Leibniz
Universität
Hannover

Interactive Systems (ISY)

Auditorium Exercise 04

Human–Computer
Interaction Group

Jan Feuchter
jan.feuchter@hci.uni-hannover.de

Lectures

Session	Date	Topic	Details
1	2.4.	Introduction	human performance, empirical research, modeling
2	9.4.	Interaction elements	input devices, interaction elements, states, layouts
3	16.4.	Event handling	events, bindings, reactive programming, scene graph
4	23.4.	Scene graphs	event delivery, coordinate systems, nodes, animation, concurrency
5	30.4.	Interaction techniques	alignment and pointing techniques
6	7.5.	Interaction techniques	
7	14.5.	Web-based user interfaces	document object model, client-server issues
	21.5.	Pfingstwoche	
8	28.5.	Web-based user interfaces	reactive Programming for the Web
9	4.6.	Experiments and data analysis	designing experiments, hypothesis testing
10	11.6.	Modeling interaction	descriptive and predictive models, keystroke-level model, regression
11	18.6.	Visualization	visual encodings, perceptual accuracy, treemaps, dynamic queries
12	25.6.	Human-Centered AI	introduction to human-centered AI, human control and automation, examples
13	2.7.	Deep learning in HCI	guidelines for human-AI interaction, neural networks
14	9.7.	Deep learning in HCI	convolutional and recurrent NNs, face recognition, gesture recognition

ASSIGNMENT 03

Aufgabe 1: No Chrome

- Idea
  - Interface elements should get out of the way
  - Let users focus directly on the content itself
  - Works well for media consumption interfaces (immersion)
  - Works well for experts (focus on task, no visual distractions)

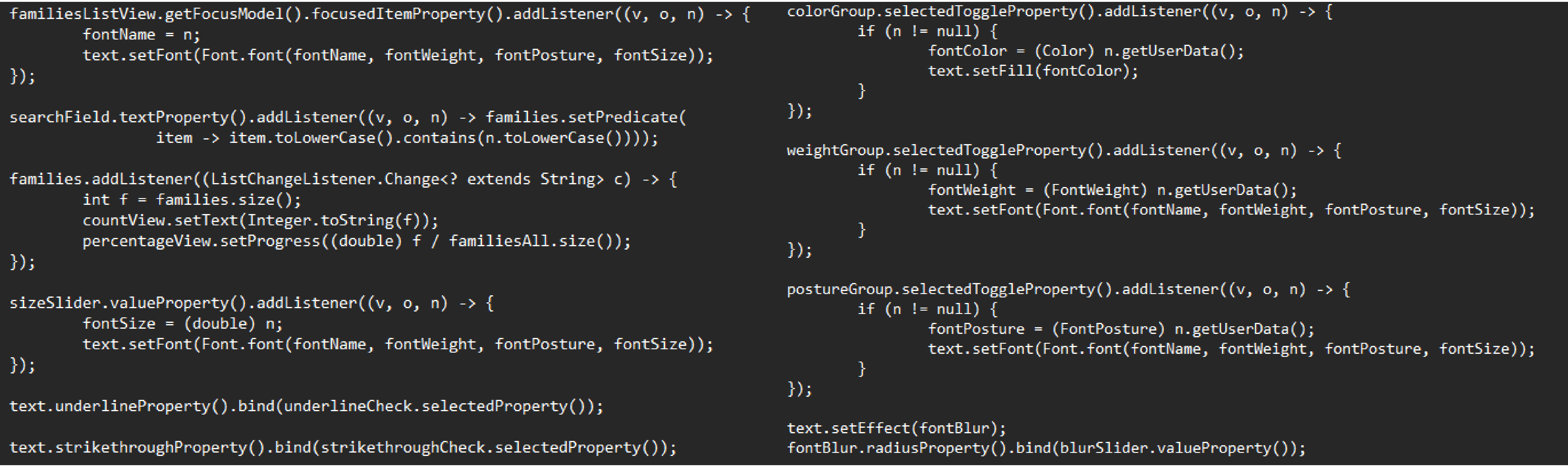
- Advantages
  - More display space
  - Less visual clutter

- Disadvantages
  - Problems of discoverability
  - High effort of learnability and/or memorizability
  - Multi-touch Interfaces on mobile devices

Aufgabe 2: Ladebalken

- Factors
  - How pulsation can be used to manipulate perceived duration
  - How animated ribbing affects perception of progress bar duration
  - Pitted three types of progress bars against each other (standard solid, best performing pulsating, best performing ribbed progress bar
  - allow participants to converge to a duration
- Can the results also be applied to round loading bars?

Aufgabe 3: Listeners, Properties, Bindings



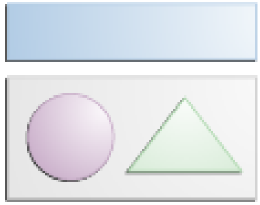
```
familiesListView.getFocusModel().focusedItemProperty().addListener((v, o, n) -> {
        fontName = n;
        text.setFont(Font.font(fontName, fontWeight, fontPosture, fontSize));
});

searchField.textProperty().addListener((v, o, n) -> families.setPredicate(
                item -> item.toLowerCase().contains(n.toLowerCase())));

families.addListener((ListChangeListener.Change<? extends String> c) -> {
        int f = families.size();
        countView.setText(Integer.toString(f));
        percentageView.setProgress((double) f / familiesAll.size());
});

sizeSlider.valueProperty().addListener((v, o, n) -> {
        fontSize = (double) n;
        text.setFont(Font.font(fontName, fontWeight, fontPosture, fontSize));
});

text.underlineProperty().bind(underlineCheck.selectedProperty());

text.strikethroughProperty().bind(strikethroughCheck.selectedProperty());
```

```
colorGroup.selectedToggleProperty().addListener((v, o, n) -> {
        if (n != null) {
                fontColor = (Color) n.getUserData();
                text.setFill(fontColor);
        }
});

weightGroup.selectedToggleProperty().addListener((v, o, n) -> {
        if (n != null) {
                fontWeight = (FontWeight) n.getUserData();
                text.setFont(Font.font(fontName, fontWeight, fontPosture, fontSize));
        }
});

postureGroup.selectedToggleProperty().addListener((v, o, n) -> {
        if (n != null) {
                fontPosture = (FontPosture) n.getUserData();
                text.setFont(Font.font(fontName, fontWeight, fontPosture, fontSize));
        }
});

text.setEffect(fontBlur);
fontBlur.radiusProperty().bind(blurSlider.valueProperty());
```

