CPT208 Human-Centric Computing

Design Principles and Heuristics

Lingyun Yu

Introduction: Design Principles

- Design principles are **fundamental pieces of advice for you to make easy-to-use, pleasurable designs**.
- You apply them when you select, create and organize elements and features in your work.
- Design principles represent the accumulated wisdom of researchers in design and related fields.
- When you apply them, you can predict how users will likely react to your design.

Introduction: Design Principles

- general guidelines for developing intuitive and successful interfaces
- qualitative character
- platform-independent
  - OS-independent
  - hardware-independent

Introduction: Design Principles

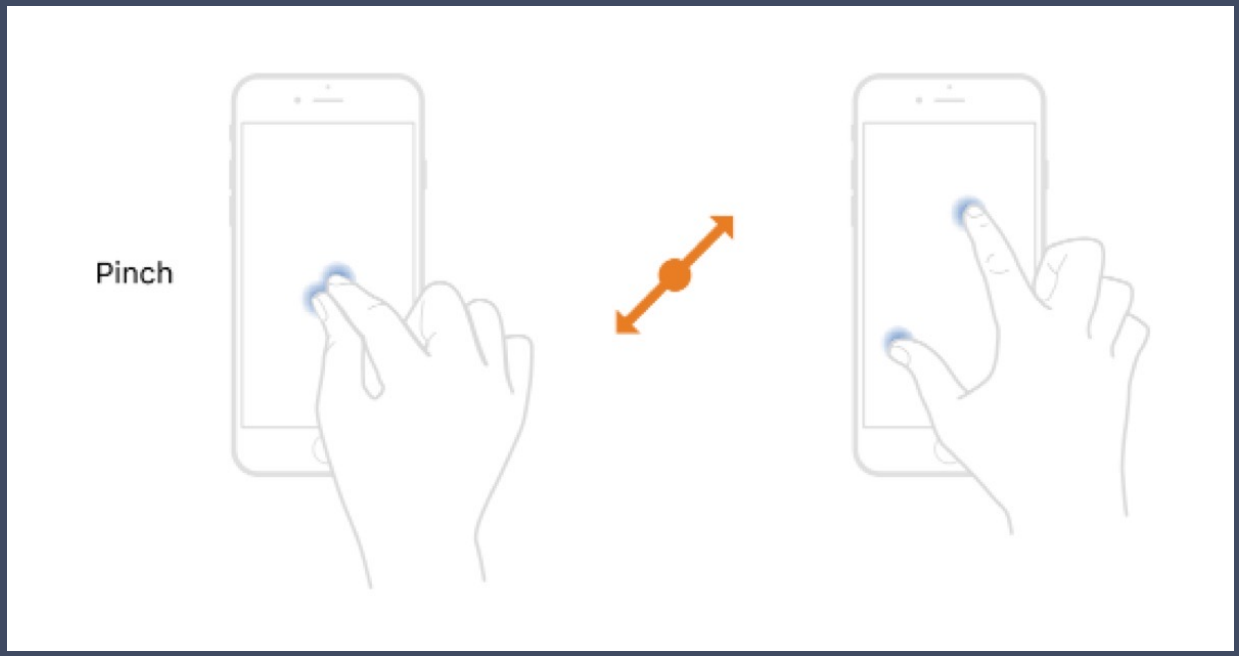
- Can be used to
  - ✓ **Design** the interface according to the “rules”
  - ✓ **Evaluate** the interface in the heuristic evaluation
  - ✓ **Communicate** with the developers

Design principles

- Simple and Natural Dialog, Let the User Develop a Mental Model
- Make System Modes Easy to Recognize and Distinguish
- Structure the Interface
- Provide Shortcuts, Make the Interface Adaptable
- Make Possible Actions Easily Visible
- Be Consistent
- Speak the User's Language
- Minimize the User's Memory Load
- Provide Feedback
- Do not Surprise the User
- Provide Clearly Marked Exits and Undo Functionality
- Deal with Errors in a Positive Manner
- Provide Help and Documentation

