RenderableHolder - LoadResource

57

- Resource holder
  - Used to keep required resources in the memory
  - Resources are either pre-loaded or loaded on use
  - **uses static initializer to load all resources on the first use**

# Shared Object - RenderableHolder - LoadResource

58

```
public class RenderableHolder {  
    private static final RenderableHolder instance = new RenderableHolder();
```

```
    private List<IRenderable> entities;  
    private Comparator<IRenderable> comparator;  
    public static Image mapSprite;  
    public static Image mineSprite;  
    public static AudioClip explosionSound;
```

```
    static {  
        LoadResource();  
    }
```

```
    public RenderableHolder() {}
```

```
    public static RenderableHolder getInstance() {}
```

```
    public static void loadResource() {  
        mapSprite = new Image(ClassLoader.getResource("Map.png").toString());  
        mineSprite = new Image(ClassLoader.getResource("Mine.png").toString());  
        explosionSound = new AudioClip(ClassLoader.getResource("Explosion.wav").toString());  
    }
```

```
    public void add(IRenderable entity) {}
```

Load Resource once  
Then  
Use Everywhere



# Example : JAVA\_FX\_TankGame - RenderableHolder

59

```
public class RenderableHolder {  
    private static final RenderableHolder instance = new RenderableHolder();  
  
    private List<IRenderable> entities;  
    private Comparator<IRenderable> comparator;  
    public static Image mapSprite;  
    public static Image mineSprite;  
    public static AudioClip explosionSound;  
  
    static {  
        LoadResource();  
    }  
  
    public RenderableHolder() {  
        entities = new ArrayList<IRenderable>();  
        comparator = (IRenderable o1, IRenderable o2) -> {  
            if (o1.getZ() > o2.getZ())  
                return 1;  
            return -1;  
        };  
    }  
  
    public static RenderableHolder getInstance() {  
        return instance;  
    }  
}
```

Singleton

Load resource only 1 time

How to sort

```
public static void loadResource() {  
    mapSprite = new Image(ClassLoader.getResource("Map.png").openStream());  
    mineSprite = new Image(ClassLoader.getResource("Mine.png").openStream());  
    explosionSound = new AudioClip(ClassLoader.getResource("explosion.wav").openStream());  
}  
  
public void add(IRenderable entity) {  
    System.out.println("add");  
    entities.add(entity);  
    Collections.sort(entities, comparator);  
}  
  
public void update() {  
    for (int i = entities.size() - 1; i >= 0; i--) {  
        if (entities.get(i).isDestroyed())  
            entities.remove(i);  
    }  
}  
  
public List<IRenderable> getEntities() {  
    return entities;  
}
```

Add entity and then  
sort

Remove dead entity

# Design Component

60

- Shared Object
- **Drawing Part**
- Logic Part

# Drawing Part

61

- Drawing Part
  - A simple subclass of Canvas that look at the shared collection and draw all objects

# Example : JAVA\_FX\_TankGame - GameScreen

62

```
public class GameScreen extends Canvas {  
  
    public GameScreen(double width, double height) {  
        super(width, height);  
        this.setVisible(true);  
        addListener();  
    }  
  
    public void addListener() {}  
  
    public void paintComponent() {  
        GraphicsContext gc = this.getGraphicsContext2D();  
        gc.setFill(Color.BLACK);  
        for (IRenderable entity : RenderableHolder.getInstance().getEntities()) {  
            if (entity.isVisible() && !entity.isDestroyed()) {  
                entity.draw(gc);  
            }  
        }  
    }  
}
```

# Design Component

63

- Shared Object
- Drawing Part
- **Logic Part**

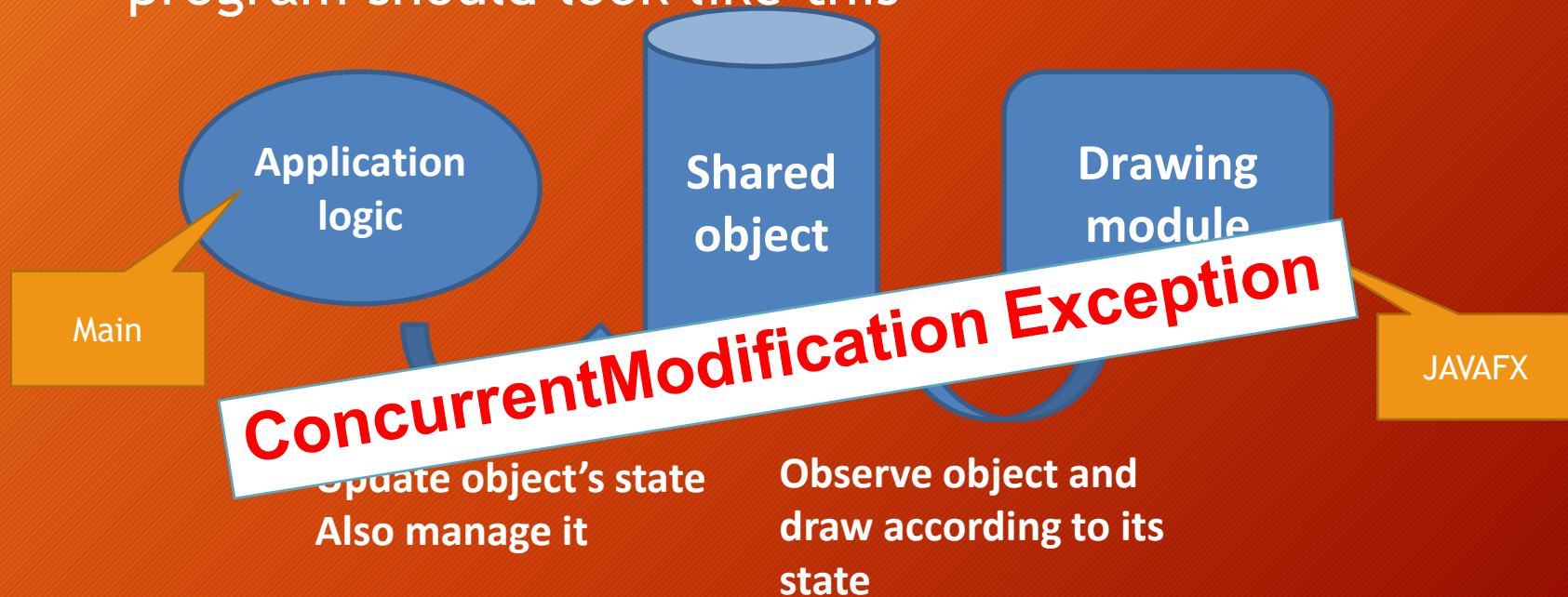


- Logic Part
  - This part is responsible for creating/removing objects and updating their state
  - When creating an object, if it must be drawn to the screen (that object is an IRenderable), add it to RenderableHolder
  - When removing an object, if it was added to RenderableHolder, don't forget to remove it from the holder

