CISC 3115 TY3

C30a: Overview of JavaFX Application: Events

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Discussed
  - Concept of Stage, Scene, and Scene Graph
  - How to build views, representing visual elements of a Graphical User interface
- To discuss
  - Handling events
    - Event
    - Event dispatcher chain
    - Registering handlers and handling events
    - Writing event handlers efficiently: inner class, anonymous inner class, and Lambda expression

Event-Driven Programming

- The main body of the program is an event loop (in pseudo code)

```
do {
        e = getNextEvent()
        processEvent(e)
} while (e != EXIT_EVENT)
```

Expected to be completed very quickly (a fraction of a second)

- This event loop often implemented by the platform
- The events are in a queue called event queue (capacity? priority?)
- Users write event handler routines (user's programs) to process events
  - processEvent in the above will invoke your event handler routines
  - Event thus drives user's programs

Questions to Answer

- What are events?
- How are events processed?
- How do the events relate to the view (visual elements) of the User Interface?

Events

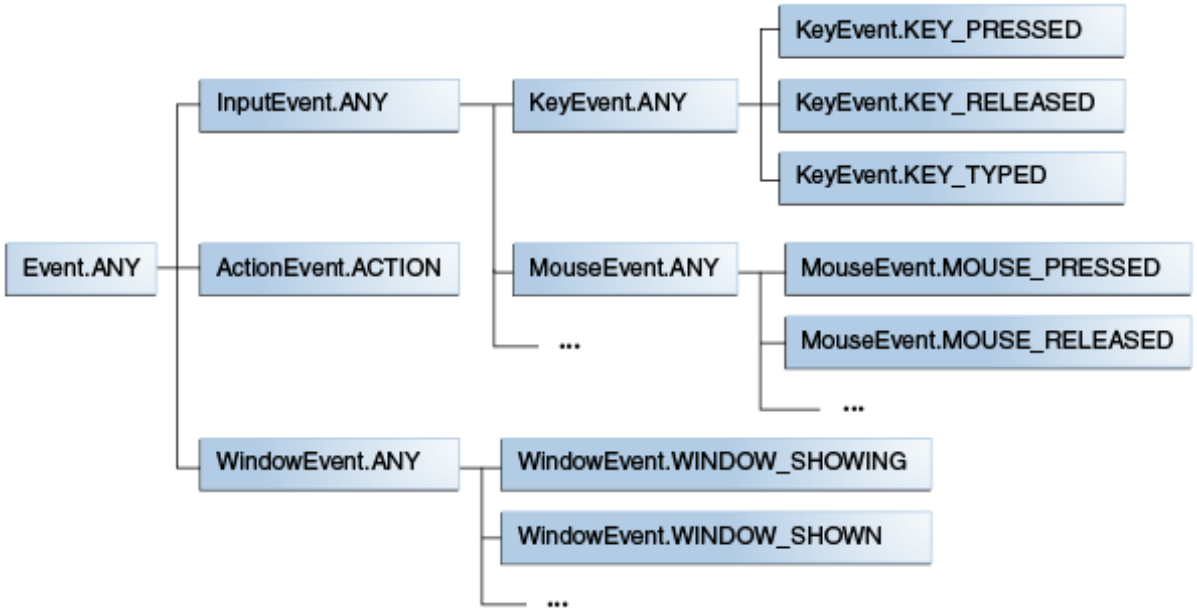
- Representing occurrence of something of the application's interest that drives the execution of the application
- Mouse events
  - Mouse pressed, mouse released, mouse clicked (pressed & released), mouse moved, mouse entered, mouse exited, mouse dragged
- Keyboard event
  - Key pressed, key released, key typed (pressed & released)
- Gesture event, touch event, …

JavaFX Events

- javafx.event.Event
  - An event is an object of the Event class or any subclass of the Event class
- There are many types of events

JavaFX Event Type

- javafx.event.EventType: specific event type associated with an Event
  - Event types forms a hierarchy

