

# Usability

---

CSE 331

University of Washington

# User Interface Hall of Shame



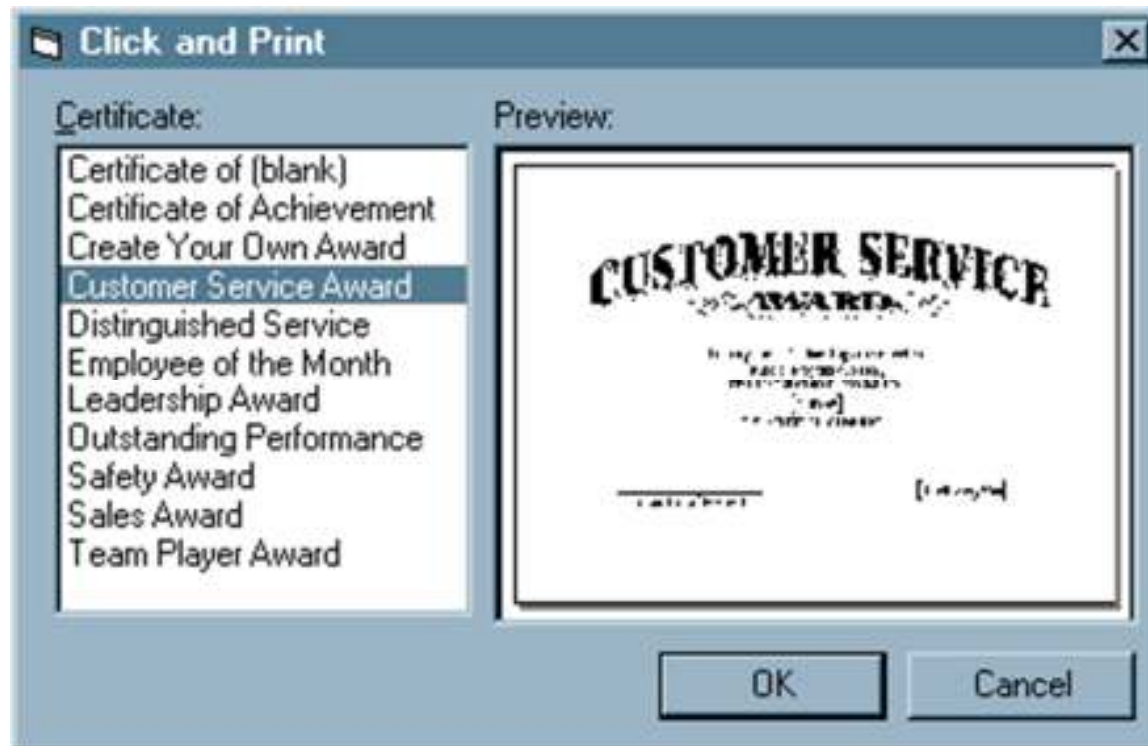
Source: Interface Hall of Shame

# User Interface Hall of Shame



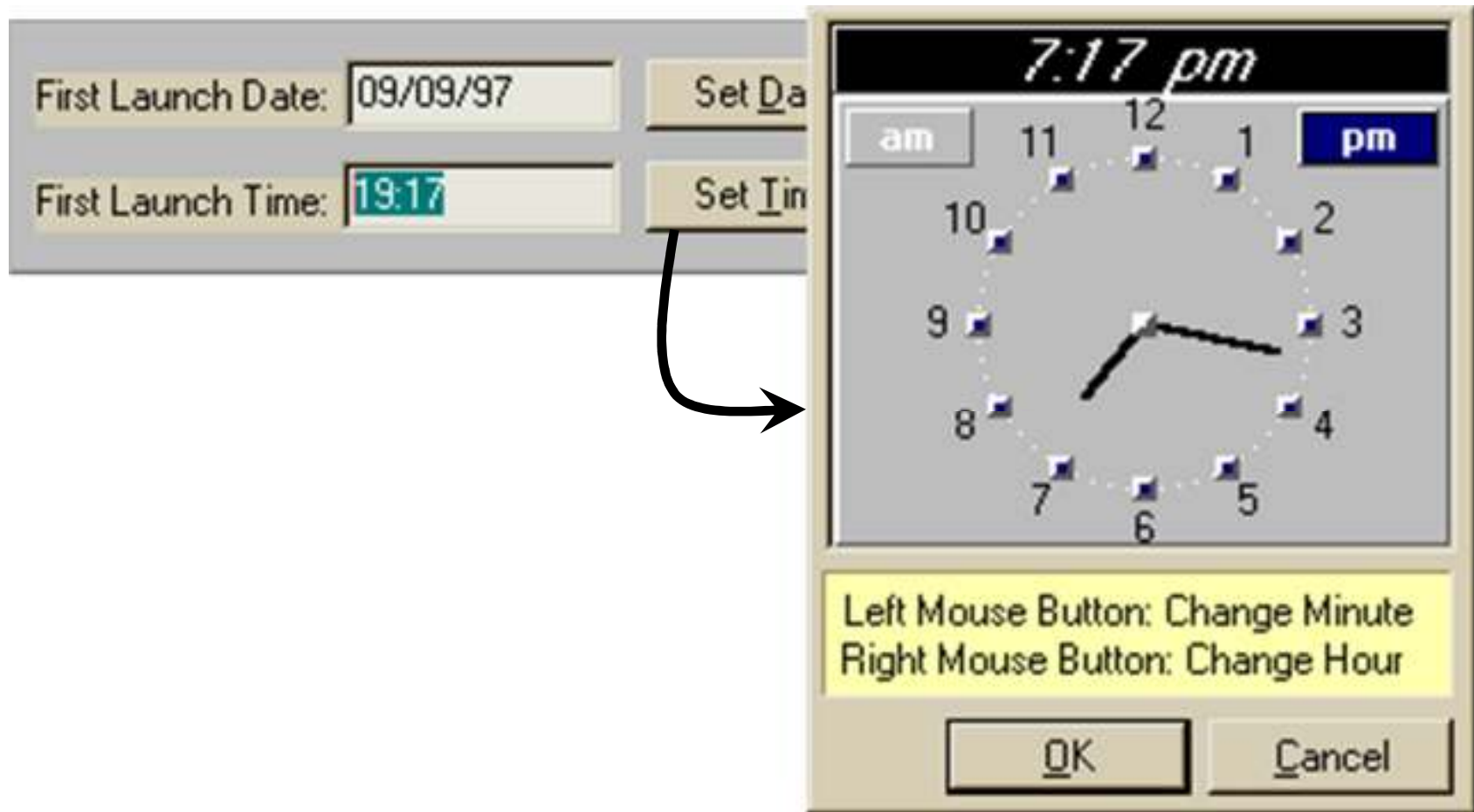
Source: Interface Hall of Shame

# Redesigning the Interface



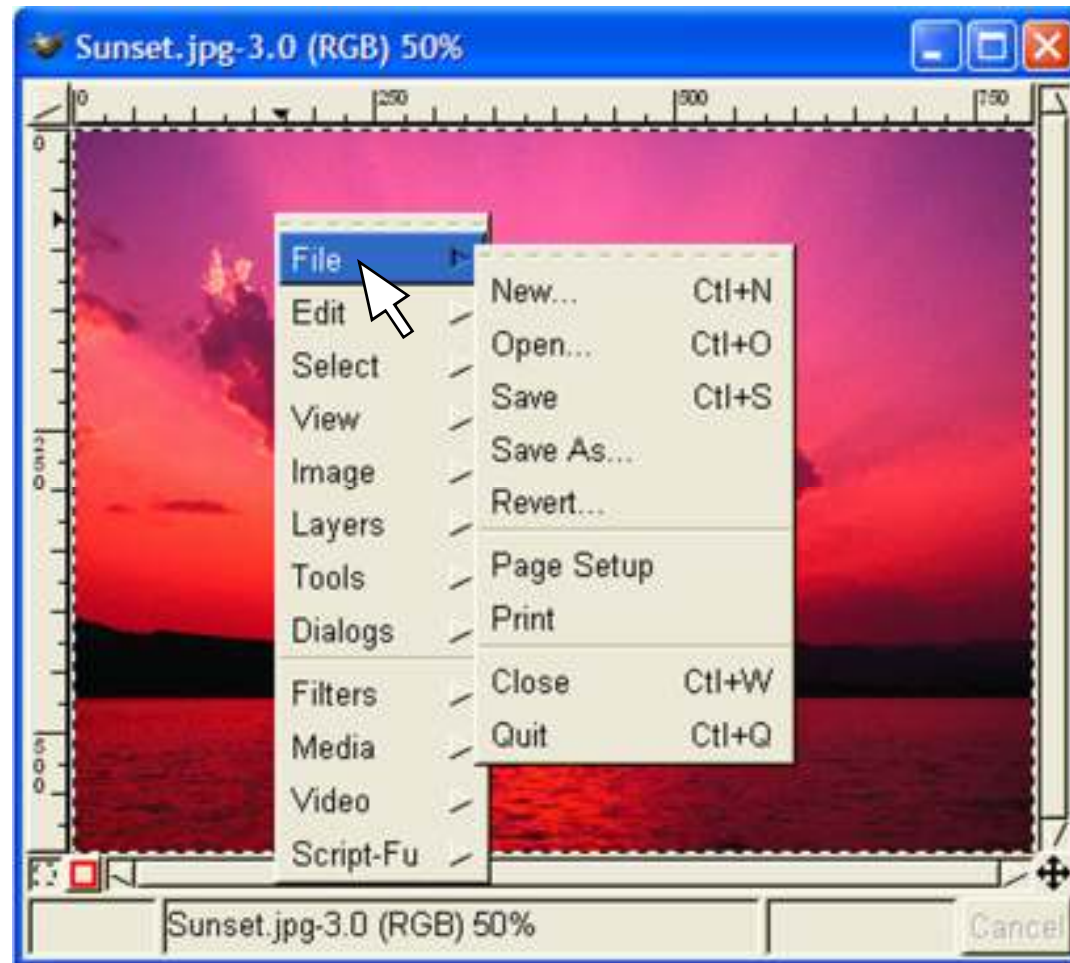
Source: Interface Hall of Shame

## Another for the Hall of Shame



Source: Interface Hall of Shame

# Hall of Fame or Hall of Shame?

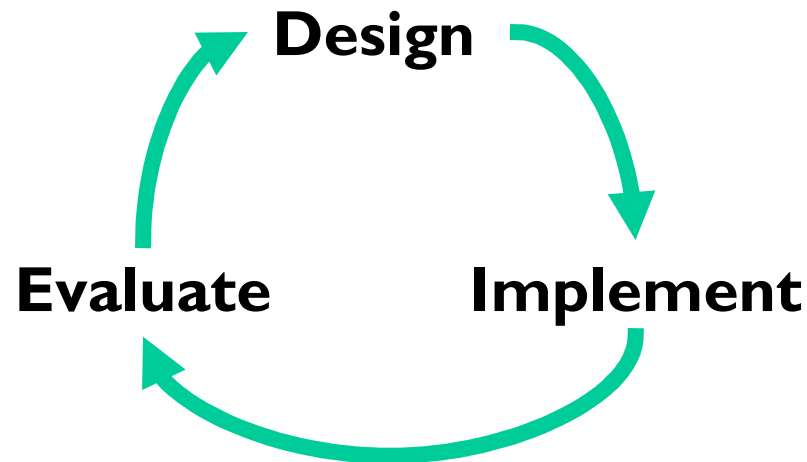