# Design Component

65

- With requirements in the previous slide, the program should look like this



# Logic Part

66

```
AnimationTimer animation = new AnimationTimer() {  
    public void handle(long now) {  
        gameScreen.paintComponent();  
        logic.logicUpdate();  
        RenderableHolder.getInstance().update();  
        InputUtility.updateInputState();  
    }  
};  
animation.start();
```

## Drawing Module

- Need to loop all Renderable.list to draw all object

## Logic Module

- Need to loop all Renderable.list to update object

- Sometime, Logic Module access list while Drawing module have not finished draw all object
- This will cause Concurrent Modification Exception

Logic Module

Drawing Module

Time

```
public class ConcurrentModificationException  
    extends RuntimeException
```

This exception may be thrown by methods that have detected concurrent modification of an object when such modification is not permissible.

# Logic Part

67

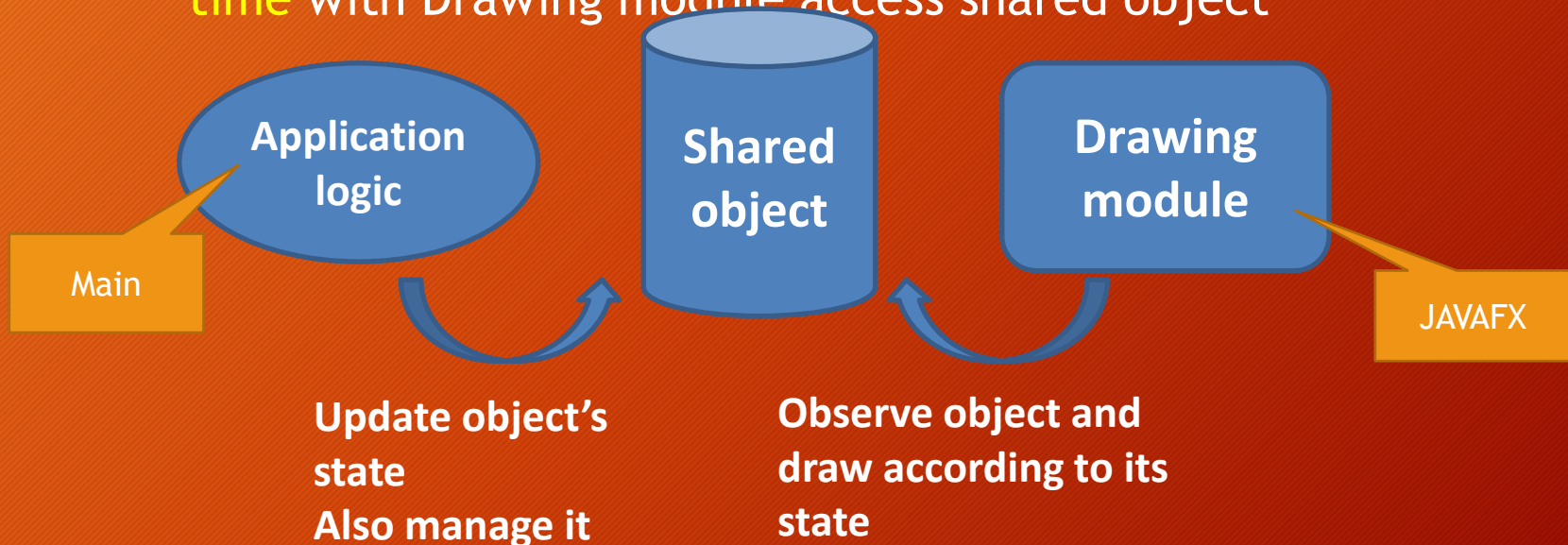
- To update objects that are added to the holder, there are 2 approaches:
  1. Loop through all objects in the holder and update any object needed (casting required)
  2. Cache local references of those objects and update through these references



# Logic Part - Update Object

68

- Approach 1 : Problem
  - Application logic access shared object at the **same time** with Drawing module access shared object