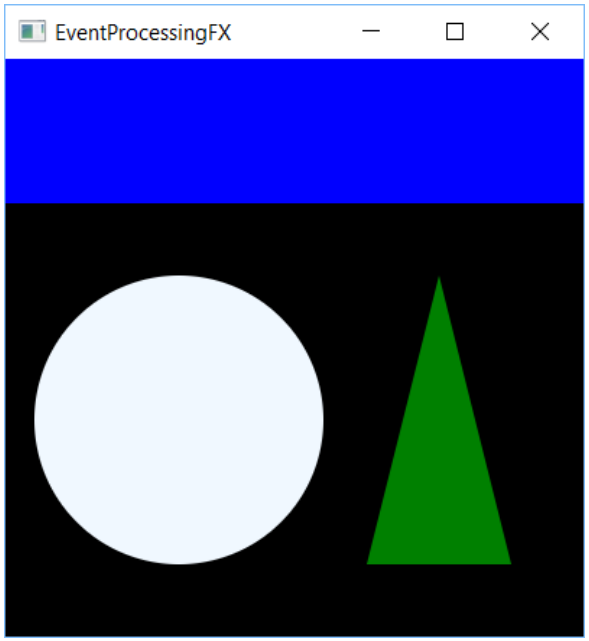


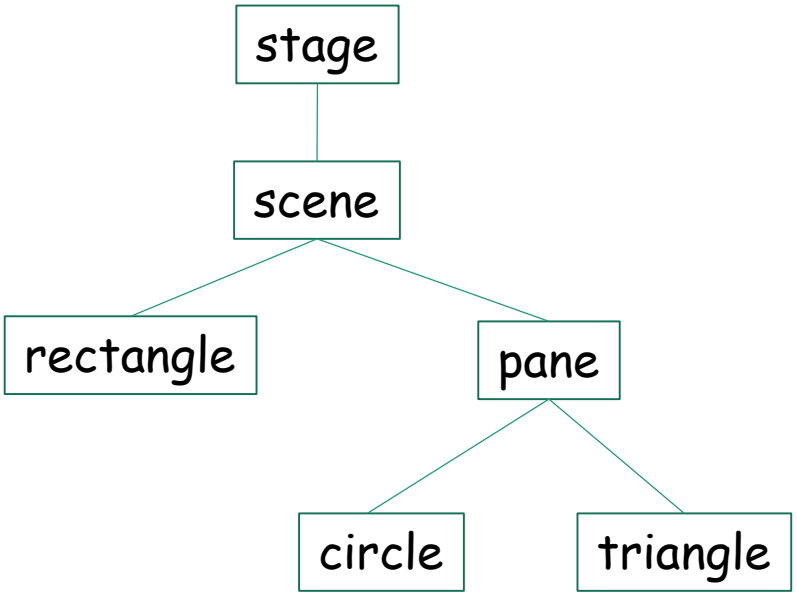
Event and Event Dispatcher Chain

- An event travels along a path called an event dispatcher chain
  - Typically, the path consists of objects of various Nodes, Stage, and Scene
  - Meaning: (the reference to) an event object is being passed from the Stage, the Scene, various nodes that constitute the path (or the event dispatcher chain)
- Each node in the chain has an interest of processing the event
- The chain is usually formed by following some parent - child hierarchy

Event Dispatcher Chain: Example

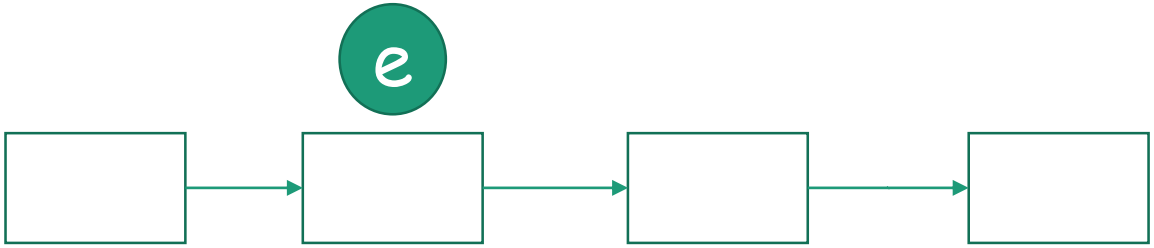
- Consider following UI and scene graph, what happens or what we wish to happen when we click on the circle?