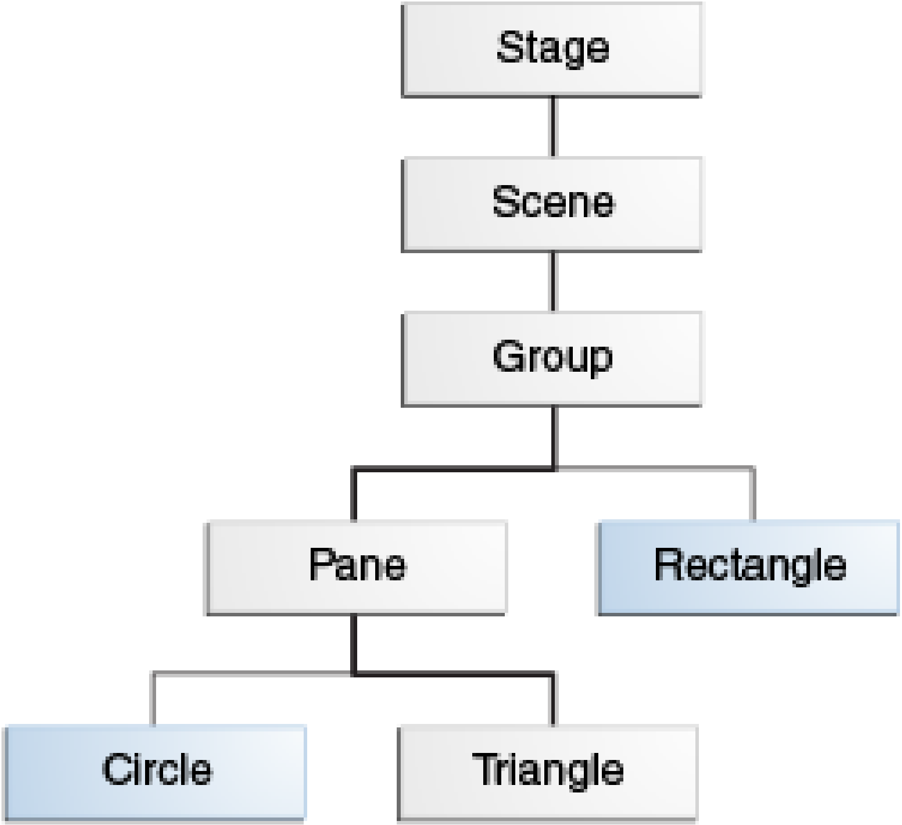
EVENTS IN SCENE GRAPHS

Example Scene and its Scene Graph

Scene



Scene Graph

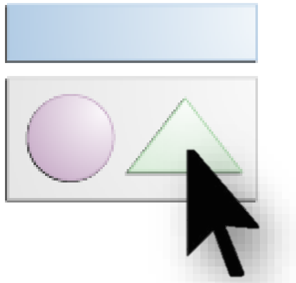


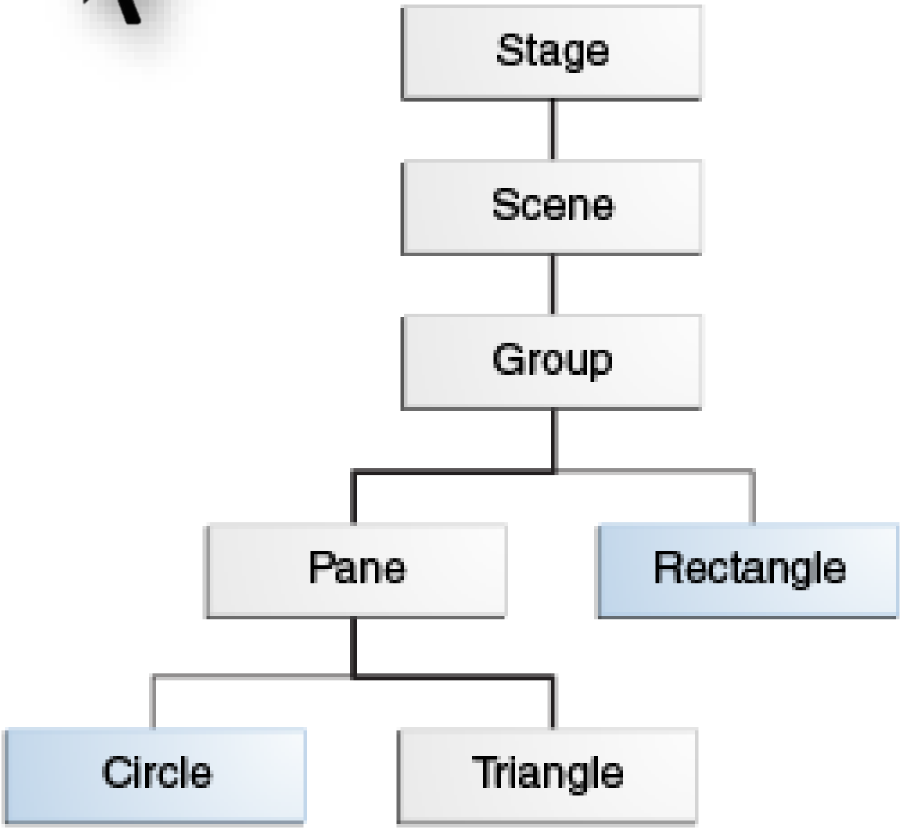
Event delivery: Events of different types are delivered to different nodes of the scene graph

http://docs.oracle.com/javase/8/javafx/events-tutorial/processing.htm

Event Filters and Event Handlers

- Wo sitzen event filters und handlers?
- Was filtern/handeln sie?
- Unterschied zwischen Event Filters und Handlers?