# Logic Part - Update Object

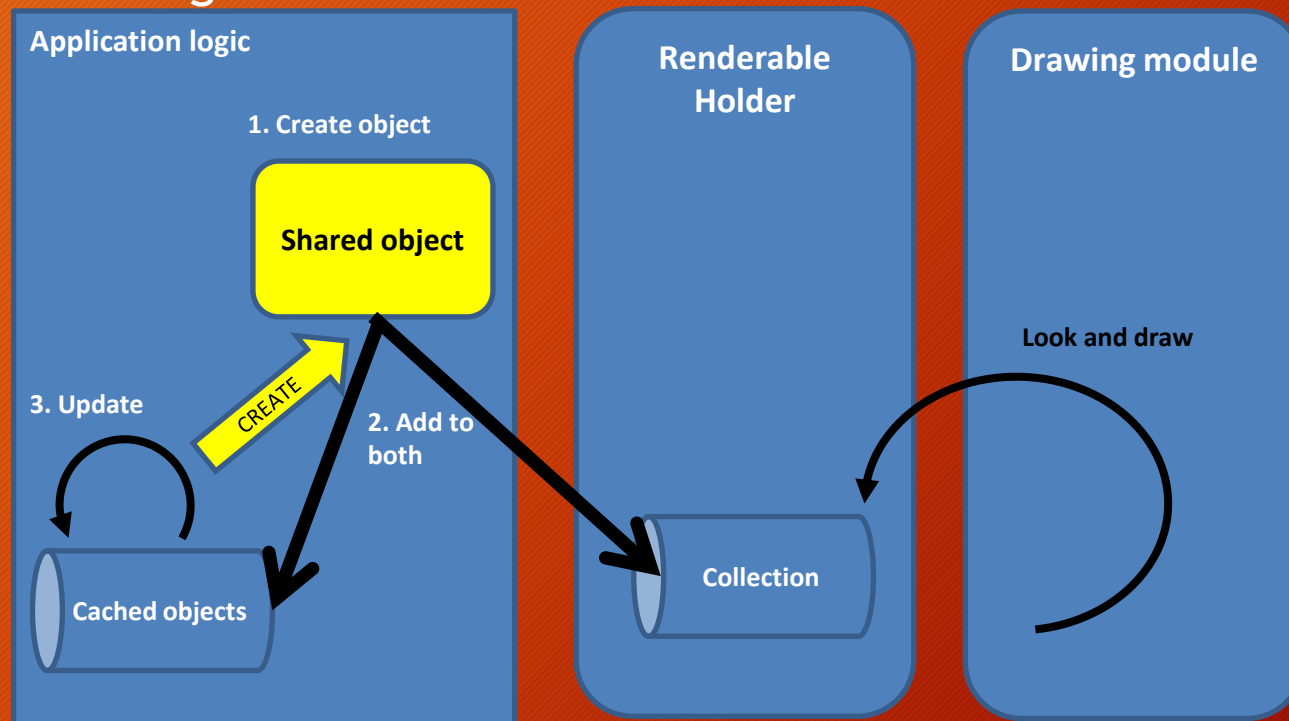
69

- Approach 1 :
  - Loop through all objects in the holder and update any object needed
- Problem :
  - Application logic access shared object at the same time with Drawing module access shared object
- Solution :
  - using synchronize (Not in our scope!)

# Logic Part - Update Object

70

- Approach 2 :
  - Cache local references of those objects and update through these references



# Example : JAVA\_FC\_TankGame - GameLogic

71

- Create object
- Add same object in both Lists.

```
public class GameLogic {
    private List<Entity> gameObjectContainer;

    private Tank tank;
    private Mine mine;

    public GameLogic(){
        this.gameObjectContainer = new ArrayList<Entity>();

        Field field = new Field();
        RenderableHolder.getInstance().add(field);
        tank = new Tank(320,240);
        mine = new Mine(100,100);
        addNewObject(tank);
        addNewObject(mine);
    }

    protected void addNewObject(Entity entity){
        gameObjectContainer.add(entity);
        RenderableHolder.getInstance().add(entity);
    }

    public void logicUpdate(){
        tank.update();
        if(!mine.isDestroyed() && tank.collideWith(mine)){
            mine.onCollision(tank);
        }
    }
}
```

# Handling User Input

# Handling User Input

73

- Detecting and processing user input in JavaFX is straightforward
- User actions that can be detected by the system, such as key presses and mouse clicks, are called *events*
- Any JavaFX class which implements the EventTarget class, such as a Scene, can "listen" for events and handle them



# Handling User Input

74

- There are many methods that listen for handling different types of input from different sources
- `setOnKey.....()` can assign an `EventHandler` that will activate when a key is .....
- `setOnMouse..... ()` can assign an `EventHandler` that activates when a mouse button is .....
- The `EventHandler` class serves one purpose: to encapsulate a method (called `handle()`) that is called when the corresponding event occurs.

# Handling User Input Example

75

- Example of Key Event

```
this.setOnKeyPressed((KeyEvent event) -> {  
    InputUtility.setKeyPressed(event.getCode(), true);  
});  
  
this.setOnKeyReleased((KeyEvent event) -> {  
    InputUtility.setKeyPressed(event.getCode(), false);  
});
```

# Handling User Input Example

76

- Example of Mouse Event

```
this.setOnMousePressed((MouseEvent event) -> {  
    if (event.getButton() == MouseButton.PRIMARY)  
        InputUtility.mouseLeftDown();  
});
```

```
this.setOnMouseReleased((MouseEvent event) -> {  
    if (event.getButton() == MouseButton.PRIMARY)  
        InputUtility.mouseLeftRelease();  
});
```

```
this.setOnMouseEntered((MouseEvent event) -> {  
    InputUtility.mouseOnScreen = true;  
});
```

```
this.setOnMouseExited((MouseEvent event) -> {  
    InputUtility.mouseOnScreen = false;  
});
```

```
this.setOnMouseMoved((MouseEvent event) -> {  
    if (InputUtility.mouseOnScreen) {  
        InputUtility.mouseX = event.getX();  
        InputUtility.mouseY = event.getY();  
    }  
});
```

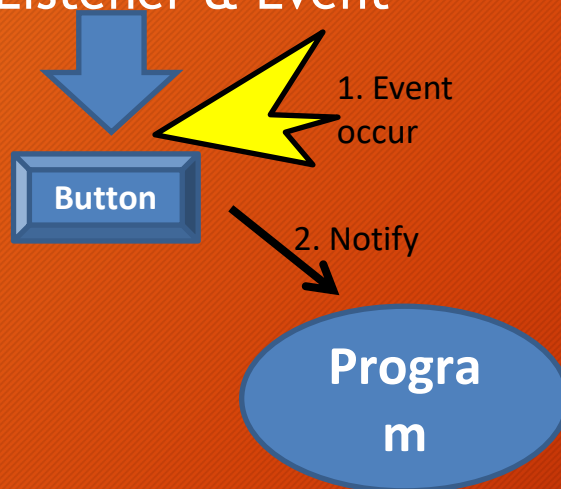
```
this.setOnMouseDragged((MouseEvent event) -> {  
    if (InputUtility.mouseOnScreen) {  
        InputUtility.mouseX = event.getX();  
        InputUtility.mouseY = event.getY();  
    }  
});
```