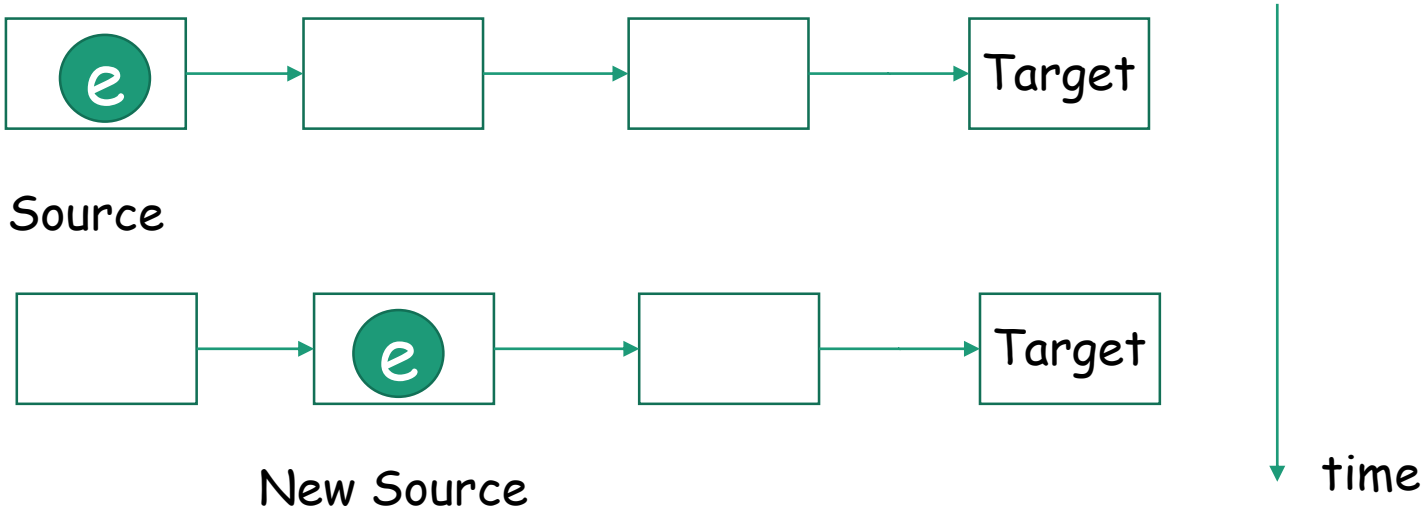
Event Dispatcher Chain

- A path of nodes, stage, and scene along which an event object is passed
- Event source
  - Where (such as, a stage, scene, or a node) an event is from. The source changes as the event is passed along the chain
- Event target
  - a node where the action occurred and the end node of the dispatcher chain. The target does not change.