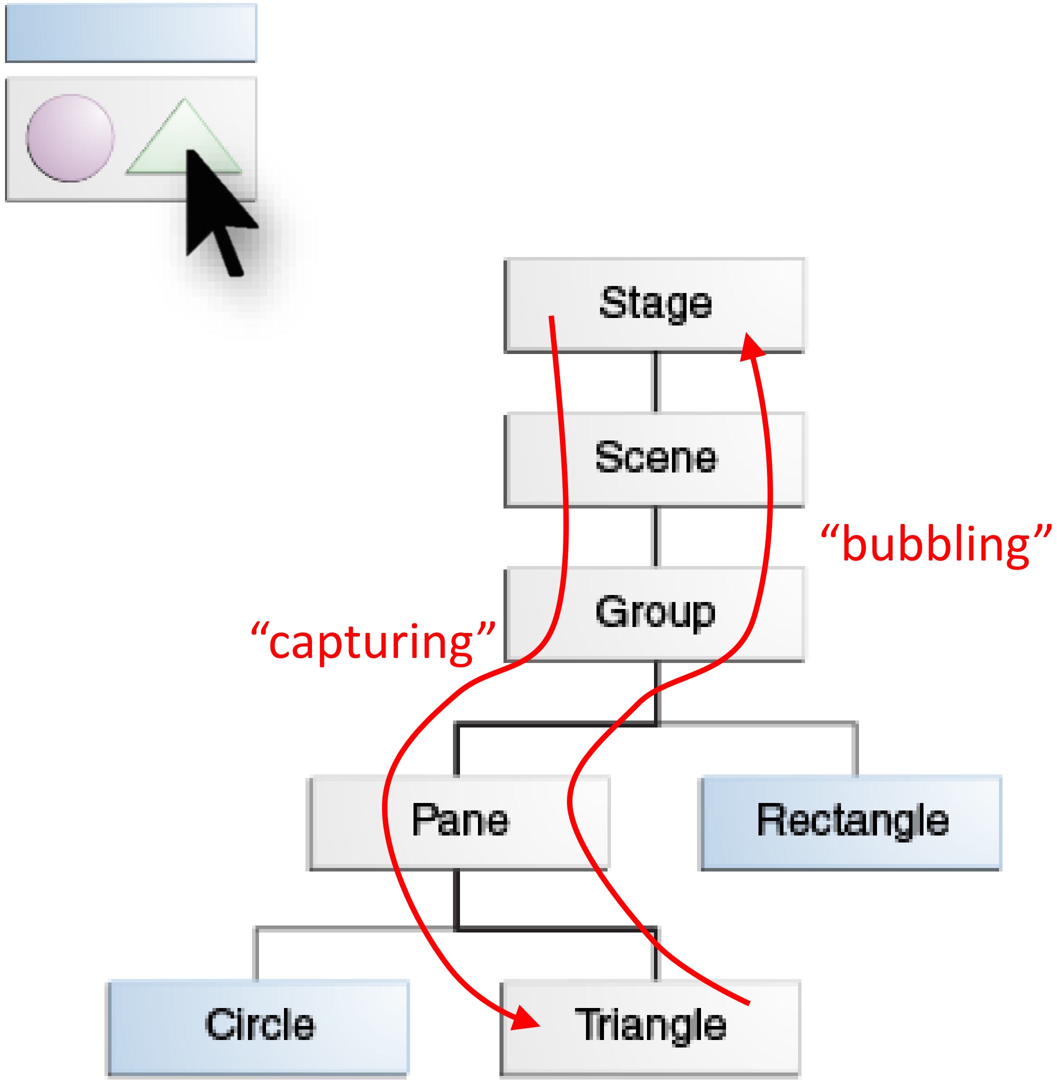




http://docs.oracle.com/javase/8/javafx/events-tutorial/processing.htm

Event Filters and Event Handlers

- Event filters: event capturing phase
  - Down the event dispatch chain
- Event handlers: event bubbling phase
  - Up the event dispatch chain

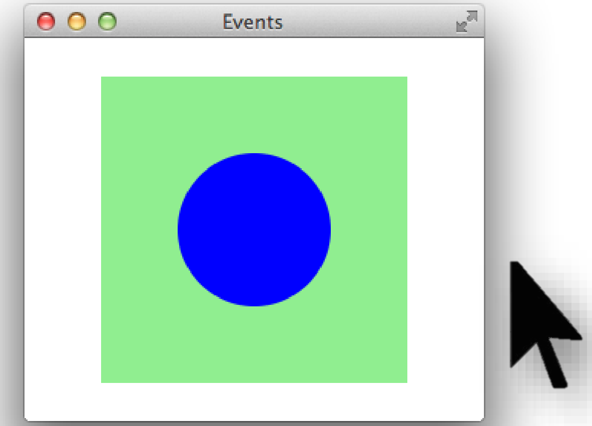


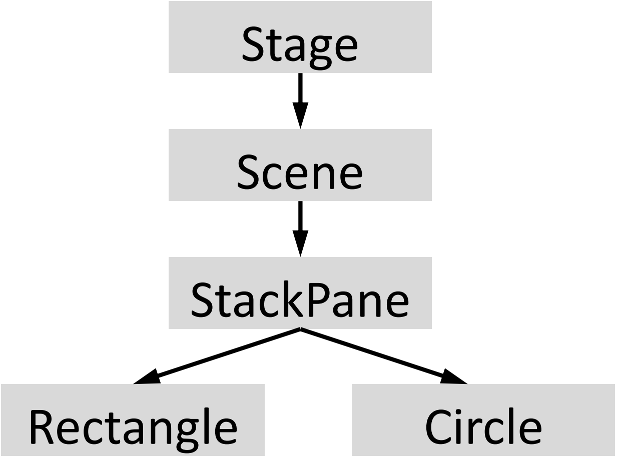
http://docs.oracle.com/javase/8/javafx/events-tutorial/processing.htm

Event Filters and Handlers

```
Circle circle = new Circle(50, Color.BLUE);

circle.addEventFilter(MouseEvent.MOUSE_PRESSED, e -> {
   System.out.println("\ncircle, event filter:");
   System.out.println(e);
   // e.consume();
});

circle.addEventHandler(MouseEvent.MOUSE_PRESSED, e -> {
   System.out.println("\ncircle, event handler:");
   System.out.println(e);
});
```

