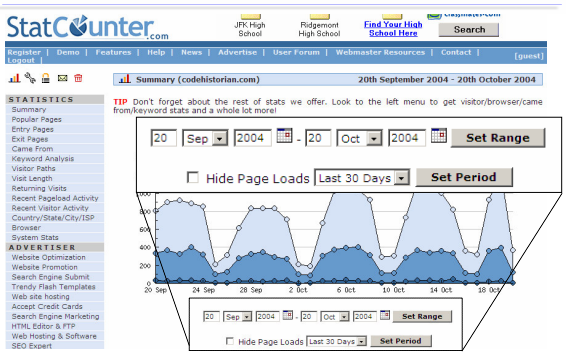
Lecture 19: Undo

UI Hall of Fame or Shame?



Today's Topics

- Undo design principles
- History visualization
- Selective undo
- Command objects

Forming a Mental Model of Undo

- Undo = reverses the effect of an action
- Questions
  - What stream of actions will be undone?
  - How is the stream divided into units?
  - Which actions are undoable?
  - How much of the previous state is actually recovered?
  - How far back can you undo?

What stream of actions will be undone?

- Actions in this window? (MS Office)
- Actions in this text widget? (web browser)
- Just my actions, or everybody's? (multiuser apps)
- What about actions made by the computer?
  - MS Office AutoCorrect and AutoFormat are undoable, even though user didn't do them

How is the stream divided into units?

- Low level
  - Mouse clicks, key presses, mouse moves
  - Nobody does it at this level
- Syntactic level
  - Commands and button presses
- Semantic level
  - Changes to application data structures (e.g., the result of an entire Format dialog)
  - This is the normal level
- Text entry is aggregated into a single action
  - But other editing commands (like Backspace) and newlines interrupt the aggregation
- What about user-defined macros?
  - Undo macro actions individually, or as a unit?

Which actions are undoable?

- User's action stream may include many actions ignored by Undo
  - Selection (of text or objects)
  - Keyboard focus (to different widgets or windows)
  - Changing viewpoint (scrolling, zooming)
  - Changing layout (opening palettes or sidebars, adjusting window sizes)
  - UI customization (adding buttons to toolbars)
- So which actions does Undo actually undo?
  - Some applications (e.g. web browsers, IDEs) Undo/Redo for the editing stream, Back/Forward for the viewpoint stream

How much state is recovered?

- Select text, delete it, and then undo
  - Text is restored
  - But is selection restored? Cursor position?

How far back can you undo?

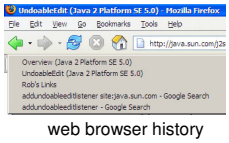
- Often a limit on history size
  - Used to be one action -- now usually hundreds, or infinite
- Does action stream persist across application sessions?
  - If so, stream must be saved to file
- Does it persist across File/Save?
  - Not in MS Office

Design Principles for Undo

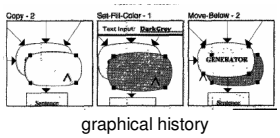
- Visibility
  - Make sure undone effects are visible
    - e.g., scrolled into view, selected, possibly animated
- Aggregation
  - Units should be "chunks" of action stream: typed strings, dialogs, macros
- Reversibility of the Undo itself
  - Support Redo as well as Undo
  - Undo to a state where user can immediately reissue the undone command, or a variant on it
    - e.g., restore selection & cursor position
- Reserve it for model changes, not view changes
  - For consistency with other applications, reserve Undo for changes to backend data
- "Undo" is not the only way to support reversibility
  - Backspace undoes typing, Back undoes browsing, Recent Files undoes file closing, scrolling back undoes scrolling
  - Forward error recovery: using new actions to fix errors

Visualizing the History

- Use Undo/Redo to browse history and view resulting application state
  - Not ideal, since user is making changes to model just to view the history
- Direct visual representation



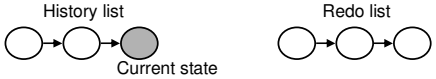
web browser history



graphical history

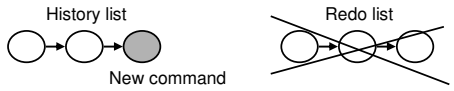
Undo and Redo Lists

- **History list** is a **script** of commands that generates the current model state
- Undo & Redo edit the script
  - Undo removes last action from history list and puts it on **redo list**
  - Redo adds back one action from redo list
  - Undo & Redo are not put in either list

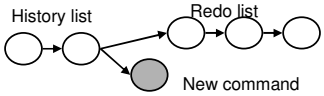


Adding New Commands to History

- New command is added to history list
  - And clears the redo list (in most apps)

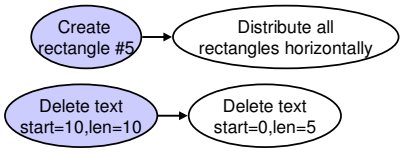


- Or new command may branch history



Removing Commands from History (Selective Undo)

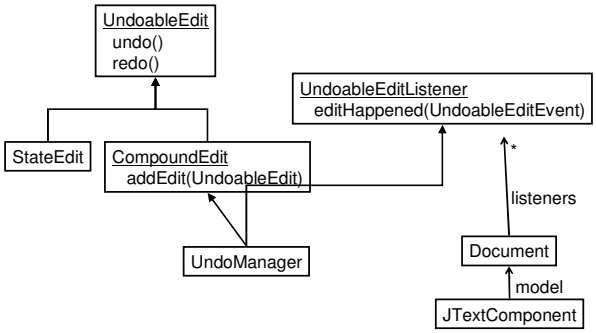
- Selective undo = deleting any action from the history list, not necessarily the last
- Selective redo = redoing any action in redo list, not necessarily the first
- Need to visualize history to choose action to undo
- Essential for multiuser applications
- Watch out for command interdependencies



Undo Representation

- History is a list of command objects
  - with undo() and redo() methods
- Command object must store enough data to implement undo and redo
  - Partial checkpoint of prior state
    - Move circle **from (0,0)** to (100,100)
    - Change text **from Times** to Arial
  - Object references vs. location descriptions
    - Insert "hello" at char position 33 vs. Insert "hello" at marker #502934
  - Relative difference vs. absolute before & after values
    - Move circle from (90,120) to (100, 100) vs. Move circle by (10,-20)

Swing Undo Architecture



Implementation Challenges

- Global changes may need to save a lot of prior state
  - e.g. whole-image operations in an image editor
- Redo of object creation must produce references usable by subsequent modification/deletion actions
  - 1: Create circle #5023 center (10,10) radius 20
  - 2: Change color circle #5023 from black to white
  - Undo 1 & 2, then redo 1
  - Redo must restore the original circle so that action 2 still refers to it
- Object references on history list prevent garbage collection of deleted objects
  - generally handled by limited history length