1) Simple and Natural Dialog, Mental Model

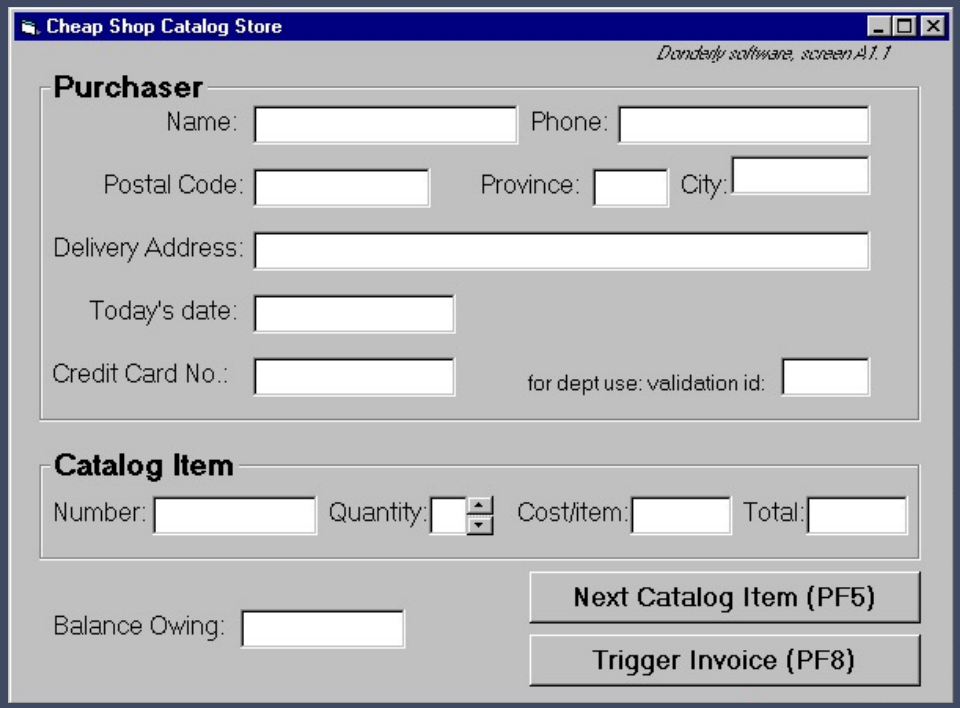
- use existing mental models



moving them closer together or further apart to change the scale factor, zoom, or level of detail of the user interface

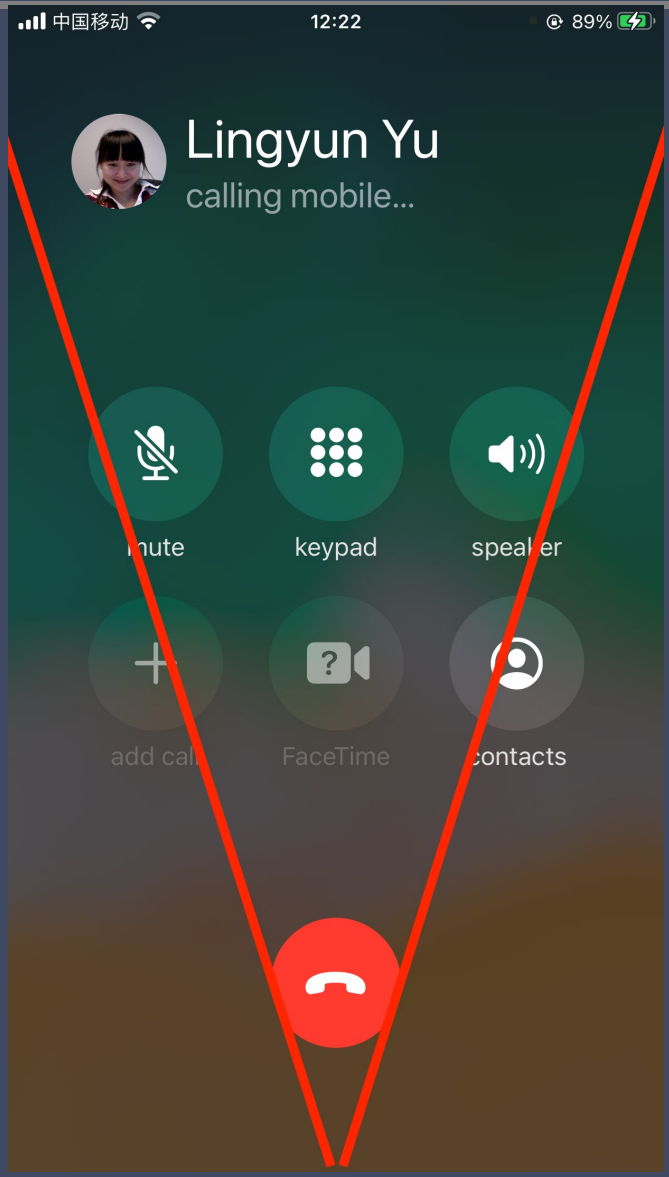
1) Simple and Natural Dialog, Mental Model