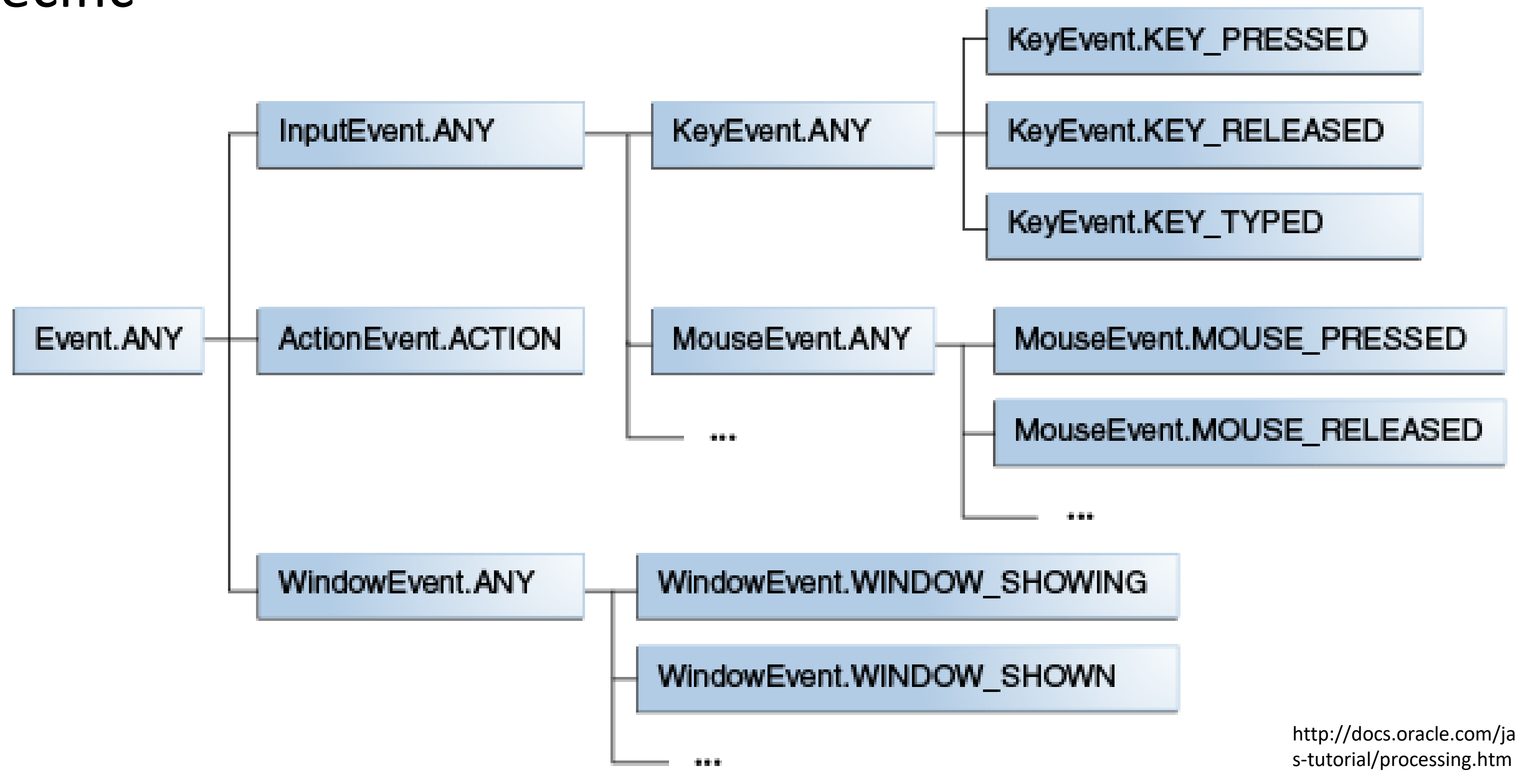
consume stops further event delivery





Event Types

- Event: Occurrence of something of interest
- Event types: Hierarchy from most general (Event.ANY) to most specific



http://docs.oracle.com/javase/8/javafx/events-tutorial/processing.htm

Event Delivery

- Target selection
  - Mouse events: Node at the cursor location
  - Key events: Node that has the focus (scene has the focus by default)
- Route construction
  - Node’s buildEventDispatchChain creates chain of dispatchers
    - Default implementation follows layout hierarchy
    - Target specifies event dispatch chain
  - Route may be modified (rarely necessary)
  - Events may be modified or consumed during delivery

Convenience Methods

- Convenience methods in class Node to simplify event handling
- Pattern:

```
setOn{event type}(EventHandler
     <? super {event class}> h)
```

- Example:

```
r.setOnMousePressed(
      (MouseEvent e) -> { ... });
```

lambda expression

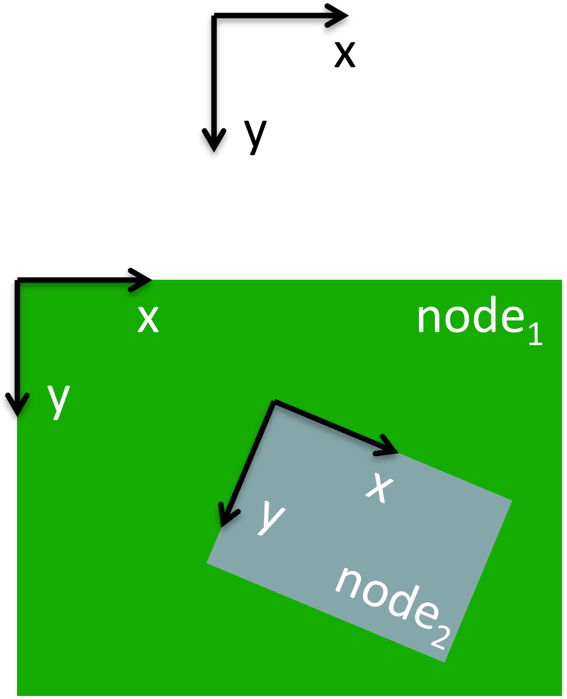
setOnX	Event Type
KeyPressed	KeyEvent
KeyReleased	KeyEvent
KeyTyped	KeyEvent
MousePressed	MouseEvent
MouseReleased	MouseEvent
MouseClicked	MouseEvent
MouseMoved	MouseEvent
MouseEntered	MouseEvent
MouseExited	MouseEvent
MouseDragEntered	MouseDragEvent
MouseDragExited	MouseDragEvent
MouseDragOver	MouseDragEvent
MouseDragReleased	MouseDragEvent
MouseDragged	MouseEvent
Action	ActionEvent
...	ScrollEvent

