External Design: Human Factors and User Interface

Not half of what you need to know, but better than nothing

We know a few things ... from psychology and human factors research

- Characteristics of human memory
- Characteristics of perception
- Problem solving behavior

Human Memory

- Short-term memory
  - Fast but very small (5 +/- 2 items)
  - Does not last long
- Long-term memory
  - Very large, but slow
    - retrieval time and difficulty depends on frequency of use
    - some tasks are harder than others (e.g., recall vs. recognition)
  - Highly organized: users discover and exploit rules
- Usable designs minimize memory “load”

Frequency of Use

- Consider two users of an airline reservations system
  - Professional travel agent: Uses the system every day, for hours at a time
  - Traveler with an online account (Expedia, Travelocity etc.): Uses the system 12 times/year
- Frequent user can memorize commands
  - Optimize for few keystrokes, short command sequences, few transaction waits
- Infrequent user will not memorize
Know Your User
The first and most important principle of interface design

- User characteristics
  - Frequent or infrequent user?
  - What expertise?
- Make appropriate tradeoffs
  - Ease of learning vs. ease of use
  - Helpfulness vs. speed

What Does Your User Know?

- Frequent mistake: Assuming the user knows what you know
- Remedies:
  - Observe untrained users (and not yourselves)
    - Really observe: Diagnose their mistakes
  - See the system through their eyes
    - A supplement, not a replacement for real observation

Recognition vs. Recall

A—Can you name the nations of Europe?
B—Is Luxembourg a nation in Europe?
- B is easier than A because recognition is easier than recall
- So: We should replace recall tasks with recognition tasks
  - A (long-term or short-term) memory load reduction: putting part of memory burden outside the user’s head.

Replacing Recognition with Recall

- Most important for
  - Novice users (of the application)
    - Mainly because they have fewer clues for guessing
  - Infrequent users (even experts)
    - Long-term memory, e.g. of commands, depends on frequency of use
    - Very frequent users can and will memorize
      - from use, not from a user manual; disclose shortcuts during normal operation
  - Information that changes
    - ex: file names
Visual Representation of State
Example: Folder Display on Mac OS 9 Desktop

- Finder displays folder contents
- Icon indicates state (Open or Closed)
- Window bar indicates currently active window
- But ...
  - this snapshot was saved as a file that was not visible on the desktop

Visual Representation of State
Windows 95 v. MacOS 9

- Windows95 desktop looks almost like a Macintosh
  - but doesn’t behave like it
- Open/closed state of folder is not indicated by folder icon
- Result: User mistakes
  - Attempting to open folders that are already open
- But ...
  - At least the snapshot was saved like cut-and-paste selections

Perception

- Visual perception is excellent for patterns and variations
  - But hearing is much faster and wider
- Visual perception has a narrow “fovea”
  - Wide field of view is partly an illusion; we see details outside the fovea only by shifting attention
  - Shifting the fovea is “expensive” in effort and lost concentration

Designing for the Fovea

- Avoid scattering detail information

If attention is focused up here

Message down here cannot be read
Patterns and Attention

- People are very good at focusing on variations and ignoring regularity
  - Avoid “noise”
  - Use variation or change to draw attention (but only when needed)

Feedback

- Another aspect of reducing memory load
  - Principle: User should never need to remember or guess the current state
  - Techniques
    - Maintain a visual representation of state as it changes
      - anything user must otherwise remember
    - Acknowledge every user action immediately
    - For long operations, provide progress indicators

Time

Response time requirements don’t have to be arbitrary

- 30hz or better looks continuous
  - Not important just for video — e.g., consider drawing with the mouse
- 10hz or better seems “immediate”
  - All forms of “echo” should take less than 0.1 second, including keystrokes and (graphic) button pushes
- Attention shifts in approximately 1 second
  - User speed and accuracy falls rapidly when response exceeds 1 second

Minimizing Pauses

- Optimize tasks by removing unnecessary pauses (0.1 second or greater)
  - Bad example: Unnecessary page transitions in DuckWeb
- Based on intended or observed use
  - Observing or tracking actual use is best
Ears are faster than eyes!

- Sound is under-used in interface design
  - Mostly for gaining attention, or just for entertainment; overcoming limitation to visual fovea
- If very fast temporal patterns are required, sound is our most developed sense
  - Both for minimum relative spacing, and for complex temporal patterns

Making Difficult Tasks Simple

  - Use both knowledge in the world and knowledge in the head
  - Simplify the structure of tasks
  - Make things visible: bridge the gulfs of Execution and Evaluation
  - Get the mappings right
  - Exploit constraints, both natural and artificial
  - Design for error
  - When all else fails, standardize

Knowledge in the world

- “Affordances” indicate how to use things
  - Example: shape of door handle says “push” or “pull”
    - If it needs a label, it is badly designed
- How to use an object should be obvious
  - If it looks like a button, push it!
- Constraints prevent mistakes
  ➫ Ex., “greying out” inapplicable commands

Permissive vs. Preemptive

- Principle: The user should be in charge
- Permissive interfaces allow the user to choose any sensible next action
- Preemptive interfaces restrict choice
- Example:
  - Enter file name: `ls`
Avoiding Preemption

- Commands instead of prompts
  – or in addition
- Multiple contexts (e.g., windows)
- Postfix syntax (esp. with mouse)
- Limit modes

- What can we do on the web?

Modes

A mode is a state that lasts for a period of time, is not associated with a particular object, and has no role other than to place an interpretation on operator input.

(Larry Tessler)

- Example: vi is a “modal” editor because the insert and command modes place an interpretation on keyboard input (e.g., “j”).
- Drawing program “tools” are usually modes

Modes are Sometimes O.K.

- Modes are sometimes useful
  – Long term (mode = program)
    • choosing an appropriate conceptual model or metaphor
  – Short term — allows shorter commands
- Modes can be ok if:
  – preemption is minimal
  – easy exit
    • Mode in restricted context (e.g., window)
    • spring-loaded modes
  – Clear visual indication of mode
    • Example: cursor shape

Principle of Least Astonishment

- Consistency is difficult to design, but you know you have achieved it when users make the right guesses
  – Rules should be few and general
  – Use clues from non-computer context when appropriate (metaphor)
Recommended reading

• D. Norman, *The Design of Everyday Things*
• N. Borenstein, *Programming as if Users Mattered*