# User Interfaces Are Hard to Design

- You are not the user
  - Most software engineering is about communicating with other programmers
  - UI is about communicating with users
- The user is always right
  - Consistent problems are the system's fault
- ...but the user is not always right
  - Users aren't designers

# Iterative Design

- UI development is an iterative process

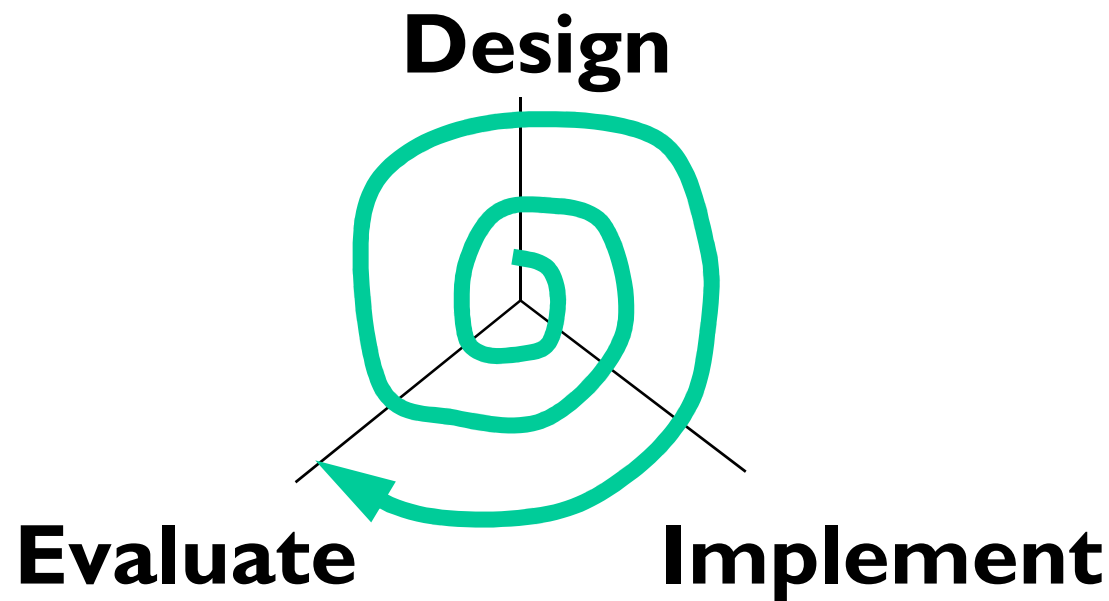


- Iterations can be costly
  - If the design turns out to be bad, you may have to throw away most of your code