- present exactly the information the user needs
  - less is more (less to learn, to get wrong, to distract, …)
  - information should appear in a natural order (cluster related information, order to match user's expectations)



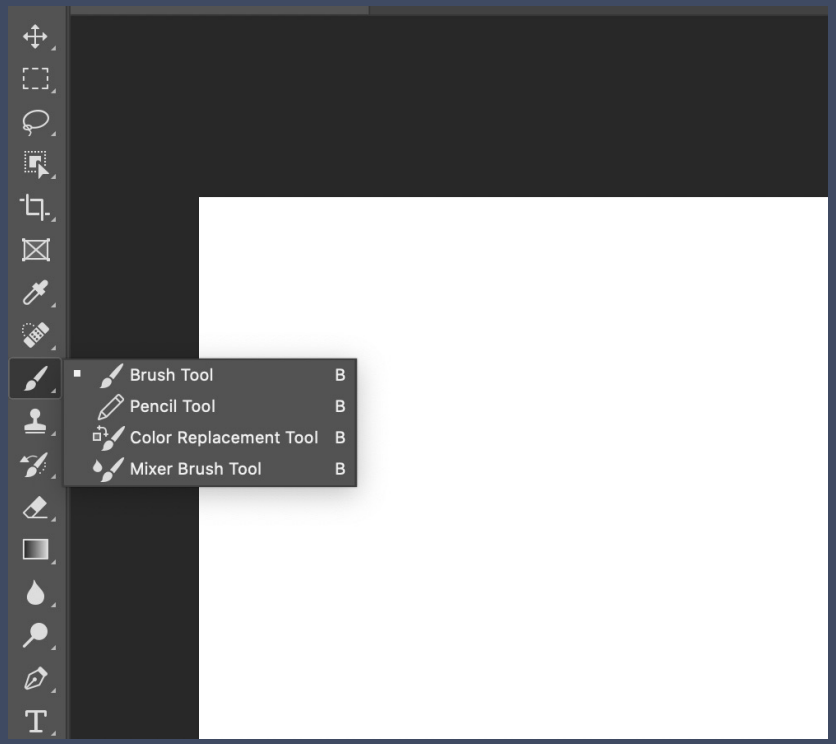
1) Simple and Natural Dialog, Mental Model

- present exactly the information the user needs
  - remove or hide irrelevant or rarely needed information (competes with important information on the screen)



1) Simple and Natural Dialog, Mental Model

- present exactly the information the user needs
  - remove (system) modes (e.g., edit mode vs. view mode)

