BIM423 - Software Engineering

Week 9 - *Usability & User Interfaces*

**Some slides are modified from ocw.mit.edu*

Hall of Shame

Is this a good design?

Hall of Shame

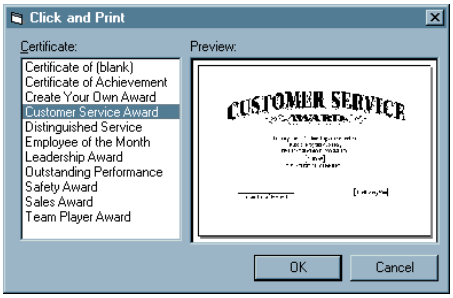
It is WYSIWYG (*What-You-See-Is-What-You-Get*)
Bad use of scroll bar
Normally scrollbars are used for scrolling content horizontally

Hall of Shame

Behaviour is not consistent with user habits
- The help message is too long for a simple task
- How can a user find a template that they found in the past
- They need to remember the exact location of the scroll

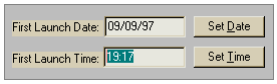
How can we make it better?

Hall of Shame



Using the correct controls

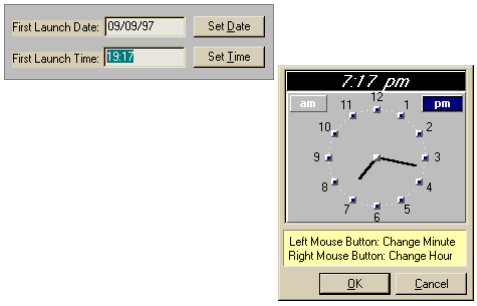
Hall of Shame



AutomatePro Scheduling an event

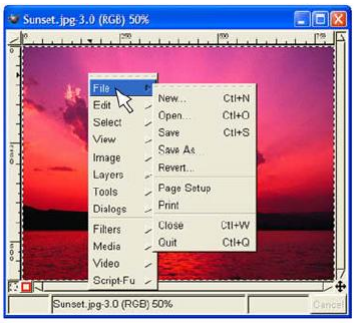
What would you expect when you click set time?

Hall of Shame



Someone put in a lot of effort to create this useless interface

Continue…



GIMP program doesn't have any menu bar

Everything is a context menu accessed with a right-click

Again consistency

The User Interface Is Important

- User interface strongly affects perception of software
  - Usable software sells better
  - Unusable web sites are abandoned
- Perception is sometimes superficial
  - Users blame themselves for UI failings
  - People who make buying decisions are not always end-users

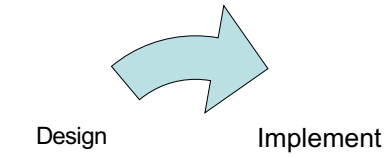
9

User Interfaces are Hard to Design

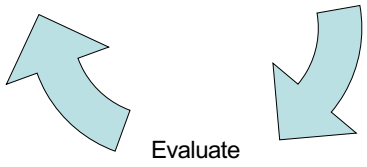
- You are not the user
  - Most software engineering is about communicating with other programmers
  - UI is about communicating with users
- The user is always right
  - Consistent problems are the system's fault
- … but the user is not always right
  - Users aren't designers, our job is to help them

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



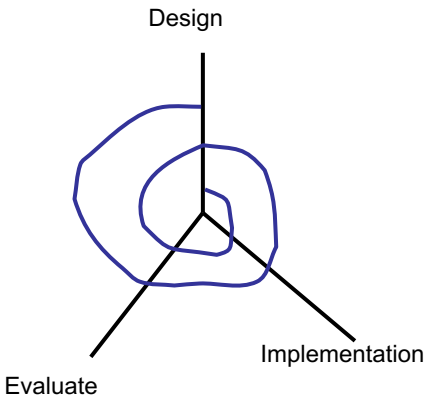
Spiral Model

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

I. Design Principles

II. Low-fidelity prototypes

III. User testing



Usability Defined

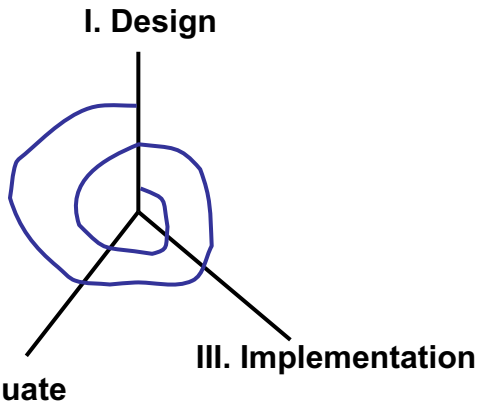
- **Usability:** How well users can use the system's functionality

