CS 349
Interactive Components

Byron Weber Becker
Spring 2009

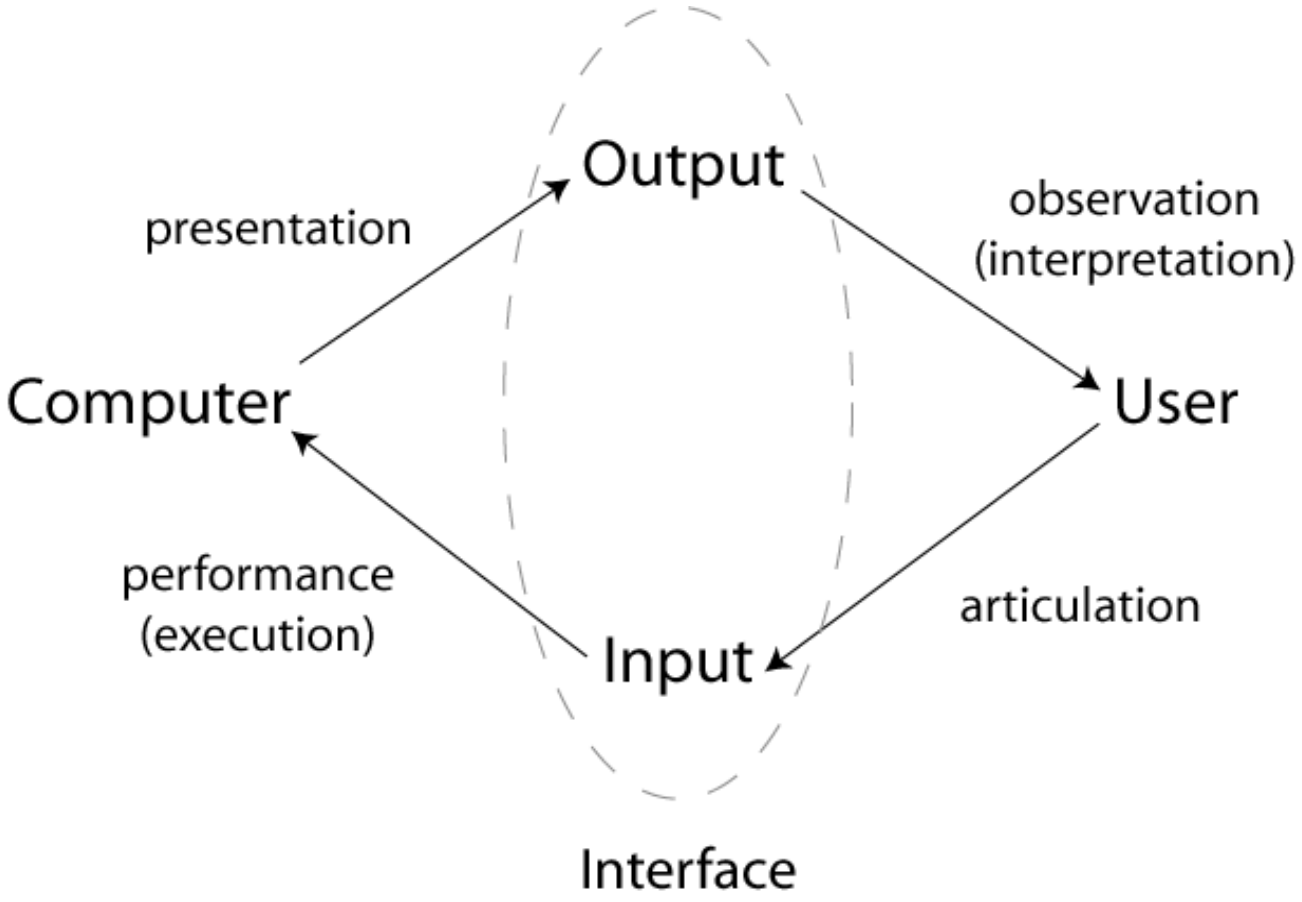
Slides mostly by Michael Terry

University of Waterloo

Interactive Components

- The method by which we communicate with the system in a graphical user interface
- Guidelines for constructing interactive components from perspective of
  - User
  - Developer/designer

Interactive Components



From Dix, Finlay, Abowd, & Beale (2004)

University of Waterloo

User's Perspective

- Interactive dialogue
  - Single thread of conversation with system
  - Explicit turn-taking
  - Set of conventions regulating turn-taking
  - Example: What are conventions for using a button?