Short Recap

COORDINATE SYSTEMS IN SCENE GRAPHS

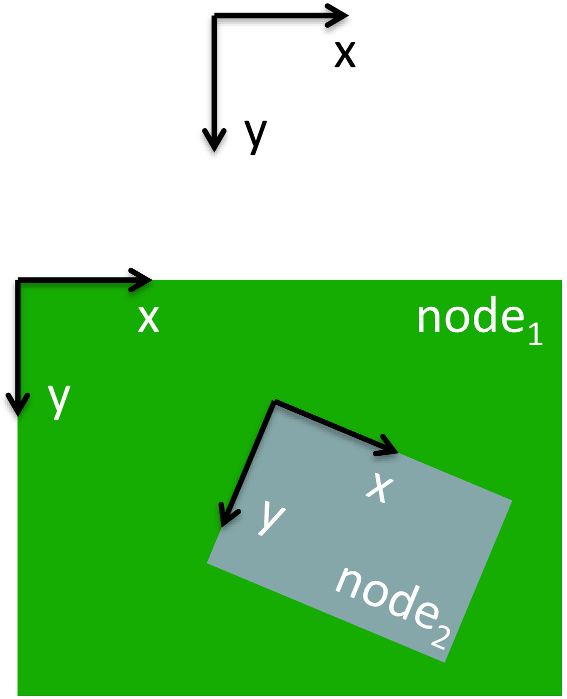
Node Coordinate Systems

- Each node has its own coordinate system
  - x axis to right, y axis downwards
  - May be translated, rotated, scaled, and sheared w.r.t. parent node's coordinate system
- Shape-nodes define their geometry within local coordinate system
  - Rectangle: x, y, width, height
  - Circle: centerX, centerY, radius



Node Coordinate Systems

- Each node has its own coordinate system
  - x axis to right, y axis downwards
  - May be translated, rotated, scaled, and sheared w.r.t. parent node's coordinate system



- Warum mehrere Koordinatensysteme?

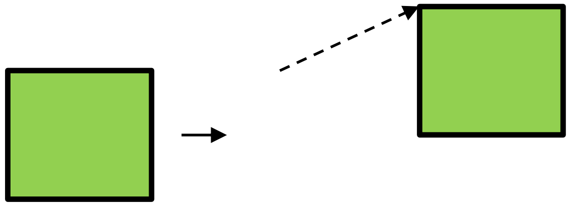
Affine Transformations

- Affine transformations are linear mappings of coordinates
  - Represented as matrices
  - Preserve straightness and parallelism of lines
- Affine transformations may be chained
  - Multiplication of matrices

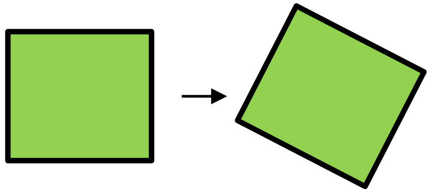
- Translation, rotation, scaling, shearing
- But not: Perspective mapping

Types of Affine Transformations

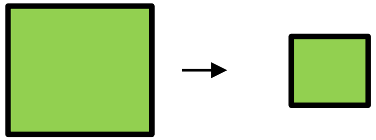
- Translation
  - Shifts the origin of the coordinate system along x or y axis



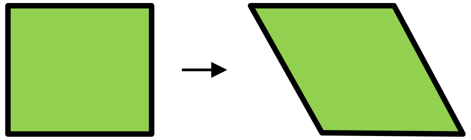
- Rotation
  - Rotates the coordinate system about a “pivot” point
  - Pivot point is center of layout bounds by default



- Scaling
  - Scales the axes of the coordinate system about a “pivot” point
  - Pivot point is center of layout bounds by default



- Shearing
  - Rotates the axes, such that they are no longer perpendicular



Conceptual Examples

COORDINATE SYSTEMS IN SCENE GRAPHS

Example: Bounding Rectangles of Node

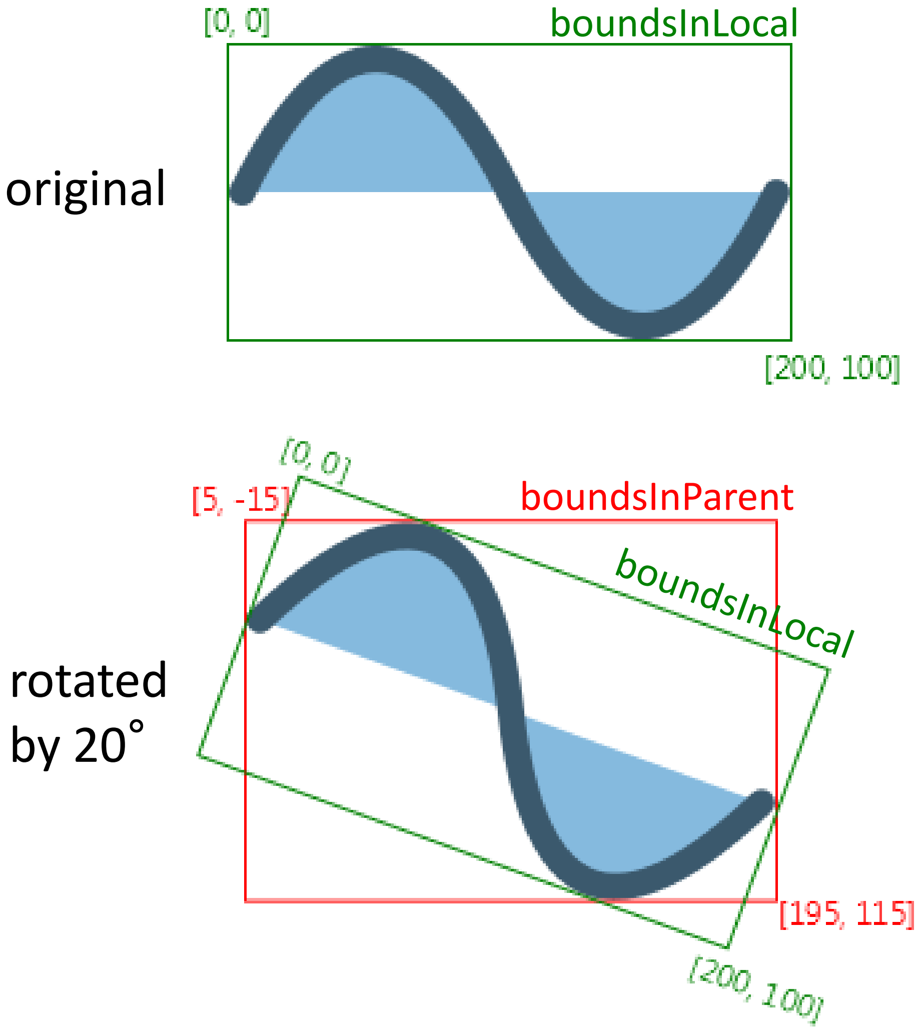
Smallest axis-parallel enclosing rectangle in the respective coordinate system