# Input handling

77

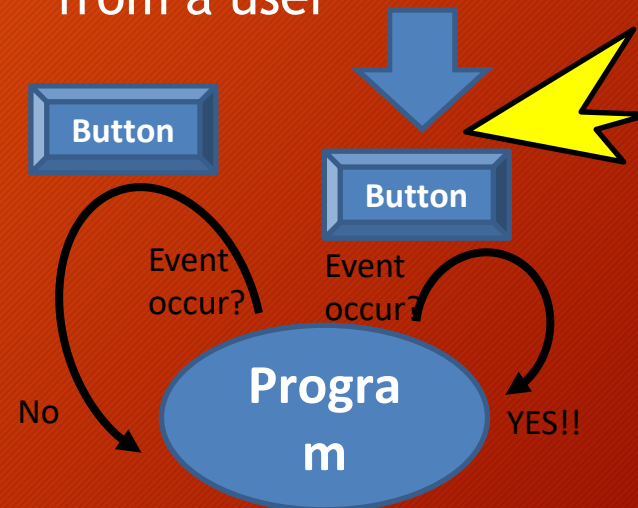
## Event-driven

- A program handle an event immediately
- Listener & Event



## Polling

- Periodically check if there is an input from a user



# Input handling

78

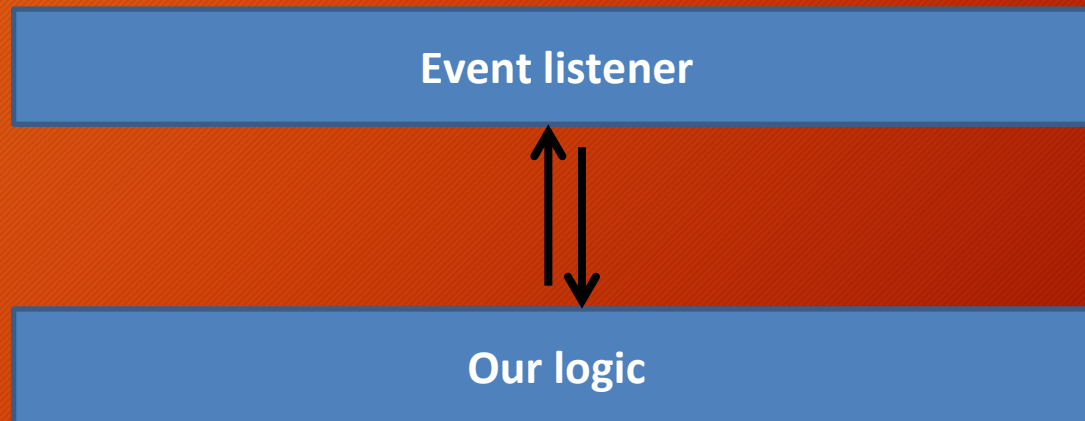
- JVM provides user's input to JavaFX application via callback method in a listener which is an **event-driven method**
- This means we don't not know when the input is coming
- In contrast, the input checking code requires the exact time to operate!
- Therefore, it is required to poll the incoming event which is **polling method**



# Input handling - Event-driven

79

- Event-driven
  - JVM provides user's input to JavaFX application via callback method in a listener
  - This means we don't not know when the input is coming



# Example : JAVA\_FX\_InputHandling\_Event\_Driven

80

```
public class Main extends Application {  
    public static void main(String[] args) {  
        Application.launch(args);  
    }  
  
    @Override  
    public void start(Stage stage) {  
        StackPane root = new StackPane();  
        GameScreen gameScreen = new GameScreen(root);  
        stage.setScene(gameScreen);  
        stage.setTitle("Click click click");  
  
        gameScreen.redraw("");  
        stage.show();  
  
        gameScreen.setOnKeyPressed((KeyEvent e) -> {  
            String new_code = e.getCode().toString();  
            gameScreen.redraw(new_code);  
        });  
    }  
}
```

Trigger -> draw

```
public class GameScreen extends Scene {  
    private Canvas canvas;  
  
    public GameScreen(Pane parent) {  
        super(parent);  
  
        canvas = new Canvas(420, 200);  
        parent.getChildren().add(canvas);  
    }  
  
    public void redraw(String code){  
        GraphicsContext gc = canvas.getGraphicsContext2D();  
        gc.setFill(Color.BLACK);  
        gc.setFont(Font.font(40));  
        gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());  
        gc.fillText("TEST SetOnKeyPressed", 10, 50);  
        gc.fillText(code, 200, 100);  
    }  
}
```

# Input handling - Polling

81

- Polling
  - the input checking code requires the exact placement in the application code. Therefore, it is required to poll the incoming event
- The idea
  - When an event takes place, notes it down somewhere
  - Our logic polls the noted event



# Input handling - Polling

82

- Polling example

```
While(true){  
    Sleep for 20 ms  
    Draw screen  
    Update logic (poll the event here)  
}
```

```
If (users press ENTER){  
    Do action 1  
}  
If(users press SPACE){  
    Do action 2  
}
```

# Example : JAVA\_FX\_InputHandling\_Polling

83

```
public class Main extends Application {  
    public static void main(String[] args) {  
        Application.launch(args);  
    }  
  
    @Override  
    public void start(Stage stage) {  
        StackPane root = new StackPane();  
        GameScreen gameScreen = new GameScreen(root);  
        stage.setScene(gameScreen);  
        stage.setTitle("Click click click");  
  
        gameScreen.redraw();  
        stage.show();  
  
        gameScreen.setOnKeyPressed((KeyEvent e) -> {  
            String new_code = e.getCode().toString();  
            CodeUtility.receiveInput(new_code);  
        });  
  
        new AnimationTimer() {  
            public void handle(long now) {  
                gameScreen.redraw();  
            }  
        }.start();  
    }  
}
```

Trigger -> store value

Loop to redraw

```
public class CodeUtility {  
    public static String code = "";  
    public static int counter = 0;  
  
    public static void receiveInput(String new_code){  
        if(code.equalsIgnoreCase(new_code)) counter++;  
        else counter = 1;  
        code = new_code;  
    }  
}  
  
public class GameScreen extends Scene {  
    private Canvas canvas;  
  
    public GameScreen(Pane parent) {  
        super(parent);  
  
        canvas = new Canvas(420, 200);  
        parent.getChildren().add(canvas);  
    }  
  
    public void redraw(){  
        GraphicsContext gc = canvas.getGraphicsContext2D();  
        gc.setFill(Color.BLACK);  
        gc.setFont(Font.font(40));  
        gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());  
        gc.fillText("TEST SetOnKeyPressed", 10, 50);  
        gc.fillText(CodeUtility.code, 200, 100);  
    }  
}
```