Interactive Dialogue: Button

- **Demo...**
- Mouse over button…
- Press button…
- Move mouse while button pressed…
- Release mouse…

20-May-09 Announcements

- Thursday's office hours will be held today at 2:30.
- Newsgroup is now being monitored.
- A1 is due Sunday night at 11:59pm
- Demos on Monday?

University of Waterloo

Interactive Dialogue

- Typical conventions:

Computer	User
Prompt	
	Begin input
Echo	
	Finish input/Trigger
Accept Trigger	
Acknowledgement	
Response	

- These stages represent multiple cycles around basic interaction cycle

University of Waterloo

Prompt

- A synchronization cue
- Inform user that the system is ready for input
  - User is in control
- Common prompts:
  - Command prompt
  - Blinking caret
  - Enabled buttons
- Inform what is possible, not possible
- Conventions for indicating system is not ready, what is not possible
  - Grey out items not enabled, show progress dialog when busy

University of Waterloo

Echo

- Provide immediate feedback when interacting with control
  - Direct manipulation: Rapid, incremental feedback

Trigger

- Inform system your input is complete
  - Hit return on command line
  - Press “OK” button in a dialog box
  - Double-click
  - Release mouse button in a drag and drop operation

Acknowledgement

- Indicate the command is executing
- Provide ability to abort processing

University of Waterloo

Response

- Show how state of system has changed

Shared Perspectives

- GUI toolkit should be:
  - Complete
    - Designer has everything they need
  - Consistent
    - Behavior is consistent across components
  - Customizable
    - Developer can reasonably extend functionality to meet particular needs of application
- Goals can be (partially) achieved via using a common GUI toolkit.
- Meeting these requirements encourages *reuse.*

University of Waterloo

Completeness

- Component types
  - Canvases
    - For drawing
  - Selectors
    - For choosing among items (list, menu, checkbox, button…)
  - Changers
    - For modifying a value (sliders)
  - Containers
    - For containing/grouping other components
  - Text input
    - For text entry

University of Waterloo

Completeness

- The “Macintosh 7” (Dix, Finlay, Abowd, Beale, 1998)
  - Button
  - Slider
  - Pulldown menu
  - Check box
  - Radio button
  - Text entry / edit fields
  - File open / save
- To see the rich spectrum of controls available in Java Swing, check out SwingSet2 in demo/jfc folder of jdk installation (/usr/jdk…/demo/jfc/SwingSet2 in VM).

Consistency

- Facilitate learning by:
  - Employing common visual and/or auditory presentation
  - Sharing look and feel
  - What is meant by “feel”?

Consistency

- Look and feel
  - Look
    - Visual (or audio) appearance
  - Feel
    - Behavior of component, its interaction
    - How it is used
    - Example: If can activate a button using keyboard input, expect to be able to manipulate other controls using keyboard input

Customizable

- Component should provide suitable hooks for extension and customization for particular applications
- Common strategies
  - Flags/variables that can be directly set
    - Examples:
      - Component color
      - Component text
      - Component size
  - Factor out behavior that can change
    - Examples:
      - Responding to an action: ActionListener
      - Swing's UIManager look and feel
      - JTable as a rich example…

University of Waterloo

Customizable: JTable

- JTable factors out much of its functionality
  - The actual data (TableModel) (part of MVC pattern)
  - Selection of items (ListSelectionModel)
  - Rendering of cells (TableCellRenderer)
  - Editing of cells (TableCellEditor)
- Developer has lots of flexibility in the data that can be represented, how it can be selected, its method of presentation, and its method of editing

Demo

- Table demo
- Source code in Resources section of website: JTableDemo.java