# Spiral Model

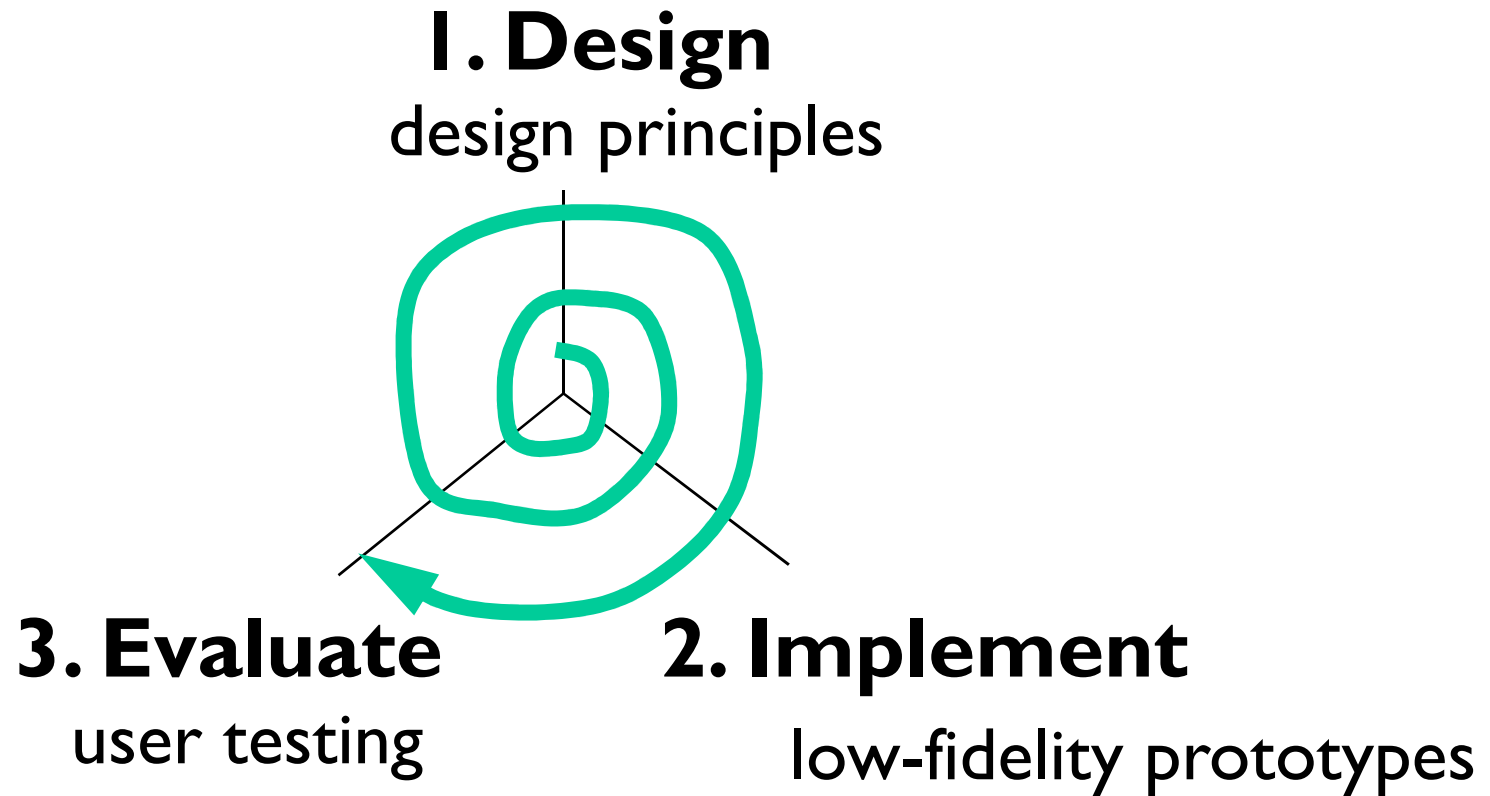
- Use throw-away prototypes and cheap evaluation for early iterations



# Usability Defined

- Usability: how well users can use the system's functionality
- Dimensions of usability
  - Learnability: is it easy to learn?
  - Efficiency: once learned, is it fast to use?
  - Memorability: is it easy to remember what you learned?
  - Errors: are errors few and recoverable?
  - Satisfaction: is it enjoyable to use?

# Lecture Outline



# Usability Goals

- Learnability
- Visibility
- Efficiency
- Error handling
- Simplicity

# Learnability



Source: Interface Hall of Shame

# Metaphorical Design



Source: Interface Hall of Shame

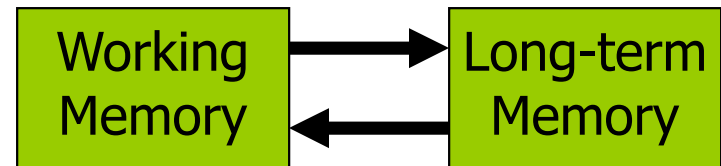
# People Don't Learn Instantly



Source: Interface Hall of Shame

## Some Facts About Memory & Learning

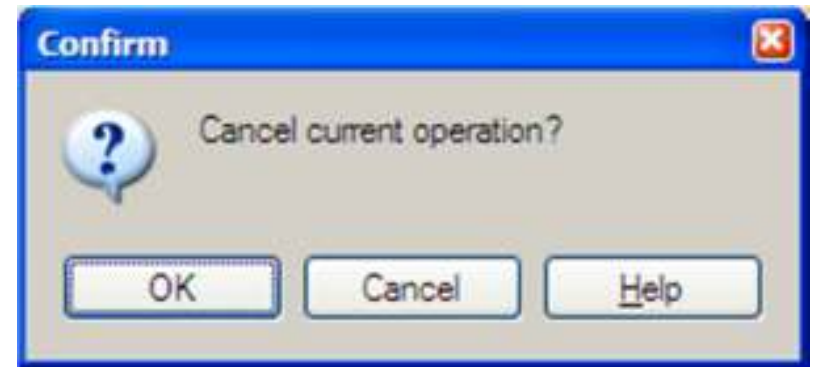
- Working memory
  - Small:  $7 \pm 2$  “chunks”
  - Short-lived: gone in  $\sim 10$  sec
  - **Maintenance rehearsal** is required to keep it from decaying (but costs attention)
- Long-term memory
  - Practically infinite in size and duration
  - **Elaborative rehearsal** transfers chunks to long-term memory





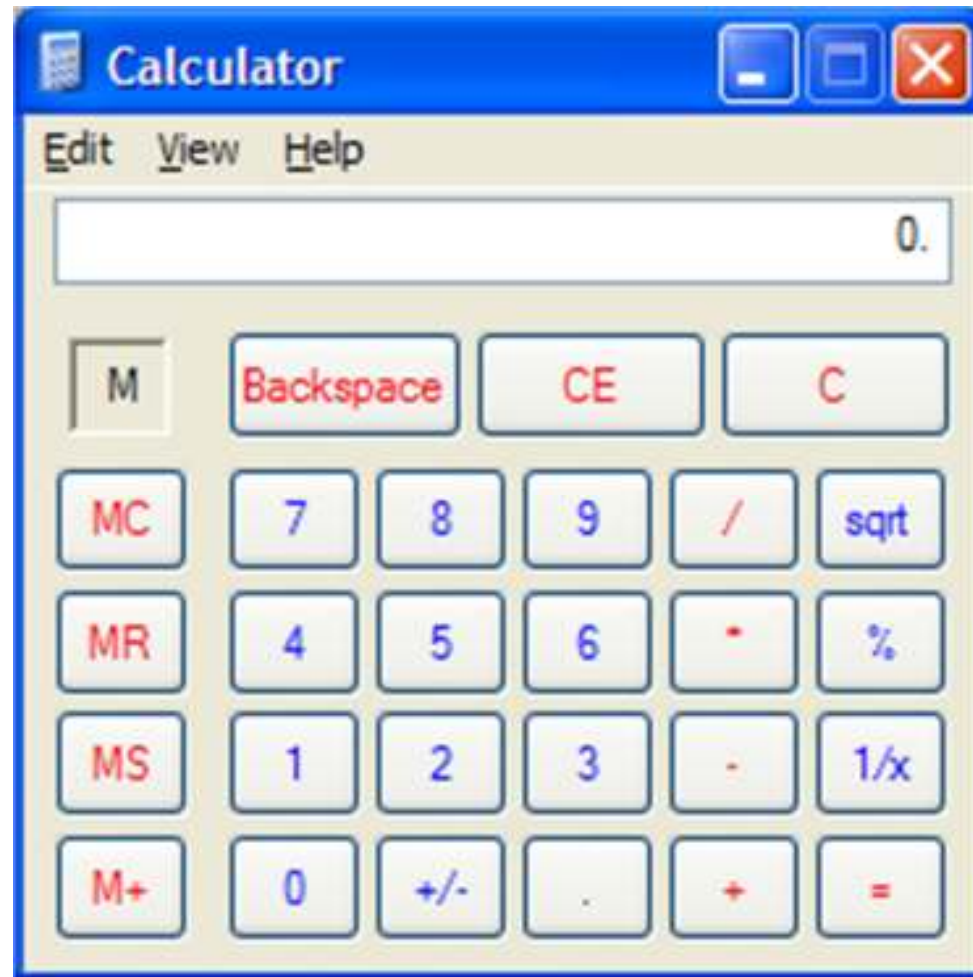
# Design Principles for Learnability

- Consistency
  - Similar things look similar, different things different
  - Terminology, location, argument order, ...
  - Internal, external, metaphorical
- Match the real world
  - Common words, not tech jargon
- Recognition, not recall
  - Labeled buttons are better than command languages
  - Combo boxes are better than text boxes