Green: boundsInLocal

Red: boundsInParent

Each node has:

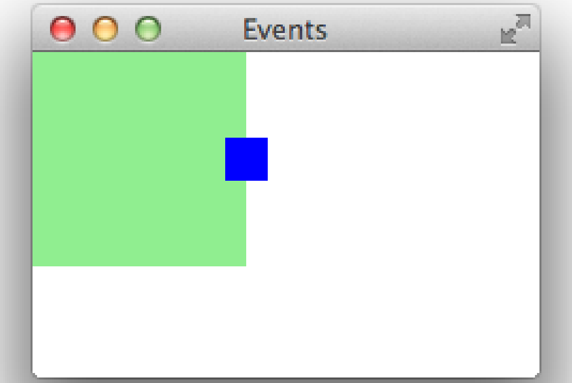
- List of transforms (applied before the following)
- translateX, translateY transforms
- scaleX, scaleY transforms
- rotate transforms



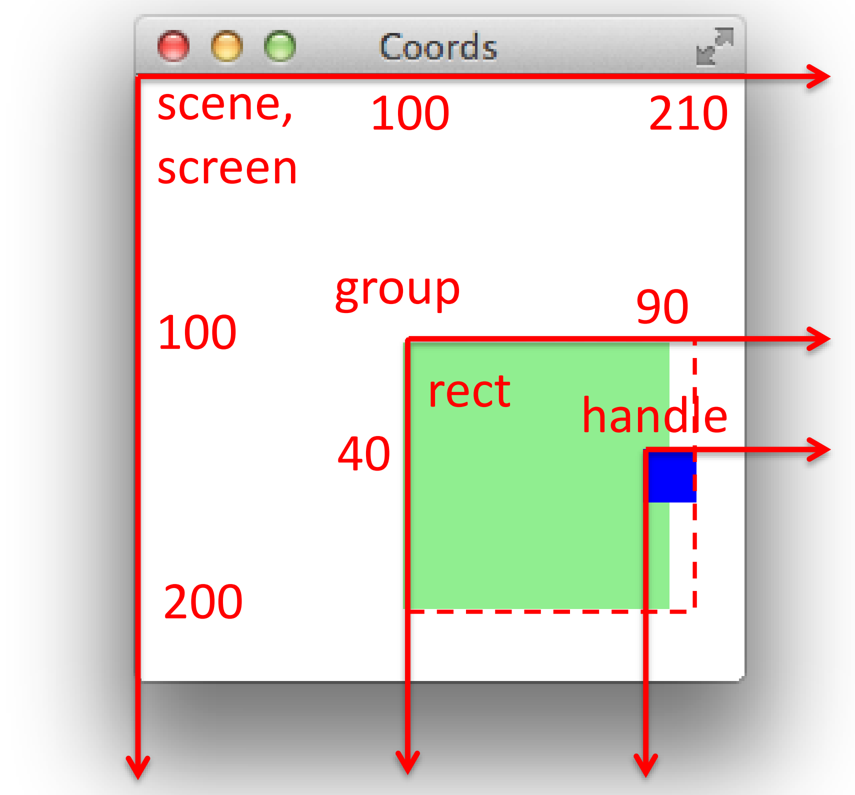
http://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html