Get code value

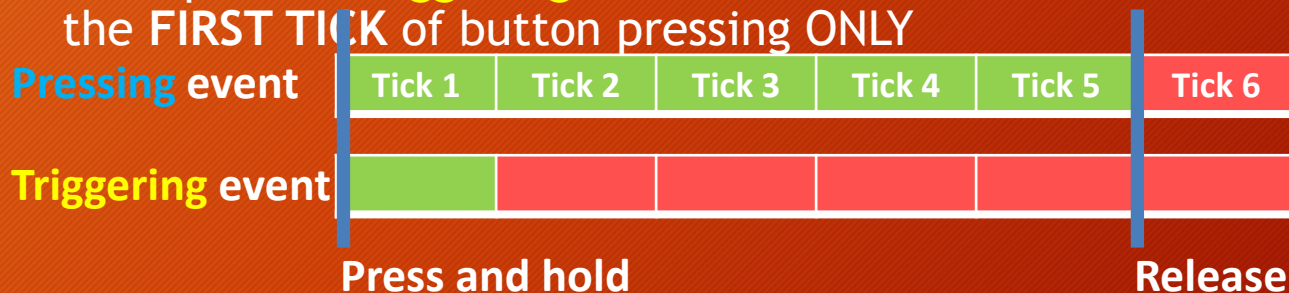


# Input handling - Pressing VS Triggering

84

- Pressing VS Triggering

- When user press and hold a button
  - If we poll for **pressing** event : it must remain TRUE until user releases the button
  - If we poll for **triggering** event : it must be TRUE on the **FIRST TICK** of button pressing **ONLY**



# Example : JAVA\_FX\_Key

85

```
public class Main extends Application {
    public static void main(String[] args) {
        Application.launch(args);
    }

    @Override
    public void start(Stage stage) {
        StackPane root = new StackPane();
        GameScreen gameScreen = new GameScreen(root);
        stage.setScene(gameScreen);
        stage.setTitle("Click click click");

        stage.show();

        gameScreen.setOnKeyPressed((KeyEvent e) -> {
            String new_code = e.getCode().toString();
            CodeUtility.receiveInput(new_code);
        });

        AnimationTimer timer = new AnimationTimer() {
            public void handle(long now) {
                gameScreen.redraw();
            }
        };
        timer.start();
    }
}
```

```
public class CodeUtility {
    public static String code = "";
    public static int counter = 0;

    public static void receiveInput(String new_code){
        if(code.equalsIgnoreCase(new_code)) counter++;
        else counter = 1;
        code = new_code;
    }
}
```

```
public class GameScreen extends Scene {
    private Canvas canvas;

    public GameScreen(Pane parent) {
        super(parent);

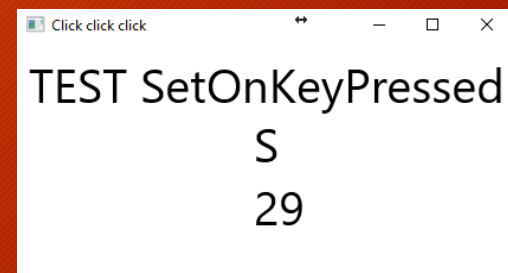
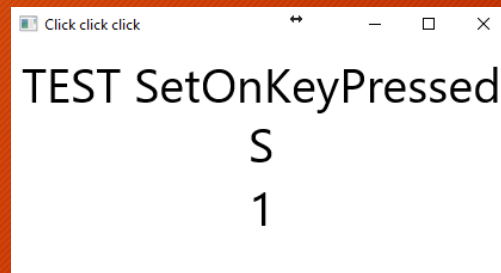
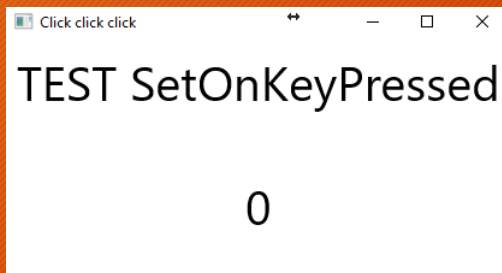
        canvas = new Canvas(420, 200);
        parent.getChildren().add(canvas);
    }

    public void redraw(){
        GraphicsContext gc = canvas.getGraphicsContext2D();
        gc.setFill(Color.BLACK);
        gc.setFont(Font.font(40));
        gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
        gc.fillText("TEST SetOnKeyPressed", 10, 50);
        gc.fillText(CodeUtility.code + "\n" + CodeUtility.counter, 200, 100);
    }
}
```

# Example : JAVA\_FX\_Key

86

- This happens when we press “S” and hold



# Input handling - Pressing VS Triggering

87

- Implementation to poll for **pressing** and **triggering** event
  - Take note of **pressing** and **triggering** event separately when “pressed” event happens
  - **Pressing** flag changes to FALSE on button releasing
  - **Triggering** flag changes to FALSE at the end of every tick

# Example : JAVA\_FX\_KeyUpgrade

88

```
public static void setPressed(boolean pressed) {  
    if (pressed) {  
        CodeUtility.pressed = true;  
    } else {  
        CodeUtility.pressed = false;  
    }  
}
```

```
public static void setTriggered(String code, boolean pressed) {  
    if (pressed) {  
        CodeUtility.triggered = true;  
        if (CodeUtility.code.equals(code))  
            counter++;  
        else {  
            CodeUtility.code = code;  
            counter = 1;  
        }  
    } else {  
        CodeUtility.triggered = false;  
    }  
}
```

```
public class Main extends Application {  
    public static void main(String[] args) {  
        Application.launch(args);  
    }  
}
```