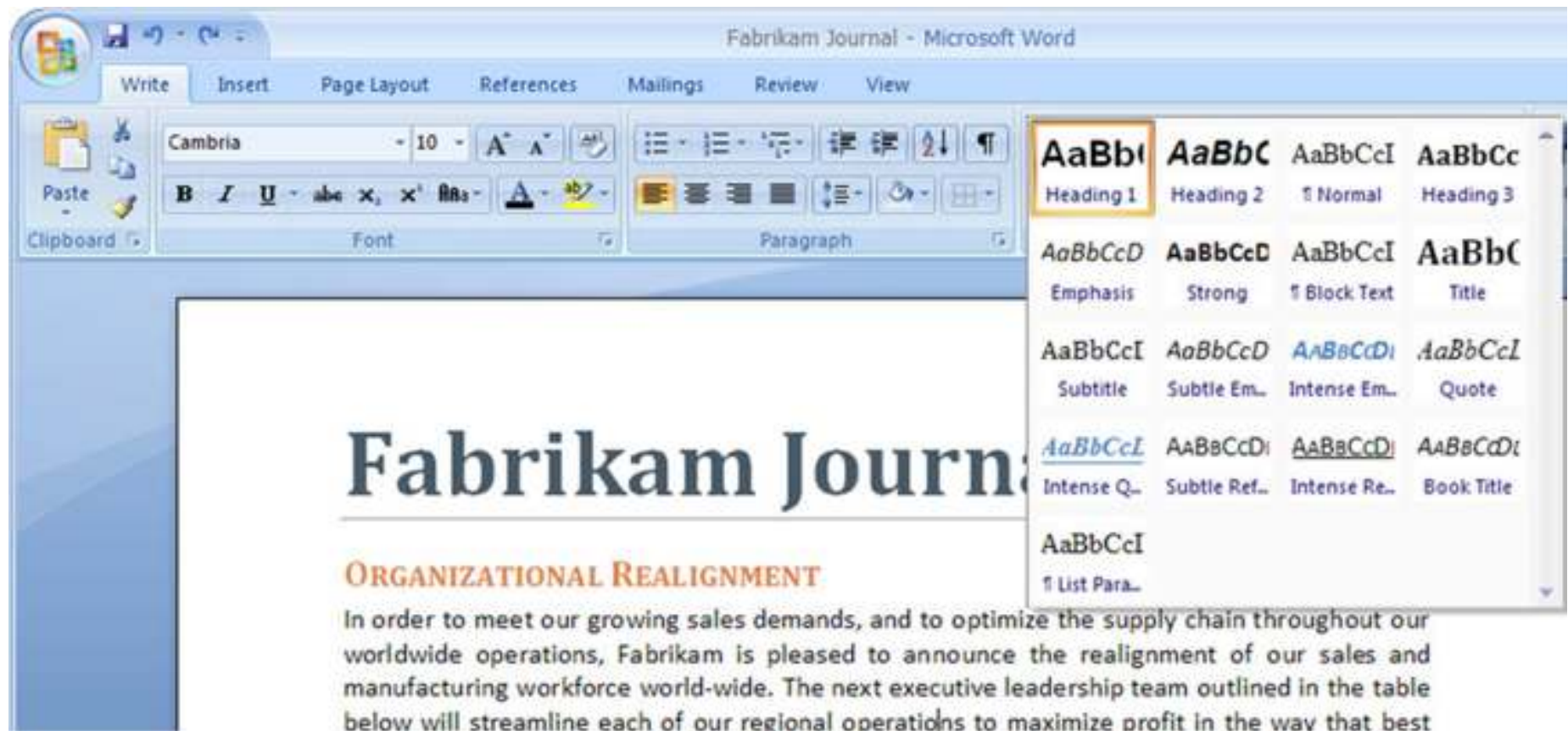


Source: Interface Hall of Shame

# Visibility



# Feedback



# Some Facts About Human Perception

- **Perceptual fusion:** stimuli  $< 100\text{ms}$  apart appear fused to our perceptual systems
  - 10 frames/sec is enough to perceive a moving picture
  - Computer response  $< 100\text{ ms}$  feels instantaneous
- **Color blindness:** many users (~8% of all males) can't distinguish red from green