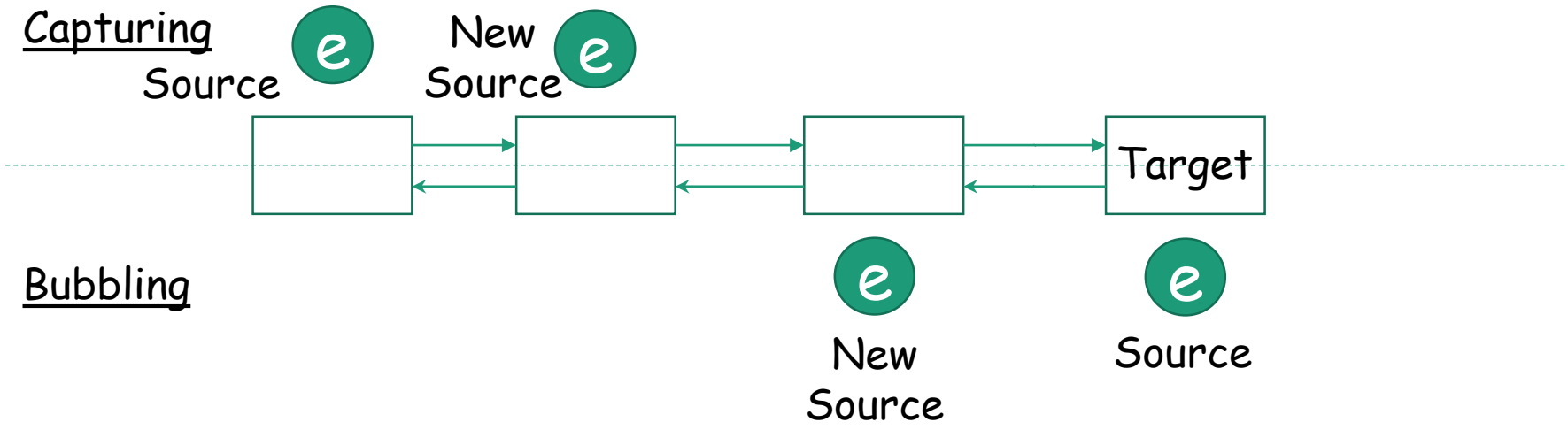
Passing Event

- Passing an Event object along the dispatcher chain
  - The source changes as it travels
  - The target remains the same



Event Capturing and Bubbling

- Undisturbed, an event is passed/travels in a two-way "round trip"
  - Capturing: source to target
  - Bubbling: target to source

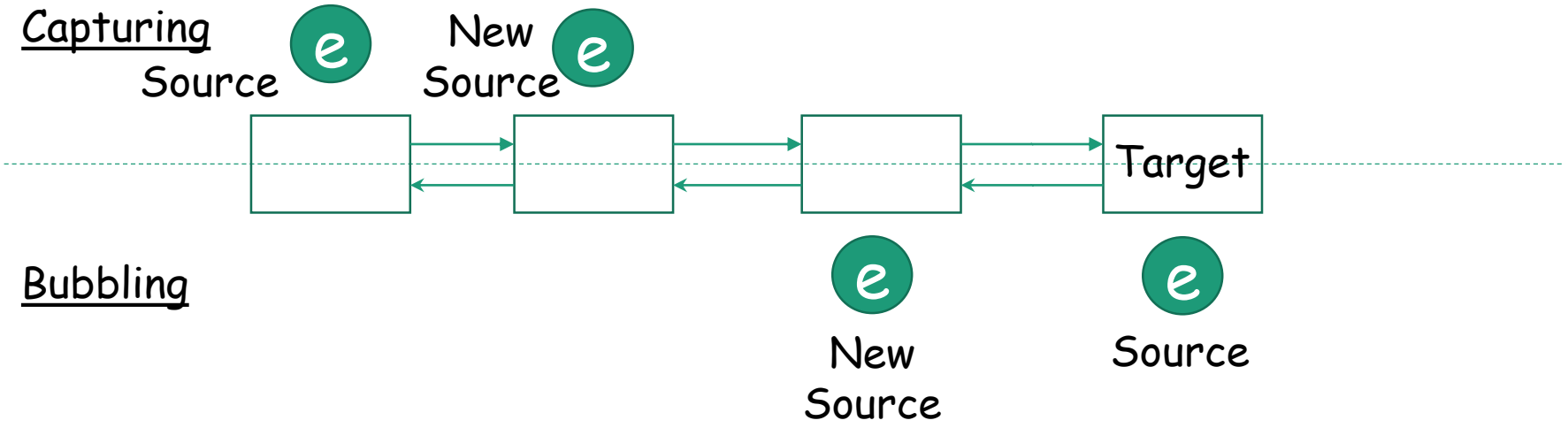


Event Handling: EventFilter and EventHandler

- Event handling via EventFilter and EventHandler
- Add one or more EventFilter at each node
  - Invoked during the capturing phase at the source
- Add one or more EventHandler at each node
  - Invoked during the bubbling phase at the source