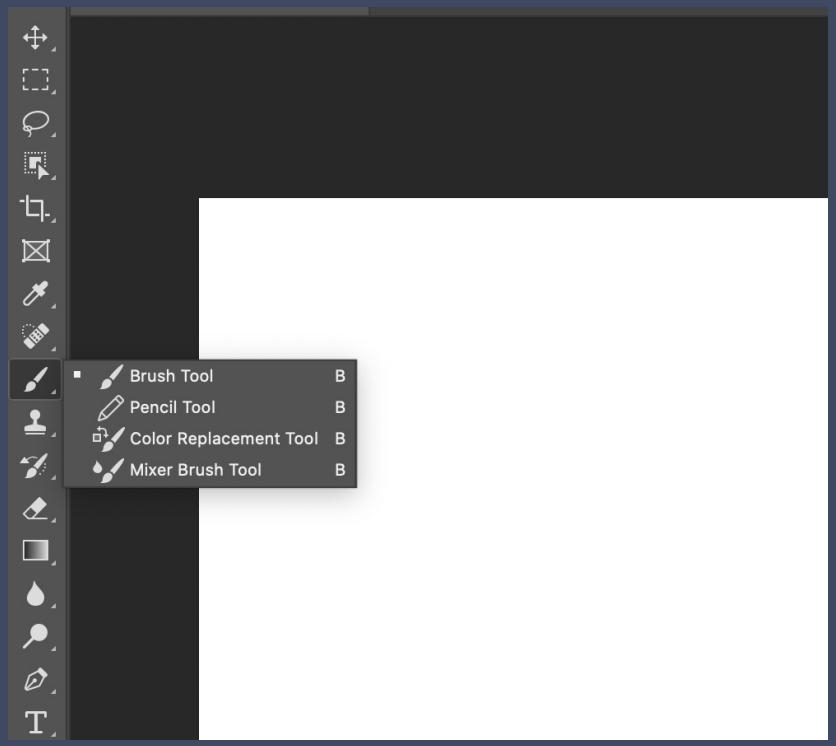


2) Modes Easy to Recognize & Distinguish

- modes: availability and meaning of commands (shortcuts, button clicks, etc.) depends on a mode
- *system-controlled* modes:
  - the system initiates the mode or maintains the mode
  - examples: select from the menu
- *user-controlled* modes:
  - the user actively initiates and maintains the mode
  - example: two-handed interaction on large displays
- **avoid** *system-controlled* modes if possible

2) Modes Easy to Recognize & Distinguish

- *system-controlled* modes VS *user-controlled* modes



2) Modes Easy to Recognize & Distinguish

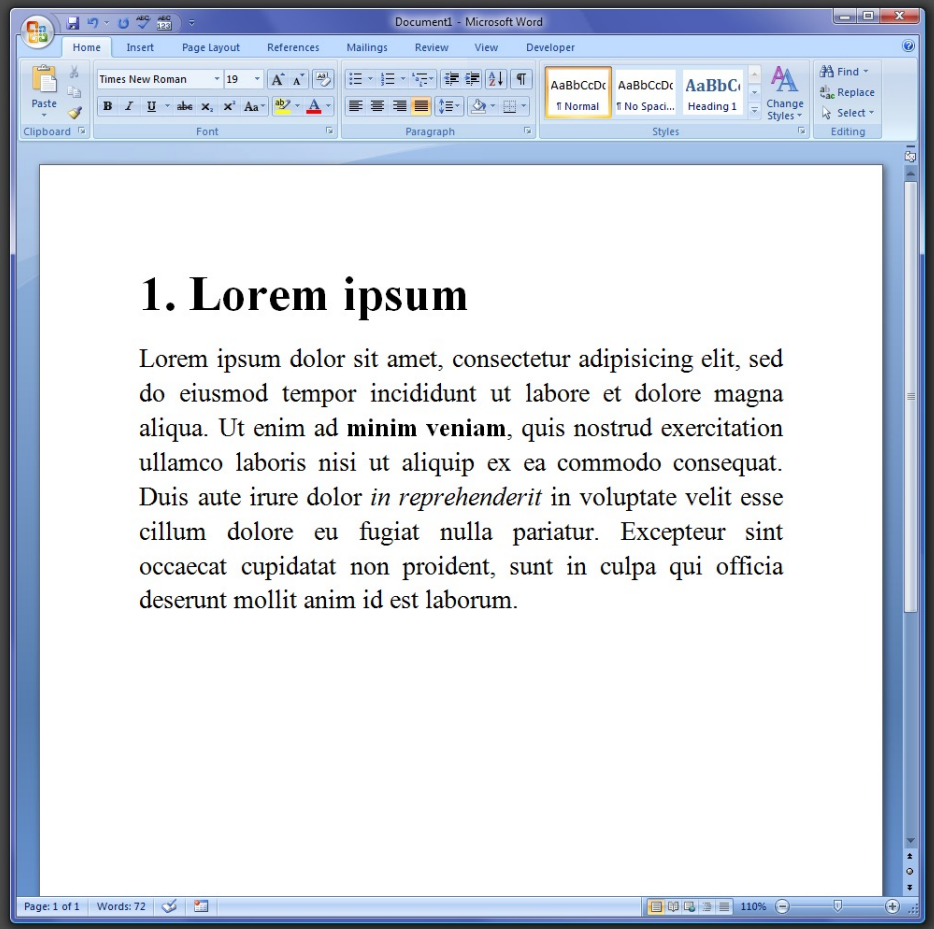
- bad example:
- better interface?
  - pen for writing
  - hold & write

