Lecture 6: Input Models

Today's Topics

- Input

Why Use Events for GUI Input?

- Console I/O uses blocking procedure calls

```
print ("Enter name:")
name = readLine();
print ("Enter phone number:")
name = readLine();
```

  - System controls the dialogue
- GUI input uses event handling instead
  - User has much more control over the dialogue
  - User can click on almost anything

Kinds of Input Events

- Raw input events
  - Mouse moved
  - Mouse button pressed or released
  - Key pressed or released
- Translated input events
  - Mouse click or double-click
  - Mouse entered or exited component
  - Keyboard focus gained or lost (loss of focus is sometimes called "blur")
  - Character typed

Properties of an Input Event

- Mouse position (X,Y)
- Mouse button state
- Modifier key state (Ctrl, Shift, Alt, Meta)
- Timestamp
  - Why is timestamp important?

Event Queue

- Events are stored in a queue
  - User input tends to be bursty
  - Queue saves application from hard real time constraints (i.e., having to finish handling each event before next one might occur)
- Mouse moves are coalesced into a single event in queue
  - If application can't keep up, then sketched lines have very few points

Event Loop

- While application is running
  - Block until an event is ready
  - Get event from queue
  - (sometimes) Translate raw event into higher-level events
    - Generates double-clicks, characters, focus, enter/exit, etc.
    - Translated events are put into the queue
  - Dispatch event to target component
- Who provides the event loop?
  - High-level GUI toolkits do it internally (Java, VB, C#)
  - Low-level toolkits require application to do it (MS Win, Palm, SWT)

Event Dispatch & Propagation

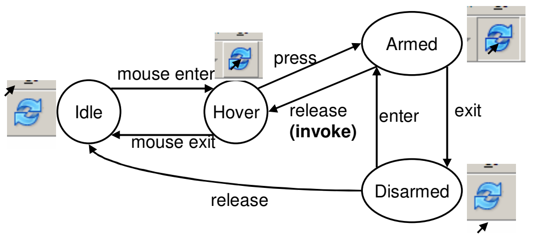
- Dispatch: choose target component for event
  - Key event: component with keyboard focus
  - Mouse event: component under mouse
    - **Mouse capture**: any component can grab mouse temporarily so that it receives all mouse events (e.g. for drag & drop)
- Propagation: if target component declines to handle event, the event passes up to its parent

Javascript Event Models

- Events propagate in different directions on different browsers
  - Netscape 4: downwards from root to target
  - Internet Explorer: upwards from target to root
  - W3C standardized by combining them
  - Netscape 6+/Mozilla/Opera/W3C: first downwards (“capturing”), then upwards (“bubbling”)

Designing a Controller

- A controller is a finite state machine
- Example: push button



Interactors

- Generic reusable controllers (Garnet and Amulet toolkits)
  - Selection interactor
  - Move/Grow interactor
  - New-point interactor
  - Text editing interactor
  - Rotating interactor
- Hide the details of handling input events and finite state machines
- Useful only in a component model
- Parameterized
  - start, stop, abort events
  - start location, inside/outside predicates
  - feedback components
  - callback procedures on event transitions