normal vision

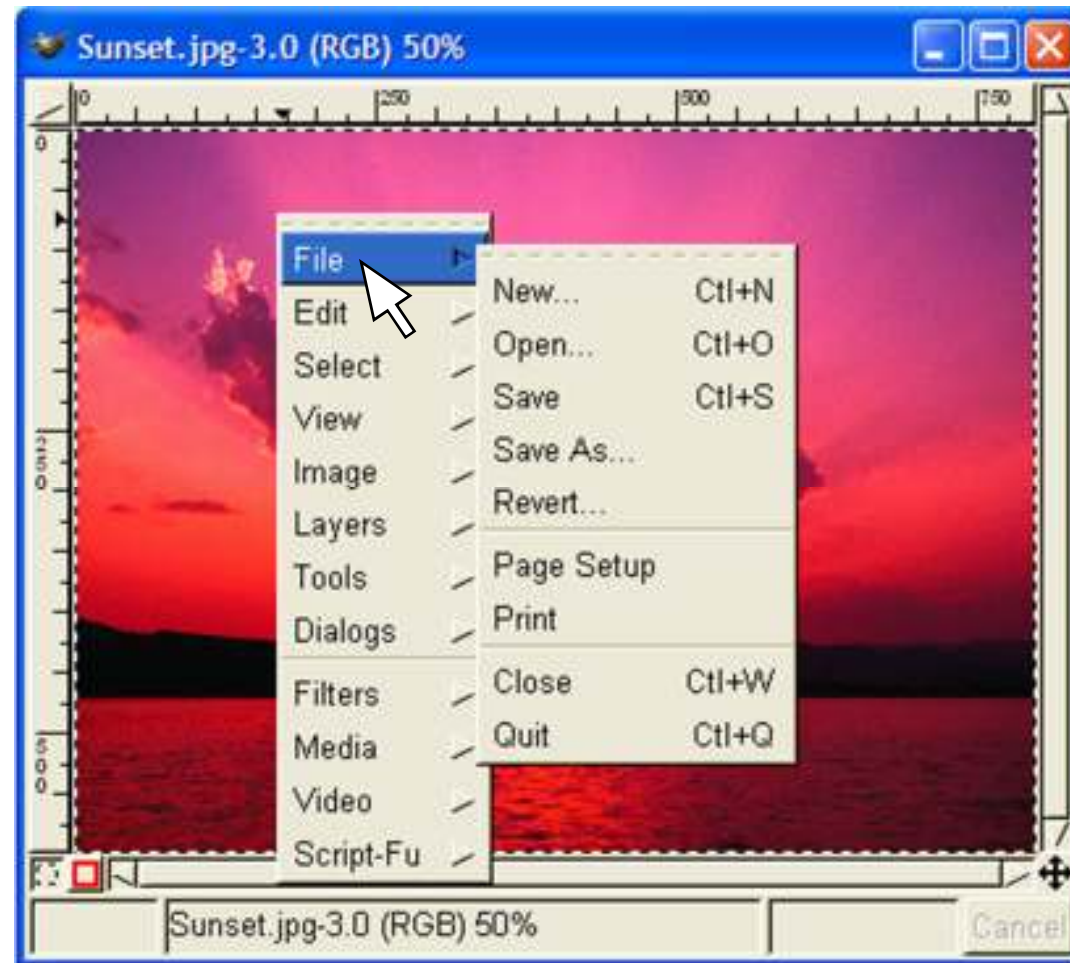


red-green deficient

# Design Principles for Visibility

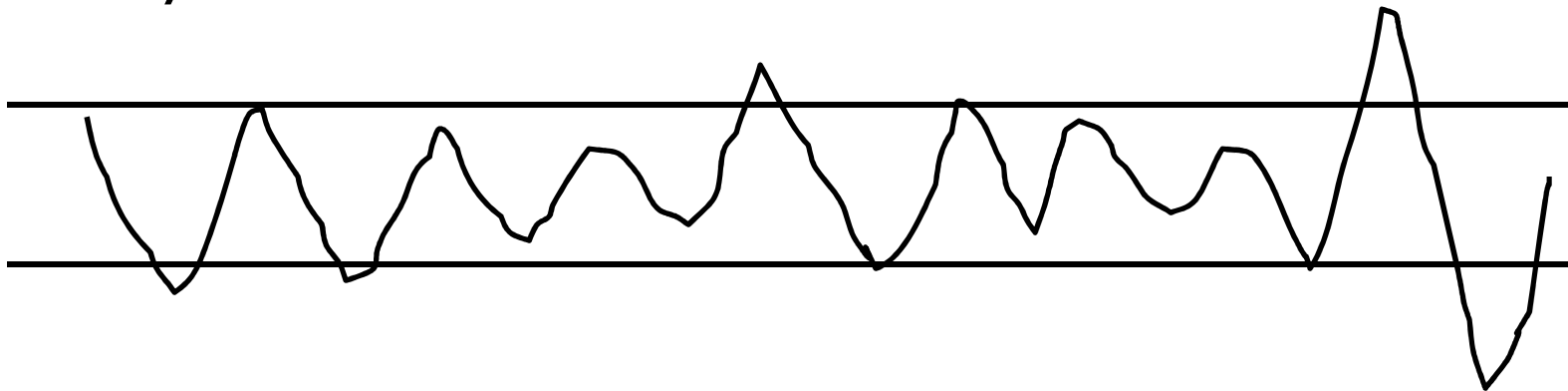
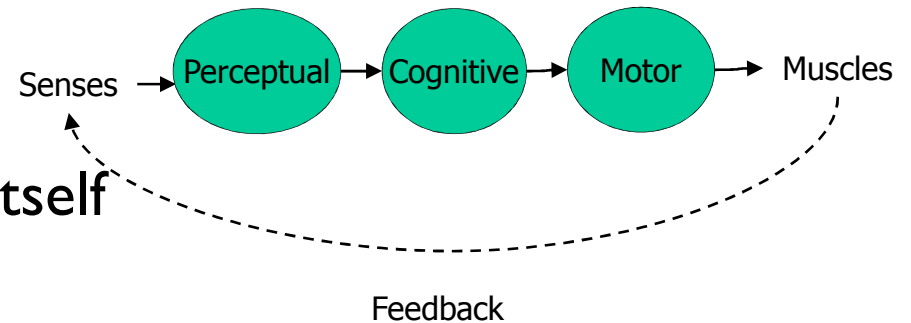
- Make system state visible: keep the user informed about what's going on
  - Mouse cursor, selection highlight, status bar
- Give prompt feedback
  - Response time rules-of-thumb
    - < 0.1 sec      seems instantaneous
    - 0.1-1 sec      user notices, but no feedback needed
    - 1-5 sec      display busy cursor
    - > 1-5 sec      display progress bar

# Efficiency



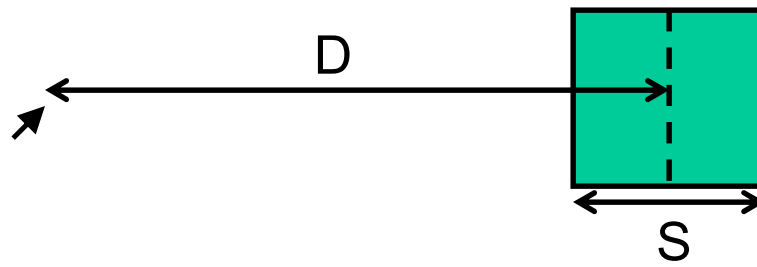
# Some Facts About Motor Processing

- Open-loop control
  - Motor processor runs by itself
  - Cycle time is  $\sim 70$  ms
- Closed-loop control
  - Muscle movements (or their effect on the world) are perceived and compared with desired result
  - Cycle time is  $\sim 240$  ms



## Pointing Tasks: Fitts's Law

- How long does it take to reach a target?

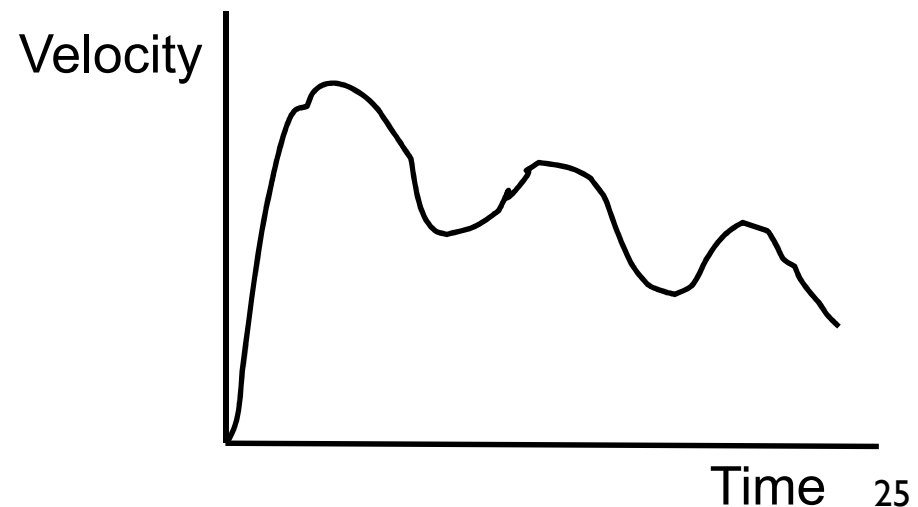
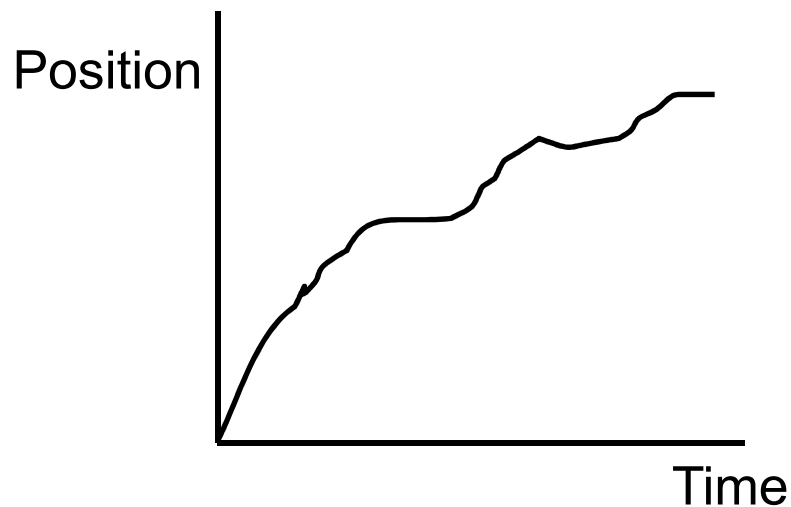