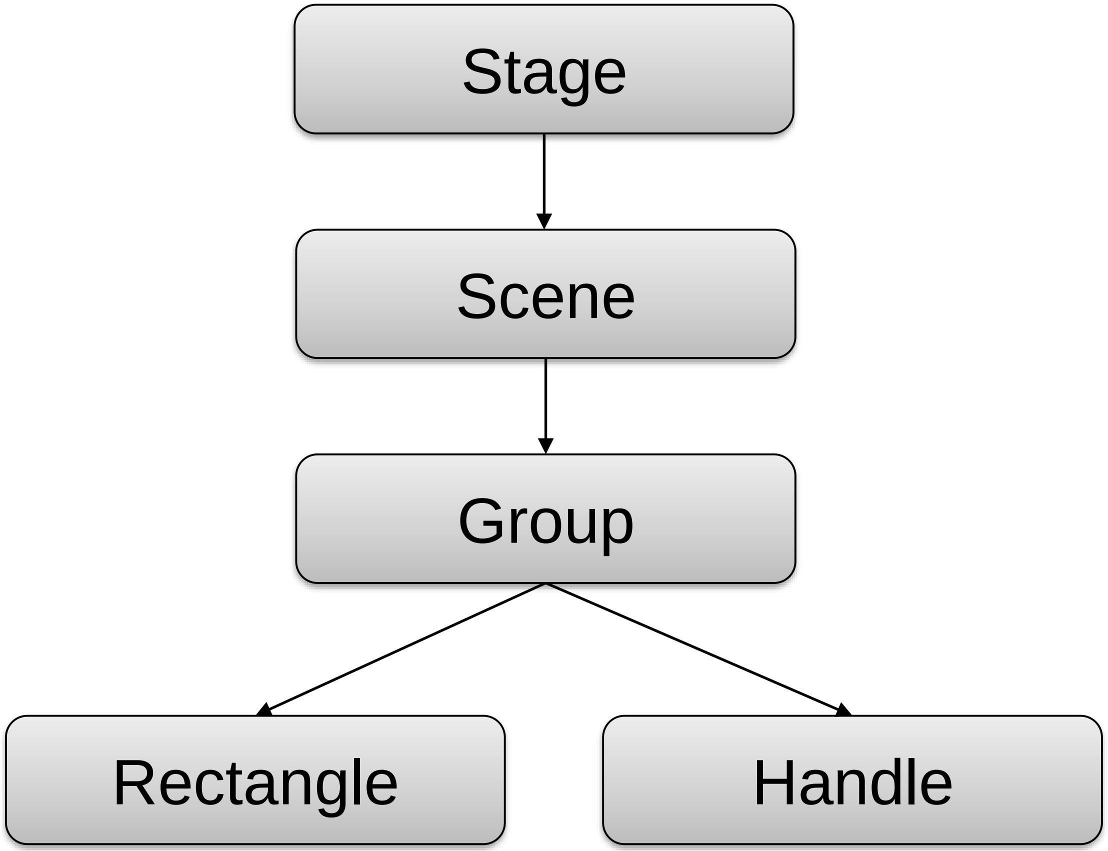
Example: A Moveable and Width-Resizable Rectangle

```
Rectangle rect = new Rectangle(100, 100, Color.LIGHTGREEN);
Rectangle handle = new Rectangle(20, 20, Color.BLUE);
handle.setTranslateX(90);
handle.setTranslateY(40);
Group group = new Group(rect, handle);

stage.setScene(new Scene(group));
stage.setTitle("Events");
stage.show();
```

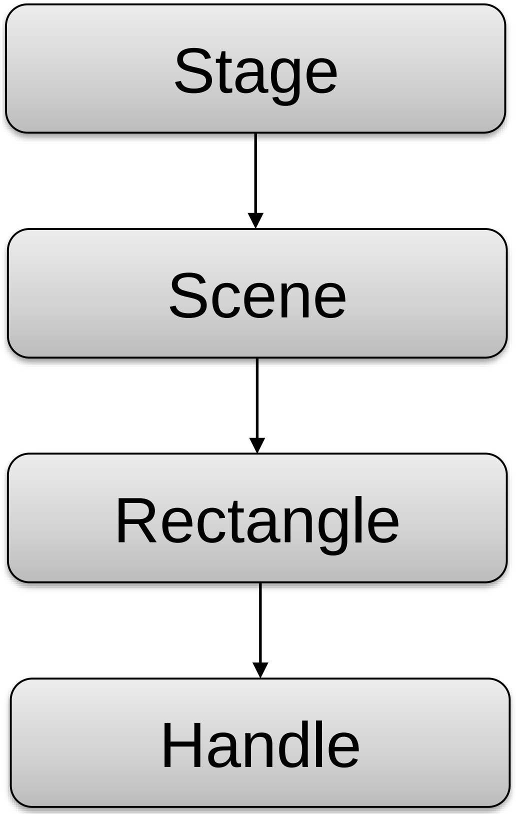


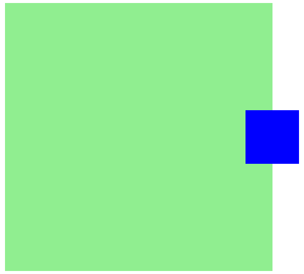
Example: A Moveable and Width-Resizable Rectangle





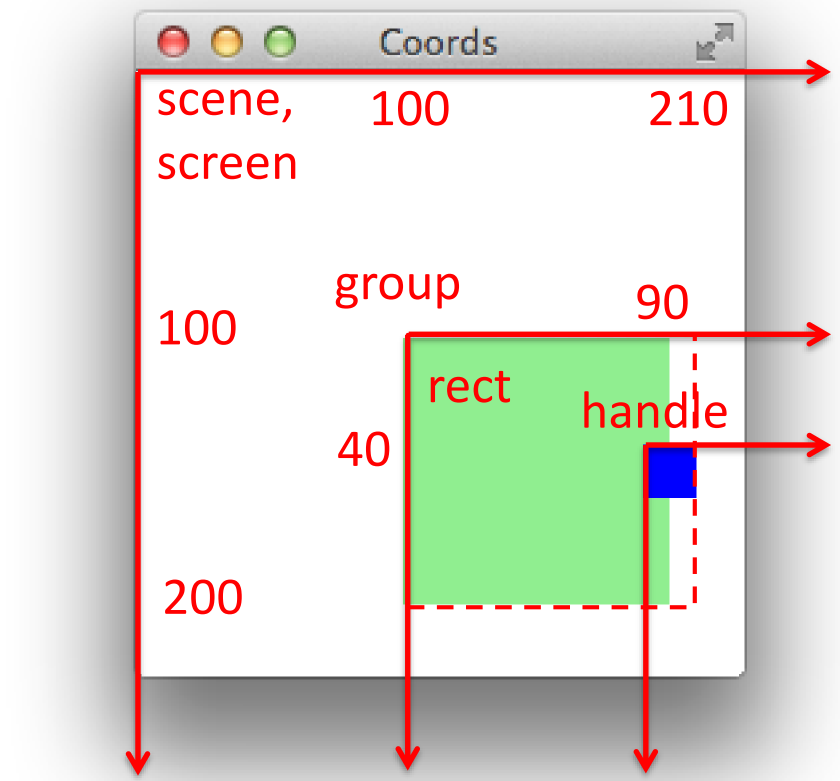
Alternative Scene Graph Structure





Ist dieses Layout auch möglich? Was macht es besser/schlechter?