@Override

```
public void start(Stage stage) {  
    StackPane root = new StackPane();  
    GameScreen gameScreen = new GameScreen(root);  
    stage.setScene(gameScreen);  
    stage.setTitle("Click click click");  
  
    stage.show();  
  
    gameScreen.setOnKeyPressed((KeyEvent event) -> {  
        String new_code = event.getCode().toString();  
        if (!CodeUtility.getPressed())  
            CodeUtility.setTriggered(new_code, true);  
        CodeUtility.setPressed(true);  
    });  
  
    gameScreen.setOnKeyReleased((KeyEvent event) -> {  
        CodeUtility.setPressed(false);  
    });  
  
    AnimationTimer timer = new AnimationTimer() {  
        public void handle(long now) {  
            gameScreen.redraw();  
            CodeUtility.postUpdate();  
        }  
    };  
    timer.start();  
}
```



# Example : JAVA\_FX\_KeyUpgrade

89

Action	Code	Counter	Pressed	Triggered
Start Program		0	False	False
Press S	S	1	True	True
CodeUtility.postUpdate()	S	1	True	False
Hold S	S	1	True	False
CodeUtility.postUpdate()	S	1	True	False
Release S	S	1	False	False
CodeUtility.postUpdate()	S	1	False	False
Press S	S	2	True	True
CodeUtility.postUpdate()	S	2	True	False

# Example : JAVA\_FX\_KeyUpgrade

90

- This happens when we press “S” continuously

Start Program

Press S

Hold S

Release S

Press S

Click click click

TEST SetOnKeyPressed

0

Click click click

TEST SetOnKeyPressed

S

1

Click click click

TEST SetOnKeyPressed

S

1

Click click click

TEST SetOnKeyPressed

S

1

Click click click

TEST SetOnKeyPressed

S

2

# Input handling - Pressing VS Triggering

91

- When we have multiple input at the same time
- One way to accomplish this is by creating an ArrayList of String objects
- When a key is initially pressed, we add the String representation of the KeyEvent's KeyCode to the list
- When the key is released, we remove it from the list.

# Input handling - Pressing VS Triggering

92

- Implement method
  - Getter and setter of keyPressed, KeyTriggered and KeyTriggerFlag yourself

(><)// wish u can do it by yourself

```
public class InputUtility {  
  
    private static int mouseX, mouseY;  
    private static boolean mouseLeftDown, mouseRightDown, mouseOnScreen;  
    private static boolean mouseLeftLastDown, mouseRightLastDown;  
    private static ArrayList<KeyCode> keyPressed = new ArrayList<>();  
    private static ArrayList<KeyCode> keyTriggered = new ArrayList<>();  
    private static ArrayList<KeyCode> keyTriggerFlag = new ArrayList<>();  
}
```

# Input handling - Pressing VS Triggering

93

- Triggering event: Keyboard VS Mouse
  - MouseListener fires “mousePressed” event only once when mouse button is pressed and held
  - KeyListener fires “keyPressed” event continuously (about every tick) as long as a button is pressed and held
    - Hold key? we only note down triggering event on the **first tick of button holding** : The tick that “pressed” flag change from FALSE to TRUE



# Example : JAVA\_FX\_Mouse

94

- Details of the program:
  - Click to increase the counter
  - Click & hold considers as “1 click”
  - The program should differentiate between “click” vs. “click & hold”

```
AnimationTimer timer = new AnimationTimer() {  
    public void handle(long now) {  
        gameScreen.redraw();  
        if(MouseUtility.isLeftClickTriggered()){  
            MouseUtility.counter++;  
        }  
        MouseUtility.postUpdate();  
    }  
};  
timer.start();
```

```
gameScreen.setOnMouseClicked(new EventHandler<MouseEvent>() {  
    public void handle(MouseEvent event) {  
        if(event.getButton() == MouseButton.PRIMARY)  
            MouseUtility.mouseLeftDown();  
    }  
});  
  
gameScreen.setOnMouseReleased(new EventHandler<MouseEvent>() {  
    public void handle(MouseEvent event) {  
        if(event.getButton() == MouseButton.PRIMARY)  
            MouseUtility.mouseLeftRelease();  
    }  
});
```

# Input handling - Mouse

95

- This happens when we hold “left click”