Event Capturing and Bubbling

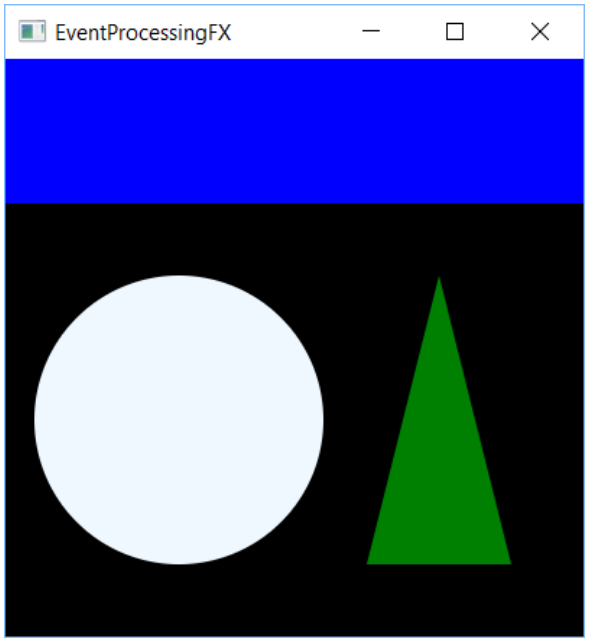
Capturing: Invoke registered (event filter) handler
Registering: Node::addEventFilter(…)

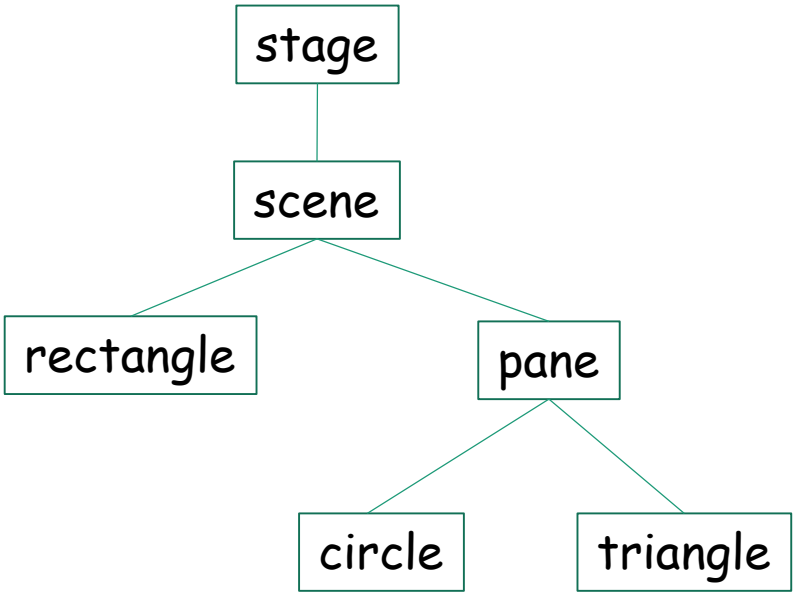


Bubbling: Invoke registered (event handler) handler
Registering: Node::addEventHandler() or convenience methods

Event Dispatcher Chain: Example

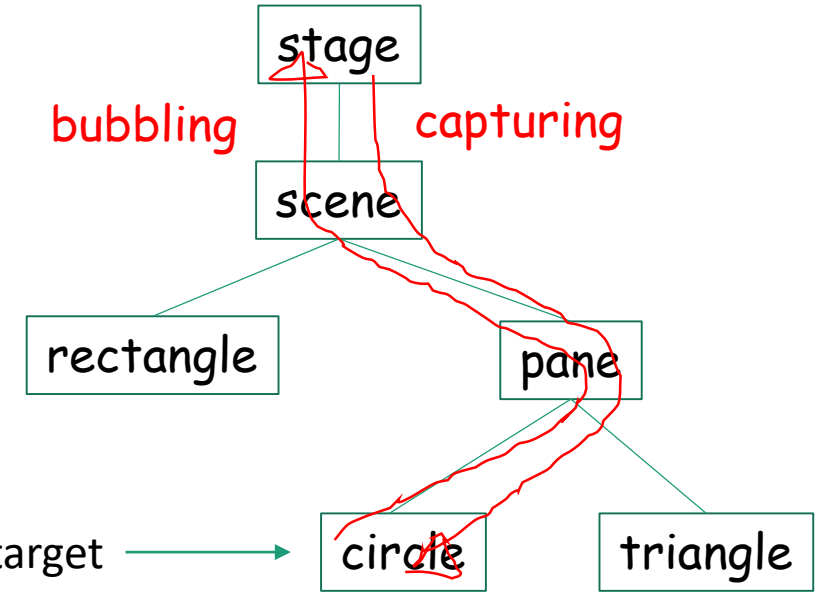
- Consider following UI and scene graph, what happens or what we wish to happen when we click on the circle?