- Moving mouse to target on screen
- Moving finger to key on keyboard
- Moving hand between keyboard and mouse



# Analytical Derivation of Fitts's Law

- Moving your hand to a target is closed-loop control
- Each cycle covers remaining distance  $D$  with error  $\epsilon D$
- After 2 cycles, within  $\epsilon^2 D$  of target

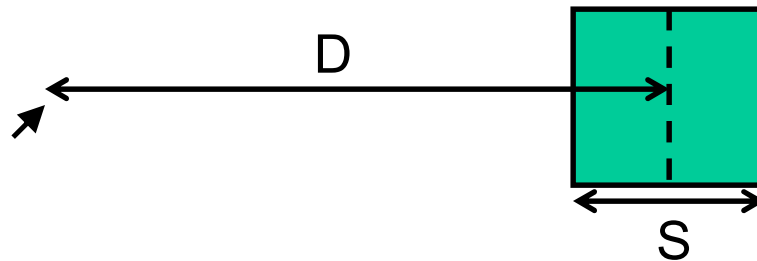


## Fitts's Law

Reaction time

Movement time

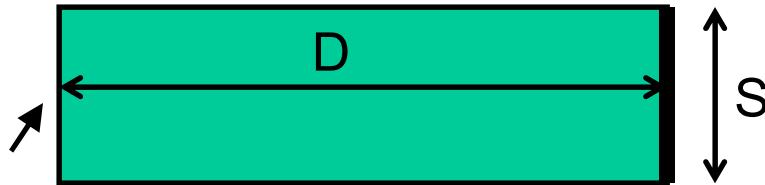
- $T = RT + MT = a + b \log(D/S)$



- $\log(D/S)$  is the *index of difficulty* of the pointing task

# Path Steering Tasks

- Fitts's Law applies only if path to target is **unconstrained**
- But the task is much harder if path is constrained to a tunnel

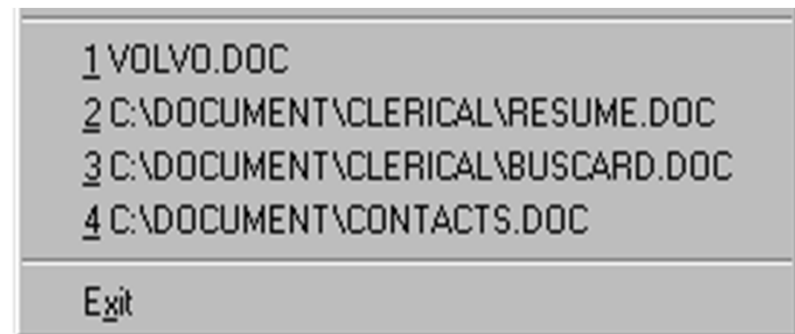


$$T = a + b (D/S)$$

- This is why cascading menus are slow!

# Design Principles for Efficiency

- Fitts's Law and Steering Law
  - Make important targets big, nearby, or at screen edges
  - Avoid steering tasks
- Provide shortcuts
  - Keyboard accelerators
  - Styles
  - Bookmarks
  - History



Source: Interface Hall of Shame

## Mode Error

- Modes: states in which actions have different meanings
  - Vi's insert mode vs. command mode
  - Drawing palette
- Avoiding mode errors
  - Eliminate modes entirely
  - Visibility of mode
  - Spring-loaded or temporary modes
  - Disjoint action sets in different modes

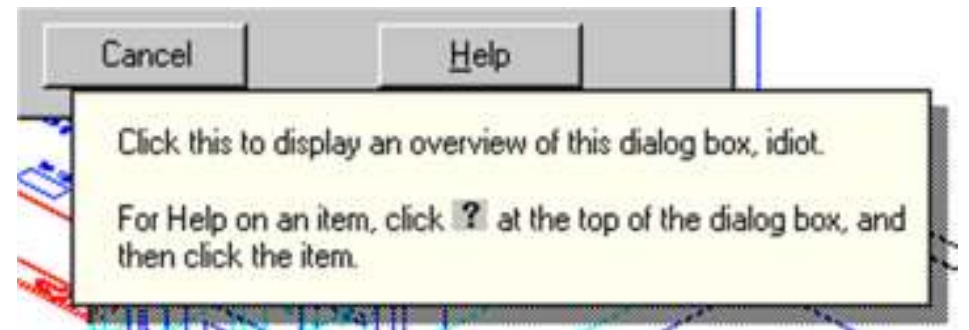


# Confirmation Dialogs



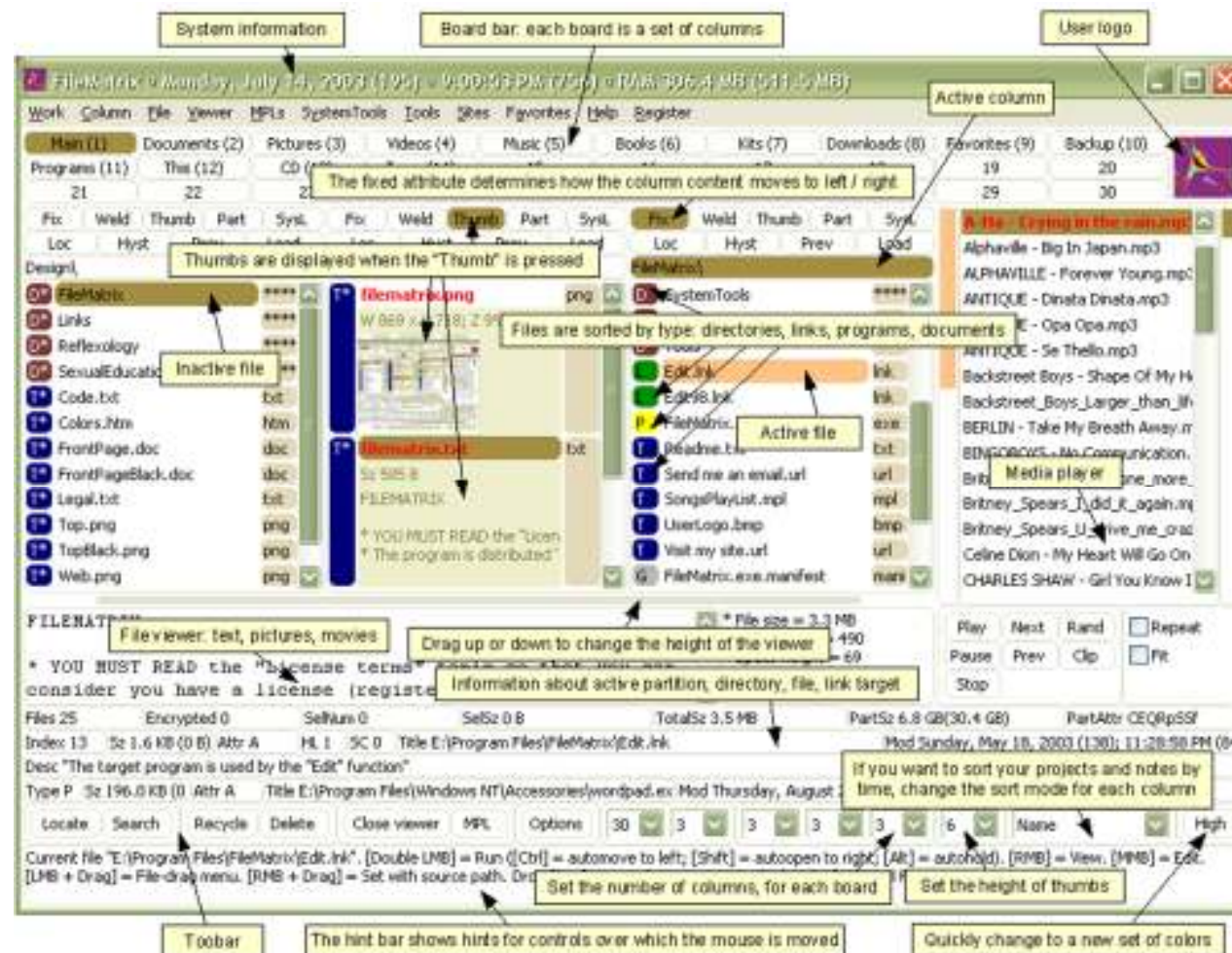
# Design Principles for Error Handling

- Prevent errors as much as possible
  - Selection is better than typing
  - Avoid mode errors
  - Disable illegal commands
  - Separate risky commands from common ones
- Use confirmation dialogs sparingly
- Support undo
- Good error messages
  - Precise
  - Speak the user's language
  - Constructive help
  - Polite