Start Program

Left Click

Hold Left Click

Release  
Left Click

Left Click

TEST Mouse

0

TEST Mouse

1

TEST Mouse

1

TEST Mouse

1

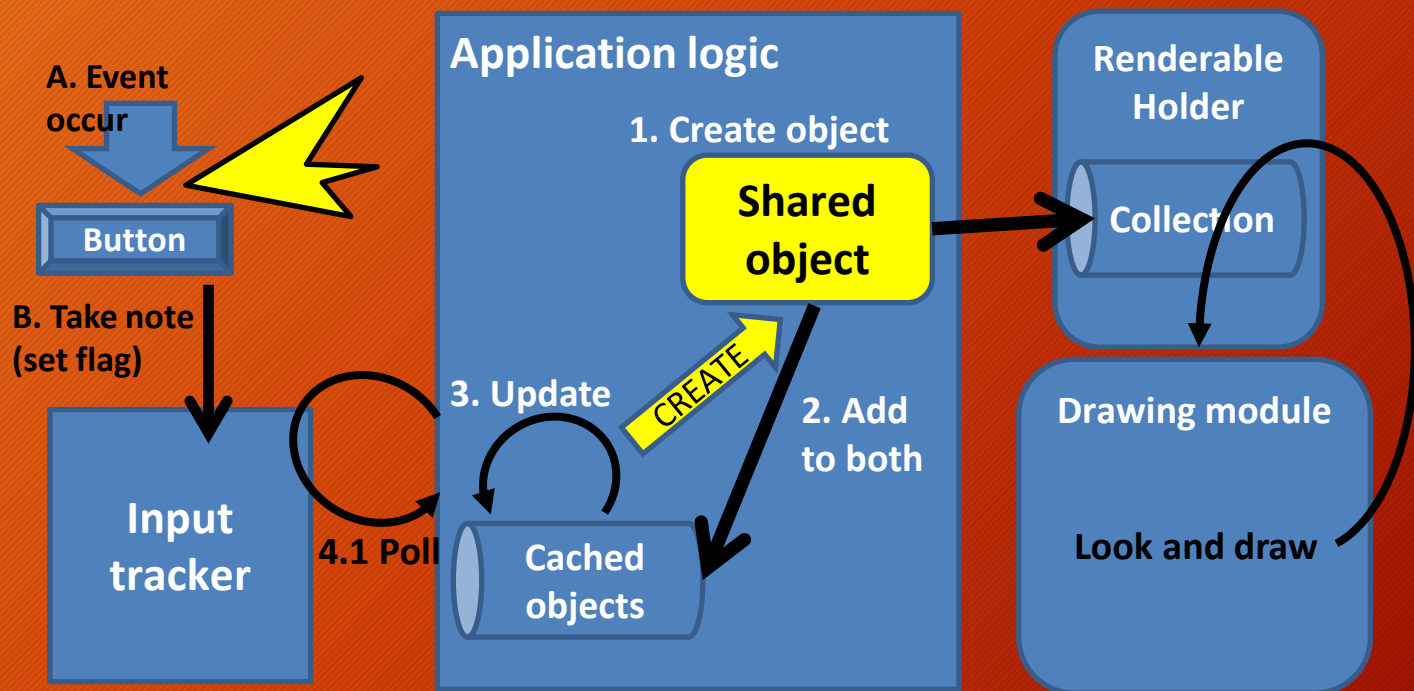
TEST Mouse

2

# Graphics + Input

# Put everything together

97

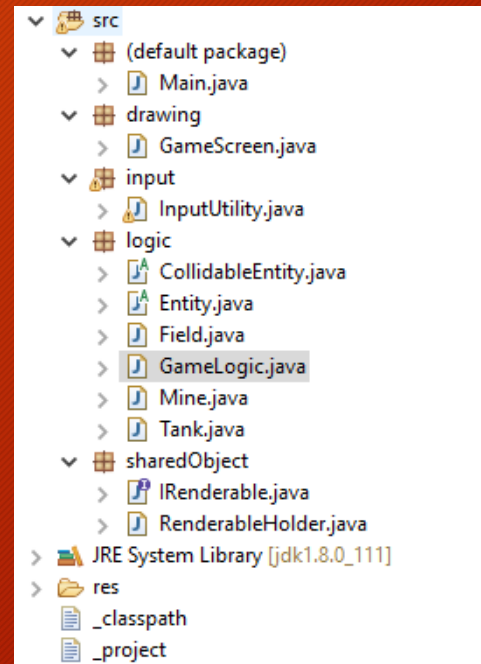
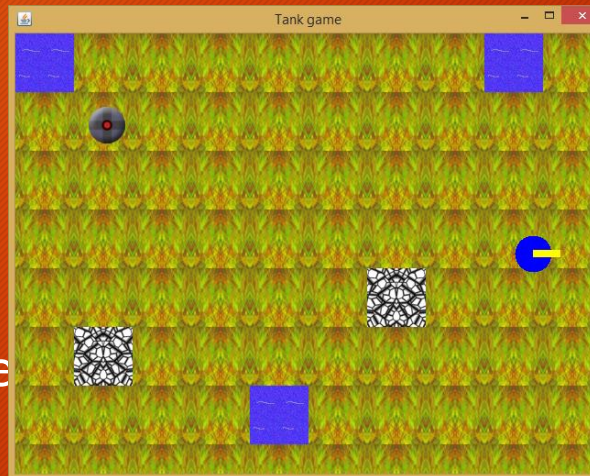




# Put everything together

98

- Put everything together : `JAVA_FX_TankGame`
  - Controllable tank
    - W: Forward
    - A: Turn left
    - D: Turn right
    - Shift: Hide
    - Click: Warp
- The code
  - Drawing the battlefield
  - A moving tank





# Input Handling - JAVA\_FX\_TankGame

99

- GameScreen Class

Key Event

```
this.setOnKeyPressed((KeyEvent event) -> {  
    InputUtility.setKeyPressed(event.getCode(), true);  
});  
  
this.setOnKeyReleased((KeyEvent event) -> {  
    InputUtility.setKeyPressed(event.getCode(), false);  
});
```

Mouse Event

```
this.setOnMousePressed((MouseEvent event) -> {  
    if (event.getButton() == MouseButton.PRIMARY)  
        InputUtility.mouseLeftDown();  
});  
  
this.setOnMouseReleased((MouseEvent event) -> {  
    if (event.getButton() == MouseButton.PRIMARY)  
        InputUtility.mouseLeftRelease();  
});  
  
this.setOnMouseEntered((MouseEvent event) -> {  
    InputUtility.mouseOnScreen = true;  
});  
  
this.setOnMouseExited((MouseEvent event) -> {  
    InputUtility.mouseOnScreen = false;  
});  
  
this.setOnMouseMoved((MouseEvent event) -> {  
    if (InputUtility.mouseOnScreen) {  
        InputUtility.mouseX = event.getX();  
        InputUtility.mouseY = event.getY();  
    }  
});  
  
this.setOnMouseDragged((MouseEvent event) -> {  
    if (InputUtility.mouseOnScreen) {  
        InputUtility.mouseX = event.getX();  
        InputUtility.mouseY = event.getY();  
    }  
});
```

Audio

# Java sound

101

- New JAVA sound API in JavaFX
  - `javafx.scene.media.AudioClip`
- Constructor
  - `AudioClip sound = new AudioClip(String source)`
- Very easy to use
  - `sound.play`
  - `sound.stop()`
  - `sound.setCycle()`
  - `sound.setVolume()`

# Example : JAVA\_FX\_Sound

102

```
public class Main extends Application {
    public static void main(String[] args) {
        Application.Launch(args);
    }

    @Override
    public void start(Stage stage) {
        StackPane root = new StackPane();
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("AnimationTimer");

        Canvas canvas = new Canvas(800, 400);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        root.getChildren().add(canvas);

        gc.setFill(Color.BLACK);
        gc.fillRect(0, 0, canvas.getWidth(), canvas.getHeight());

        AudioClip sound = new AudioClip("file:res/audio/Meow.wav");

        scene.setOnMouseClicked(new EventHandler<MouseEvent>() {
            public void handle(MouseEvent event) {
                createCat(gc);
                createCat(gc);
                createCat(gc);
                sound.play();
            }
        });

        stage.show();
    }

    public void createCat(GraphicsContext gc) {
        int random = (int) (Math.random() * 5 + 1);
        Image image = new Image("file:res/image/cat" + random + ".jpg", 100, 100, false, false);
        double width = Math.random() * gc.getCanvas().getWidth();
        double height = Math.random() * gc.getCanvas().getHeight();
        gc.drawImage(image, width, height);
    }
}
```