Event Dispatcher Chain: Default Event Dispatcher Chain

- By default, the chain is usually formed by following some parent - child hierarchy
- Observe the demo



Event Delivery Process

- Target selection
- Route construction
- Event capturing
- Event bubbling

Target Selection

- When an action occurs, JavaFX determines which node is the target based on internal rules
- Examples:
  - Key events: the target is the node that has focus.
  - Mouse events: the node at the location of the cursor.
  - Gesture events: the node at the center point of all touches at the beginning of the gesture; or the node at the location of the cursor.
    - Swipe events: the node at the center of the entire path of all of the fingers; or or the node at the location of the cursor.
  - Touch events: the node at the location first pressed.
  - If more than one node is located at the cursor or touch, the topmost node is considered the target.

Route Construction

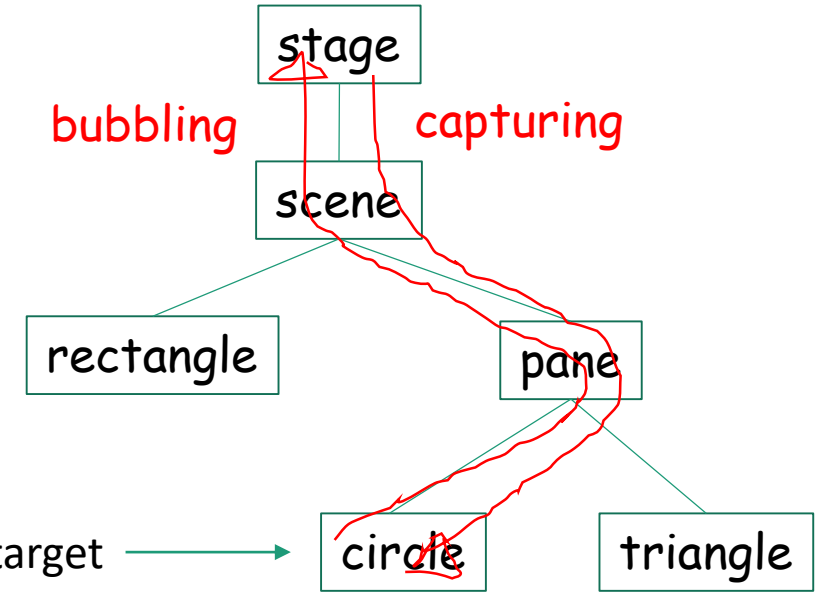
- Selected event target determine the initial dispatch chain
  - It implements the buildEventDisptachChain(…) method in the EventTarget interface
  - The implementation of the method determines the initial chain
- By default, the chain is usually formed by following some parent - child hierarchy

Consume Events

- Events can be consumed
  - Stop passing the event to next node along the event dispatcher chain in either direction
    - event capturing
    - event bubbling
  - In an event filter
    - Stops capturing
  - In an event handler
    - Stops bubbling

Event Dispatcher Chain: Default Event Dispatcher Chain

- Events can be consumed at any source during either the capturing or the bubbling phase
- Observe the demo



Question?