- Dimensions of usability
  - Learnability: Is it easy to learn?
  - Efficient: 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?

I. Design Principles

II. Low-fidelity prototypes

III. User testing



Usability Goals

- Learnability
  - Easy to learn or not
- Visibility
  - Interface gives feedback, makes its state easy for user to see
- Efficiency
  - Is it fast to operate or not?
- Error Handling
  - Frequency and cost of errors
- Simplicity
  - Fewer parts easier to understand and use

Jakob Nielsen's 10 heuristics

(1) Learnability

Intuitive?

User-friendly?

Scrollbar in this context is unfamiliar and inconsistent

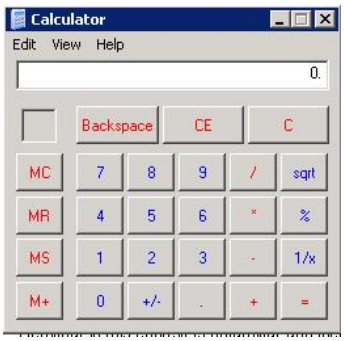
Some Facts About Memory and Learning

- Working Memory
  - Small: 7+/-2 “chunks”
  - Short-lived: gone in 10 seconds
  - Maintenance rehearsal is required to keep it from decaying
- Long term memory
  - Practically infinite in size and duration
  - Elaborative rehearsal transfers chunks to long-term memory

Design Principles for Learnability

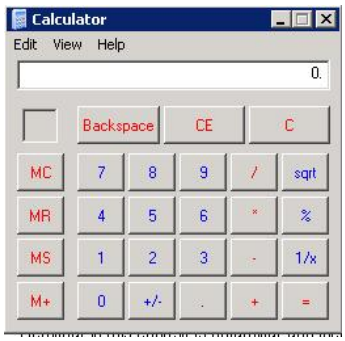
- Consistency
  - Similar things look and act similar
  - Different things look and act similar
  - Consistency of wording location argument order
  - Internal consistency: within your UI
  - External consistency: with other UIs
- Match the real world
  - Use common words not technical jargon
- Recognition, not recall
  - Labeled buttons are better than command languages
  - Combo boxes are better than text boxes

(2) Visibility



What is wrong with the default windows calculator?

(2) Visibility



Sqrt button

* For multiplication

Backspace

For a calculator, software interface can be better than a regular calculator to show current state (i.e. 3+4=7)

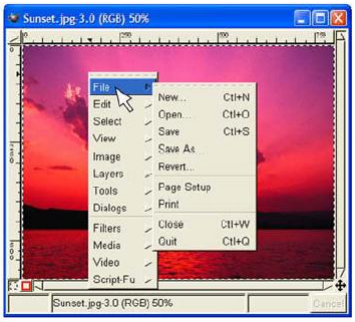
Some Facts About Human Perception

- Perceptual fusion: stimuli < 100ms apart
  - Computer response < 100ms feels instantaneous
    - Imagine a word processing program that takes more than 100ms to display characters on the screen

Design Principles for Visibility

- Make the system state visible: keep the user informed about what is 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
    - > 5 sec display progress bar

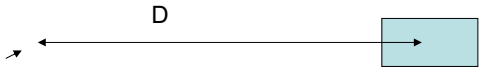
(3) Efficiency



How quickly an expert user can operate the system

Pointing Tasks

- How long does it take to reach a target?

