Event-driven Programming: GUIs

Elements of Graphics
CS324e
Fall 2020
Event-driven Programming

- Programming model where code runs based on *events*
- Events occur asynchronously throughout program execution
  - System-generated events
  - User-generated events
- Some part of system signaled/messaged when event is triggered
- Change program flow based on user input, sensor output, or system messages
System-generated Events

- System initiates an event outside of user’s control
- Generated by:
  - External hardware beyond application (e.g. a system timer)
  - Internal software within application (e.g. notification of task completion)
- Application responds to event
User-generated Events

- System initiates an event based on user input onto connected hardware
  - Keyboard press
  - Mouse movement / click
  - Joystick control
- Operating system stores user input as event in a queue
- UI toolkits provide checks and responses to events
- Programmer determines behavior based on events
GUI and Menus

- Graphical user interfaces (GUIs) determine input based on mouse (or stylus) position on the screen
- Standard events already built into system
  - Window minimize, window close, etc
- Custom events added by programmer
  - Game paused, change music volume, etc
- User interacts with elements at any point of the program execution
Callbacks

❖ Tells the system what to do when particular event arrives
❖ Necessary code executes automatically
❖ Standard technique for a GUI system:
  1. Application implements function to handle event
  2. Application notifies GUI which function to call
  3. GUI handles this functionality when user interacts with the system
❖ What are some callbacks built into Processing?
Graphical User Interface

- Computer interface with a visual component
- Direct interaction with the screen rather than interactions via command line
- Designed for easier, more intuitive experience
- Based on event-driven programming
Uses

- Text editors
- Web browsers
- Music controls
- Video games
- Many, many more…

(iMovie)
Consider

- How are some ways we can interact with a GUI?
Widgets

- Interactable objects within a GUI:
  - Buttons
  - Check boxes
  - Radio buttons
  - Sliders
- Provide different ways of interacting with program behavior
Example Widget

(http://compsci.ca)
Buttons

- Allow for functionality upon mouse click
- Must be aware of mouse position and button boundary
- Circles and rectangles have accessible formulae to determine boundaries
  - Circles check based on radius from center position
  - Rectangles check based on width/height distance from corner (or center) position
- What do these equations look like?
Thinking More about Events

- Consider `mousePressed()`
  - Listens for events while `draw()` runs
  - Called every time a mouse button is pressed
- What is `mousePressed(MouseEvent event)`?
  - Event *explicitly* accessible to `mousePressed` callback
  - Tracks x and y position and mouse button pressed
  - Information accessible outside of the event (a Processing-specific feature)
Event Class

- Contains action, modifiers and length of action
- Parent class of:
  - KeyEvent
    - Key and key code information included
  - MouseEvent
    - Position, button and click count included
  - TouchEvent
Custom Events

- Possible to implement custom events
- Based on observer pattern
  - Object maintains list of dependent objects (observers)
  - Object notifies its observers of state changes
  - Appropriate observer method called in response
- Same concept used in GUI programming and for communicating in MVC (Model-View-Controller) pattern
Observers and Observables

- `java.util.Observer` creates Observer interface
- Notified of changes in associated observable objects
- Receive notification via `update` method
- `java.util.Observable` is data that is observed
- Notifies one or more observers of change to some value
- Sends notification via `notifyObservers` method
Instapoll Question: Callbacks

- What does a callback function contain?
  - Information the operating system needs to process an event
  - Custom code within an application for handling an event
  - Information about the event passed from the driver
Hands-on: Buttons

Today’s activities:

1. Implement a Button class that checks when the mouse is over it, and when the mouse clicks on it
2. Create both rectangular and circular buttons
3. Experiment with the mousePressed and mouseReleased event calls
4. Add functionality so that the sketch’s background color changes every time a button is pressed