- JavaFX event handling
- Event dispatcher chain
  - Event source, event target, event capturing phase, event bubbling phase
- Default/initial event dispatcher chain constructed by Nodes
- Event handling
  - event handlers and event filters
- Events can be consumed

Basic Event Handling

- Register event handlers at nodes
- Response to
  - mouse events, keyboard events, action events, drag-and-drop events, window events, and others.
- At the capturing or the bubbling phase depending on the event is being registered

Event Registration

- More than one handlers can be registered at any node during either the bubbling and capture phase
- Use Node::addEventFilter(…) for processing/handling event at the capturing phase
- Use Node::addEventHandler(…) for processing/handling event at the bubbling phase
- Use Node's convenience methods for processing/handling event at the bubbling phase

Convenience Method

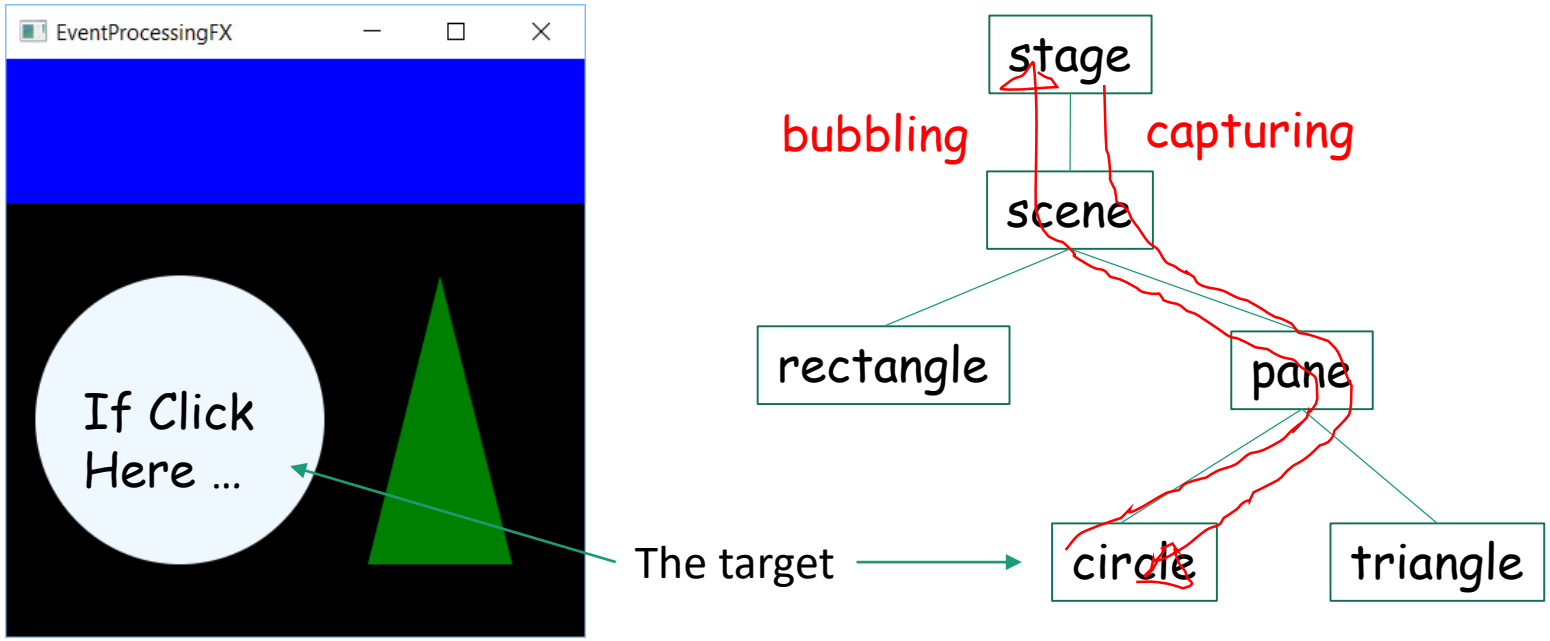
- Commonly via the convenience methods provided by the target nodes
  - setOnMouseClicked, setOnMouseEntered, seOnMouseExited …
  - Naming convention: setOnEvent-type

Event Handler List

- A node can have more one handler at either the capturing or the bubbling phase
- They are invoked according to the insertion order
- Event Handler registered via the convenience method is always placed at the tail of the list
  - Meaning: it is to be invoked last

Basic Event Handling: Example

- Run it again with revision



Questions

- Basic event handling
- Event handler list
- How to register event handler at the capturing phase?
- How to register event handler at the bubbling phase?
- What if there is more than one handler at either one of the two phases?