Source: Interface Hall of Shame

# Simplicity



Source: Alex Papadimoulis



# Simplicity



# Design Principles for Simplicity

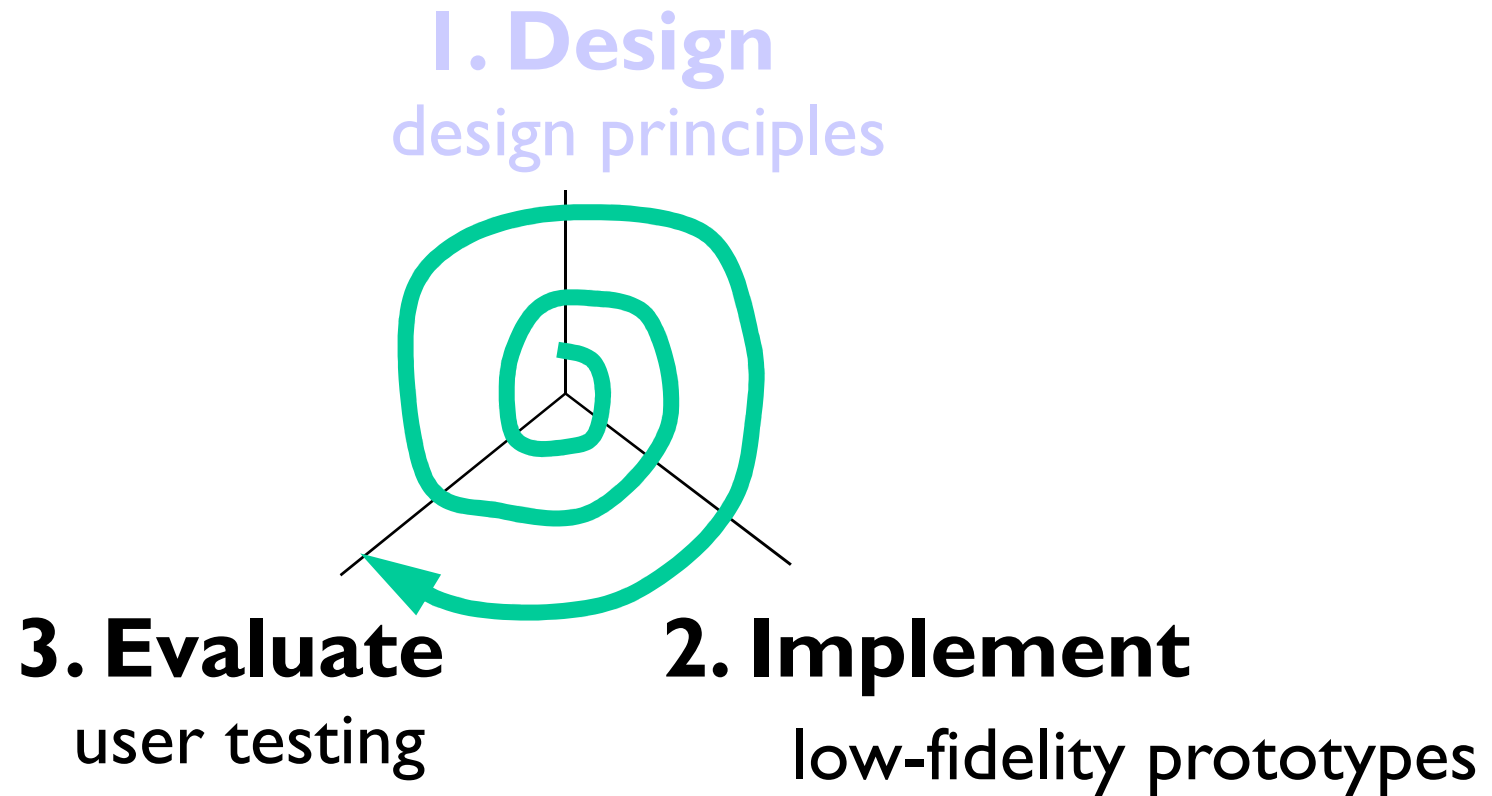
- “Less is More”
  - Omit extraneous information, graphics, features
- Good graphic design
  - Few, well-chosen colors and fonts
  - Group with whitespace
- Use concise language
  - Choose labels carefully



# Document your system

- Write the user manual
  - Program and UI metaphors
  - Key functionality
  - Not: exhaustive list of all menus
- What is hard to describe?
- Who is your target user?
  - Power users *need* a manual
  - Casual users might not
  - Piecemeal online help is no substitute