But It can't be  
exported!! You  
need to use  
GetSystemResource

# Example : JAVA\_FX\_Sound\_Fix - Main

103

```
public class Main extends Application {
    public static void main(String[] args) {
        Application.launch(args);
    }

    @Override
    public void start(Stage stage) {
        StackPane root = new StackPane();
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("AnimationTimer");

        Canvas canvas = new Canvas(800, 400);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        root.getChildren().add(canvas);

        gc.setFill(Color.BLACK);
        gc.fillRect(0, 0, canvas.getWidth(), canvas.getHeight());

        System.out.println(ClassLoader.getResource("audio/Meow.wav").toString());
        AudioClip sound = new AudioClip(ClassLoader.getResource("audio/Meow.wav").toString());

        scene.setOnMouseClicked(new EventHandler<MouseEvent>() {
            public void handle(MouseEvent event) {
                createCat(gc);
                createCat(gc);
                createCat(gc);
                sound.play();
            }
        });

        stage.show();
    }

    public void createCat(GraphicsContext gc) {
        int random = (int) (Math.random() * 5 + 1);
        System.out.println(ClassLoader.getResource("image/cat" + random + ".jpg").toString());
        Image image = new Image(ClassLoader.getResource("image/cat" + random + ".jpg").toString(), 100, 100, false, false);
        double width = Math.random() * gc.getCanvas().getWidth();
        double height = Math.random() * gc.getCanvas().getHeight();
        gc.drawImage(image, width, height);
    }
}
```



# Put everything together

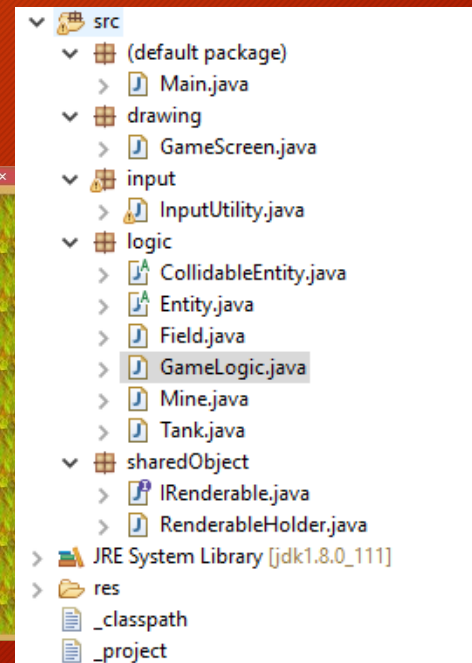
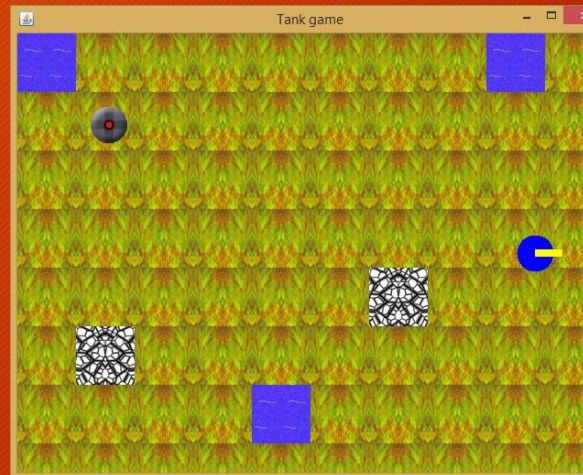
104

- Put everything together : JAVA\_FX\_TankGame

- Move tank to the same position of mine
- It will cause Explosion sound

- The code

- Drawing the battlefield
- A moving tank



# Java sound - JAVA\_FX\_TankGame

105

```
public class Mine extends CollidableEntity{

    public Mine(int x,int y){
        this.x = x;
        this.y = y;
        this.z = -100;
        this.radius = 20;
    }

    public void onCollision(Tank tank){
        tank.hitByMine();
        RenderableHolder.explosionSound.play();
        this.destroyed = true;
    }
}
```

```
public class RenderableHolder {
    private static final RenderableHolder instance = new RenderableHolder();

    private List<IRenderable> entities;
    private Comparator<IRenderable> comparator;
    public static Image mapSprite;
    public static Image mineSprite;
    public static AudioClip explosionSound;

    static {
        loadResource();
    }

    public RenderableHolder() {}

    public static RenderableHolder getInstance() {}

    public static void loadResource() {
        mapSprite = new Image(ClassLoader.getResource("Map.png").toString());
        mineSprite = new Image(ClassLoader.getResource("Mine.png").toString());
        explosionSound = new AudioClip(ClassLoader.getResource("Explosion.wav").toString());
    }
}
```

# Conclusion

# What you've learned

107

- Drawing on Canvas
- Input polling based on listener & event
- (Very simple) audio playback
- Application design pattern example

Last Suggestion

108

Ask friend, then  
Google, or gpt



- <https://openjfx.io/javadoc/22/javafx.graphics/javafx/scene/canvas/Canvas.html>
- <https://openjfx.io/javadoc/22/javafx.graphics/javafx/scene/canvas/GraphicsContext.html>
- <http://zetcode.com/gui/javafx/canvas/>
- <https://gamedevelopment.tutsplus.com/tutorials/introduction-to-javafx-for-game-development--cms-23835> (maybe too old)
- <https://jaxenter.com/tutorial-a-glimpse-at-javafxs-canvas-api-105696.html> (maybe too old)
- <https://examples.javacodegeeks.com/desktop-java/javafx/javafx-canvas-example/> (Maybe too old)