Example: A Moveable and Width-Resizable Rectangle



BoundingBoxes:

group in local: ⬚
[minX:0, minY:0, maxX:110, maxY:100, width:110, height:100]

group in parent: ⬚
minX:100, minY:100, maxX:210, maxY:200, width:110, height:100]

rect in local: 🟩
[minX:0, minY:0, maxX:100, maxY:100, width:100, height:100]

rect in parent: 🟩
[minX:0, minY:0, maxX:100, maxY:100, width:100, height:100]

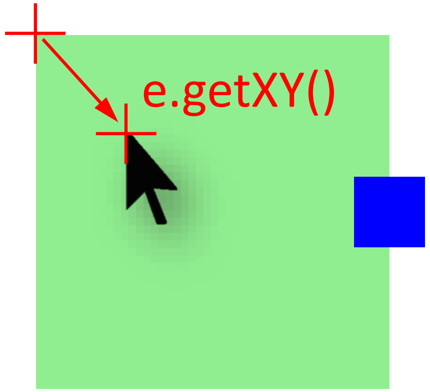
handle in local: 🟦
[minX:0, minY:0, maxX:20, maxY:20, width:20, height:20]

handle in parent: 🟦
[minX:90, minY:40, maxX:110, maxY:60, width:20, height:20]

Example: A Moveable and Width-Resizable Rectangle

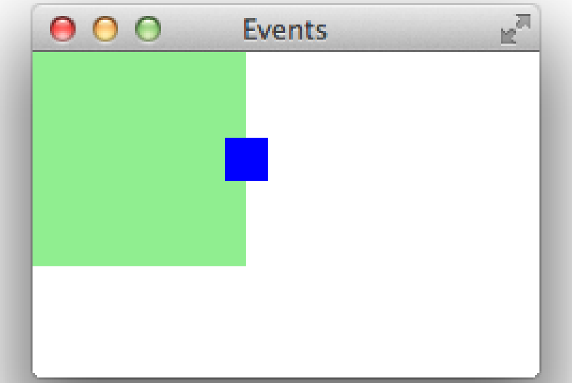
Drag group (green rectangle and handle):

```
group.addEventHandler(MouseEvent.MOUSE_PRESSED, e -> {
   offsetX = e.getX();
   offsetY = e.getY();
});

group.addEventHandler(MouseEvent.MOUSE_DRAGGED, e -> {
   group.setTranslateX(e.getSceneX() - offsetX);
   group.setTranslateY(e.getSceneY() - offsetY);
});
```

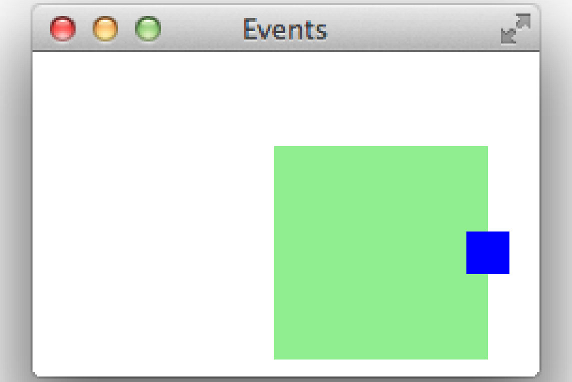


Example: A Moveable and Width-Resizable Rectangle

- Before dragging:
  - group: layoutX = 0, translateX = 0
  - rect: layoutX = 0, translateX = 0, x = 0
  - handle: layoutX = 0, translateX = 90, x = 0



- After dragging:
  - group: layoutX = 0, **translateX = 113**
  - rect: layoutX = 0, translateX = 0, x = 0
  - handle: layoutX = 0, translateX = 90, x = 0



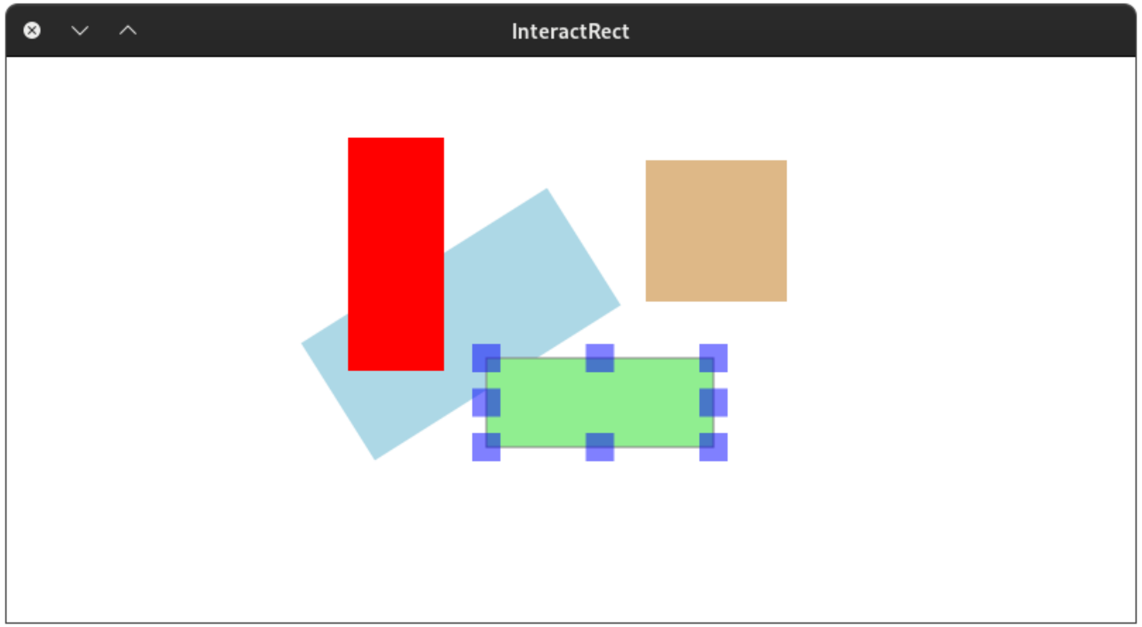
ASSIGNMENT 4

Assignment 4: Exercise 1

- Interactive Rectangles

- Uses hierarchical coordinate systems and transformations
  - Position and rotation of handles is always influenced by the parent rectangle's position and rotation

- Assignment will be available this afternoon


InteractRect