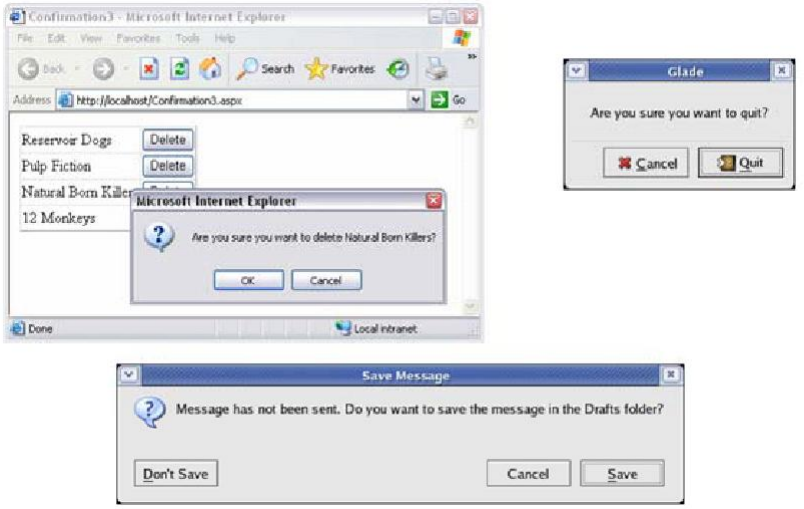


- Moving mouse to target on screen
- Moving finger to key on keyboard
- Moving hand between keyboard and mouse

Design Principles for Efficiency

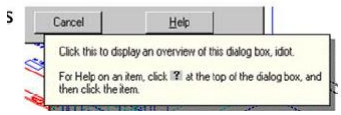
- Make important targets big, nearby or at screen edges
- Avoid steering tasks
- Provide Shortcuts
  - Keyboard accelerators
  - Styles
  - Bookmarks
  - History

(4) Error Handling: Confirmation Dialogs

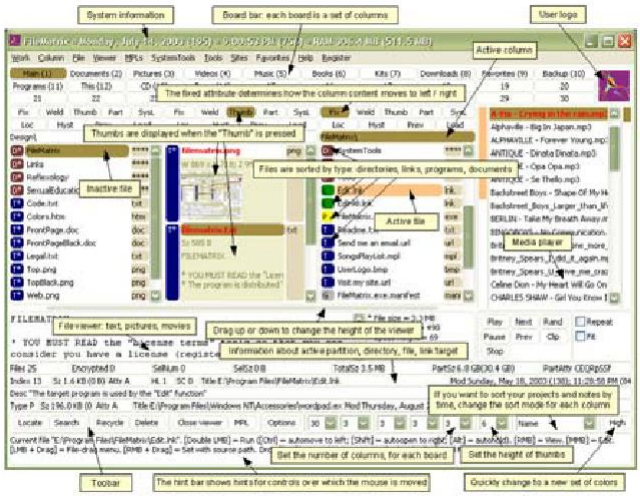


Design Principles for Error Handling

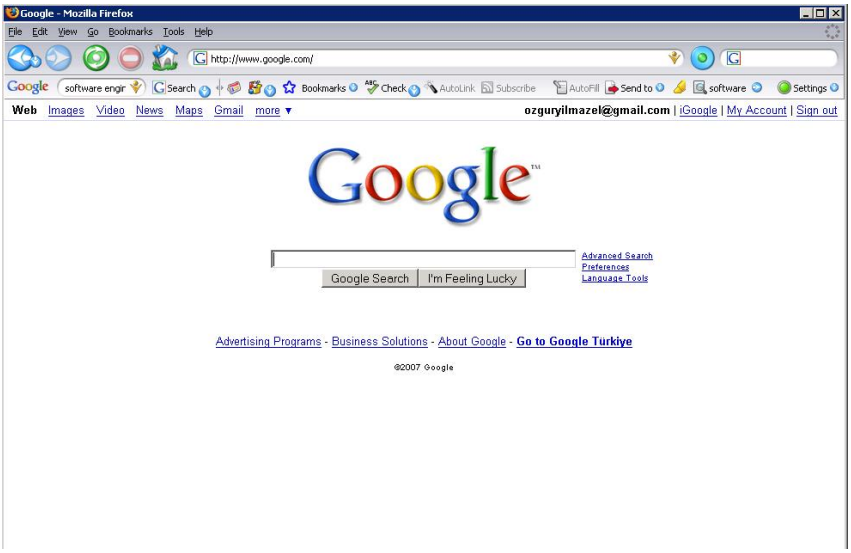
- Prevent Errors as much as possible
  - A selection is better than typing
  - Disable illegal commands
  - Separate risky commands from common ones
- Use confirmation dialogs sparingly
- Support Undo
- Good error messages
  - Precise
  - Polite
  - Constructive help



