Undo

Arnav Gupta

April 18, 2024

Contents

1 Principles and Concepts

Benefits:

- allows recovery from errors (input/human and interpretation/computer)
  - can work quickly and without fear
- enables exploratory learning
  - try things without knowing consequences
  - try alternative solutions
- evaluate modifications

    – fast do-undo-redo cycle to evaluate last change to document

Checkpointing is a manual undo method, save the current state to rollback later.

2 Undo Patterns

Undo design choices:

1. **undoable actions**: what actions can/should be undone
2. **state restoration**: what part of UI is restored after undo
3. **granularity**: how much should be undone at a time
4. **scope**: is undo global, local, or somewhere in between

2.1 Undoable Actions

Some actions may be omitted, destructive, not easily undone, or cannot be undone.

Suggestions:

- all changes to the document/model should be undoable
- changes to the view/interface should only be undoable if tedious or requiring significant effort
- ask for confirmation before doing a destructive action that cannot be easily undone

2.2 State Restoration

User interface state should be meaningful after redo action:

- restore prior selection of objects restored by redo
- scroll to show restored objects (if necessary)
- give focus to the control where the state was restored

2.3 Granularity

A **chunk** is conceptual change from one state to another, where interaction can be divided into undoable chunks and undo reverse 1 chunk.

For drawing interactions, undo full lines, not pixels.

For text interactions, rules differ but can be by word, sentence, timing, line, etc.

Suggestions:

- ignore direct manipulation intermediate states
- delimit chunks on discrete input breaks
- chunk all changes resulting from a single interface event

2.4 Scope

One option is to treat all windows as a single model, with a single set of actions that can be undone, so on undo/redo, change window focus.

Another option is to have multiple undo/redo stacks, one for each window, and use whichever has focus (more typical).

3 Implementation

General approaches are:

- **forward undo**: start from base document, maintain list of changes to compute current document, undo by removing last change from list when computing current document
- **reverse undo**: apply change to update document but also save reverse change, undo by applying reverse change to document

A **change record** defines a single transformation to the document/model.

3.1 Forward Undo

Save baseline document state at some past point: S^*

Save change records to transform baseline document into current document state: $S = (c(b(a(S^*))))$

To undo last action, don’t apply last change record: \$S’ = (b(a(S*)))

3.2 Reverse Undo

Save complete current document state: S

Save reverse change records to return to previous state: $\{c^{-1}, b^{-1}, a^{-1}\}$

To undo last change, apply last reverse change record: $S' = c^{-1}(S)$

3.3 Implementation with Stacks

Both options require stacks:

- **undo stack**: all change records, saved as actions performed
- **redo stack**: change records that have been undone (must be reapplied with redo)

3.3.1 Reverse Undo Command Pattern

User issues a command, which is executed to create a new document state, then reverse command is pushed onto the undo stack, and redo stack is cleared.

Undo pops reverse command from undo stack and executes it to create new document state and pushes command onto redo stack.

Redo pops command off redo stack and executes it to create new document state and push reverse command onto undo stack.

Reverse Change Record Implementation Options:

- **command** pattern: save command and reverse command to change state
- **memento** pattern: save snapshots of each document state, could be complete state or difference from last state
  - executing undo moves top memento to redo stack, then uses new top of undo stack to set model
  - needs base memento in constructor, so when undo stack is empty, base memento is used

3.4 Destructive Commands

One option is to use forward command undo.

Another option is to use reverse command undo, but the un-execute command stores previous state for destructive commands (a memento). Could require a lot memory, so some application limit size of undo stack.