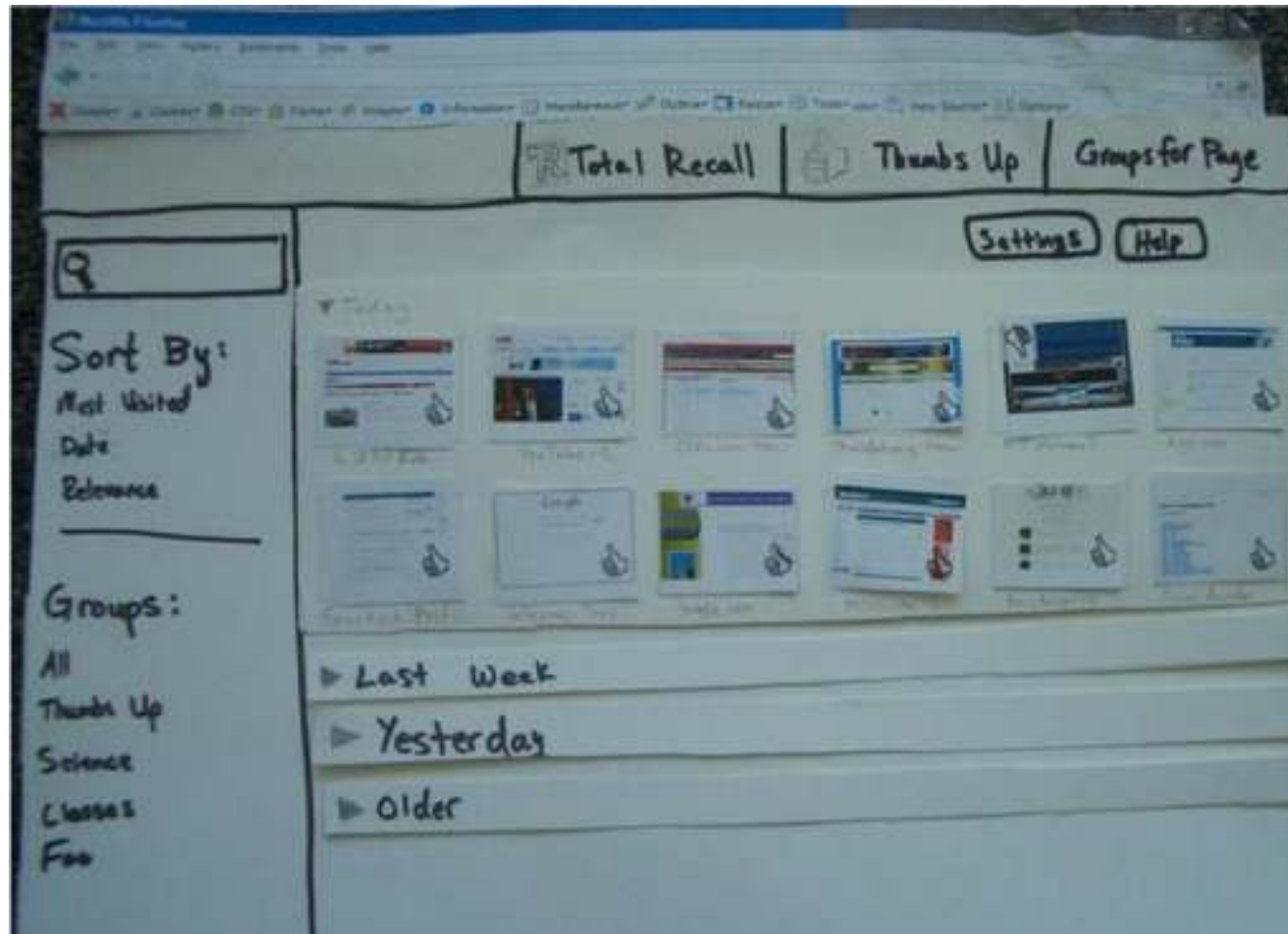
# Lecture Outline



# Low-fidelity Prototypes

- Paper is a very fast and effective prototyping tool
  - Sketch windows, menus, dialogs, widgets
  - Crank out lots of designs and evaluate them
- Hand-sketching is OK – even preferable
  - Focus on behavior & interaction, not fonts & colors
  - Similar to design of your data structures & algorithms
- Paper prototypes can even be executed
  - Use pieces to represent windows, dialogs, menus
  - Simulate the computer's responses by moving pieces around and writing on them

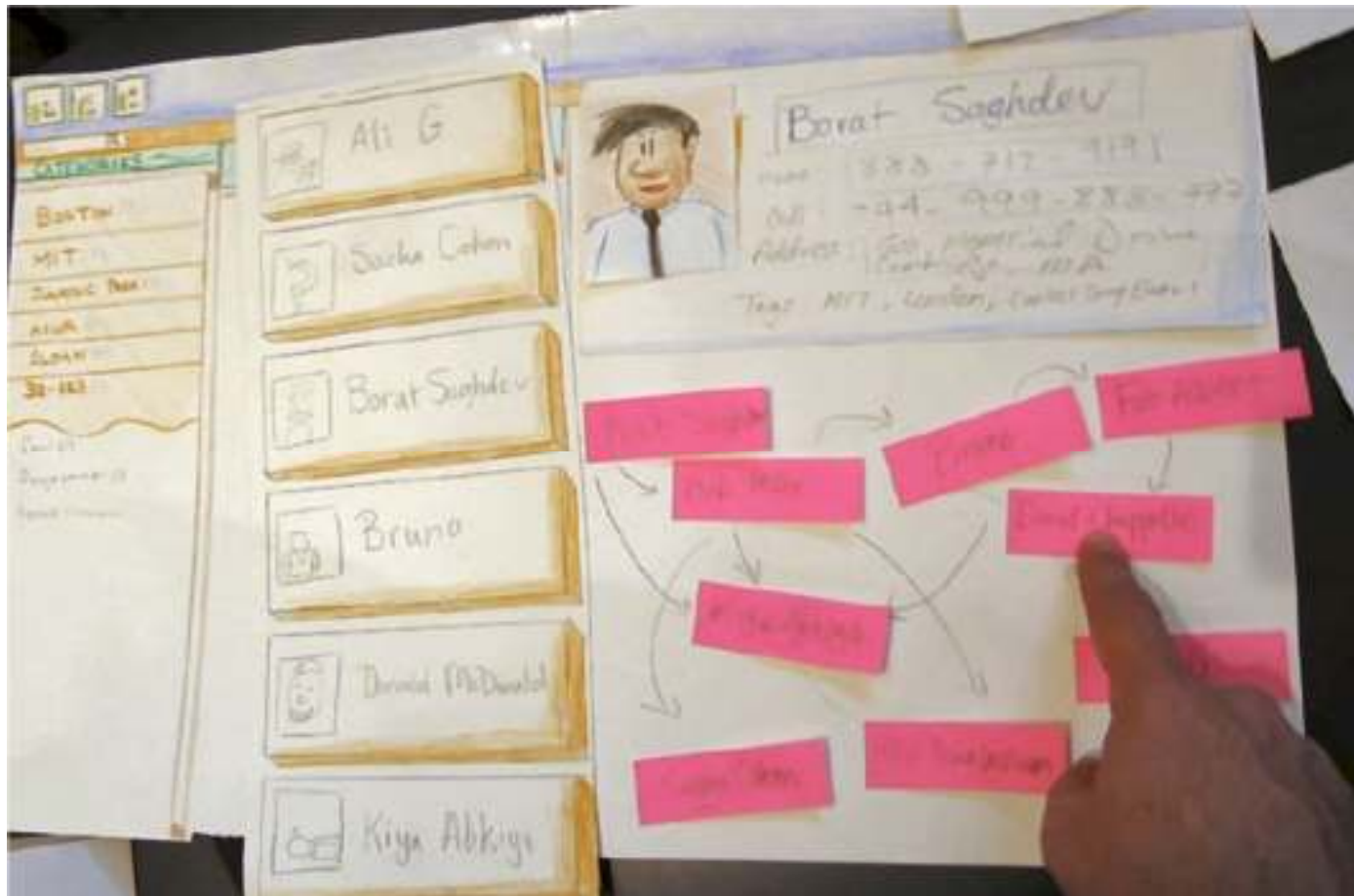
# Paper Prototypes



# Paper Prototypes



# Paper Prototypes





# User Testing

- Start with a prototype
- Write up a few representative tasks
  - Short, but not trivial
  - e.g.: “add this meeting to calendar”,  
“type this letter and print it”
- Find a few representative users
  - 3 is often enough to find obvious problems
- Watch them do tasks with the prototype

# How to Watch Users

- Brief the user first (being a test user is stressful)
  - “I’m testing the system, not testing you”
  - “If you have trouble, it’s the system’s fault”
  - “Feel free to quit at any time”
  - Ethical issues: informed consent
- Ask user to think aloud
- Be quiet!
  - Don’t help, don’t explain, don’t point out mistakes
  - Sit on your hands if it helps
  - Two exceptions: prod user to think aloud (“what are you thinking now?”), and move on to next task when stuck
- Take lots of notes

# Watch for Critical Incidents

- Critical incidents: events that strongly affect task performance or satisfaction
- Usually negative
  - Errors
  - Repeated attempts
  - Curses
- Can also be positive
  - “Cool!”
  - “Oh, now I see.”

# Summary

- You are not the user
- Keep human capabilities and design principles in mind
- Iterate over your design
- Write documentation
- Make cheap, throw-away prototypes
- Evaluate them with users

# Further Reading

- General books on usability
  - Johnson. *GUI Bloopers: Don'ts and Dos for Software Developers and Web Designers*, Morgan Kaufmann, 2000.
  - Jef Raskin, *The Humane Interface*, Addison-Wesley 2000.
  - Hix & Hartson, *Developing User Interfaces*, Wiley 1995.
- Low-fidelity prototyping
  - Rettig, “Prototyping for Tiny Fingers”, CACM April 1994.
- Usability heuristics
  - Nielsen, “Heuristic Evaluation.”  
<http://www.useit.com/papers/heuristic/>
  - Tognazzini, “First Principles.”  
<http://www.asktog.com/basics/firstPrinciples.html>