(5) Simplicity



(5) Simplicity



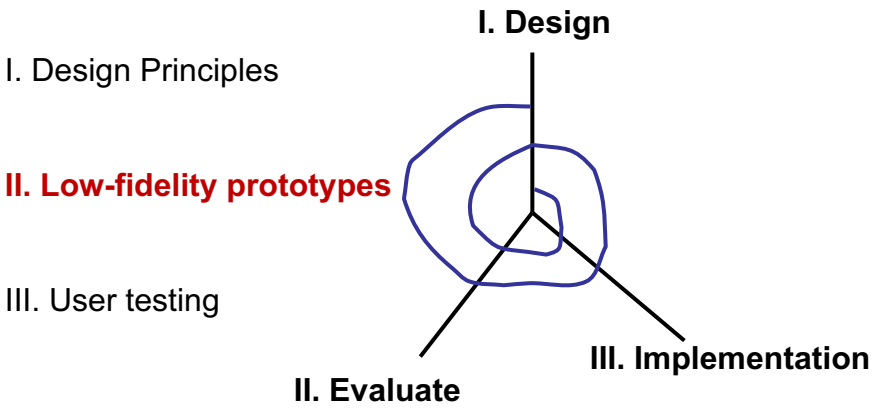
Design Principles for Simplicity

- 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

I. Design Principles

II. Low-fidelity prototypes

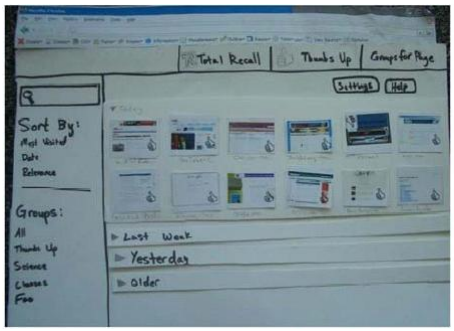
III. User testing



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 Ok
  - Focus on behavior and interaction, not fonts and colors
- Paper prototypes can even be executed
  - Use pieces to represent windows, dialogs, menus
  - Simulate the computer's response by moving pieces around and writing on them

Paper Prototypes

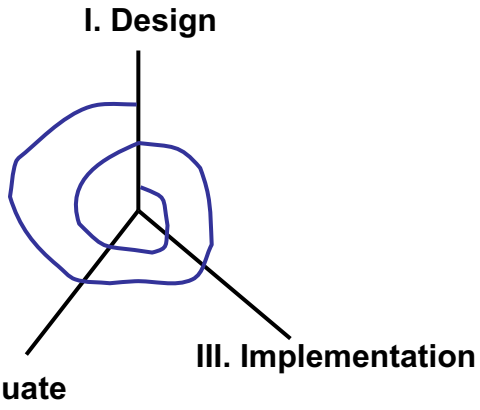


Paper Prototypes

I. Design Principles

II. Low-fidelity prototypes

III. User testing



User Testing

- Start with a prototype
- Write up a few representative tasks
  - Short but not trivial
  - E.g.
    - Add this meeting to the 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 am testing the system not testing you
  - If you have trouble, it's the system's fault
  - Feel free to quit at any time

  Always to an Informed Consent
- Ask the 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: make user to think aloud ("what are you thinking now?") and move on to the next task when stuck
- Take lots of notes – or even record the session

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
- Make cheap throw away prototypes
- Evaluate them with users

Further Reading

- Nielsen “Heuristic Evaluation”
- Tognazzini “First Principles”
- “GUI Bloppers: Don’ts and Dos for Software Developers and Web Designers” Johnson, Morgan Kaufmann, 2000