2) Modes Easy to Recognize & Distinguish

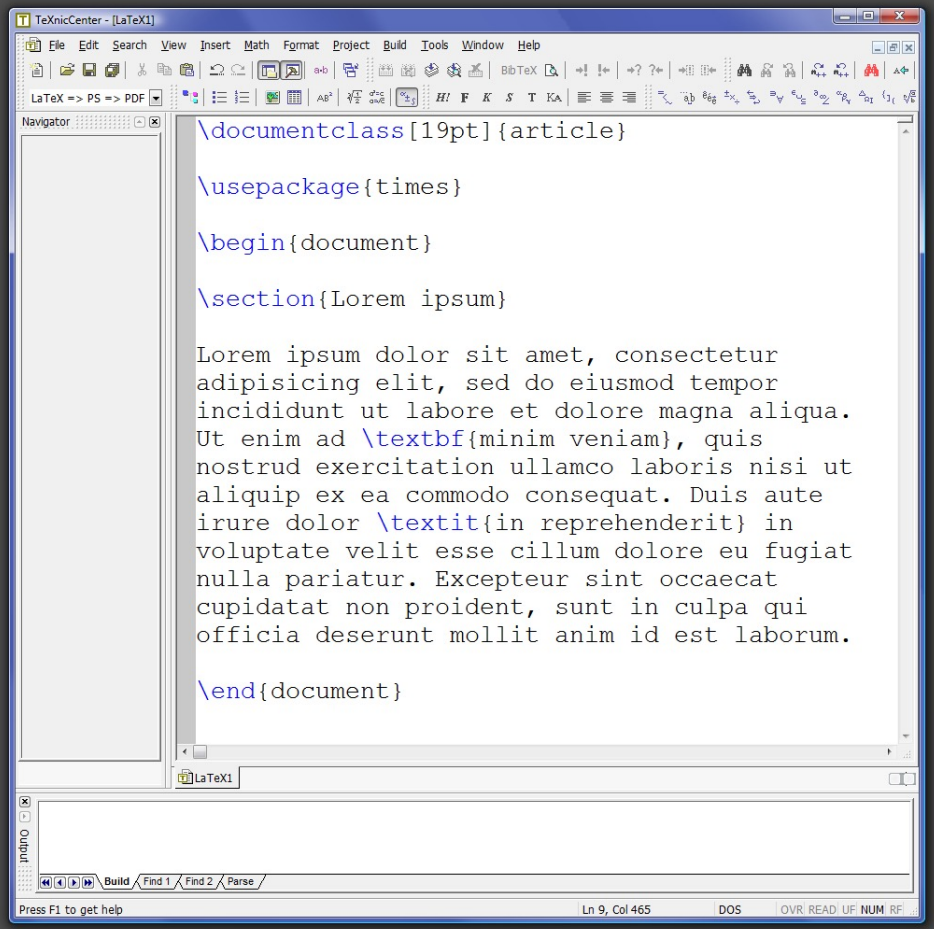
- much functionality to support, thus sometimes modes are necessary
- then need to make the modes **recognizable** and **distinguishable**

2) Modes Easy to Recognize & Distinguish

- modes **recognizable** and **distinguishable**



WYSIWYG



LaTeX

Example: 3D object manipulation

Output:

Translation along x, y, z

Rotation around x, y, z

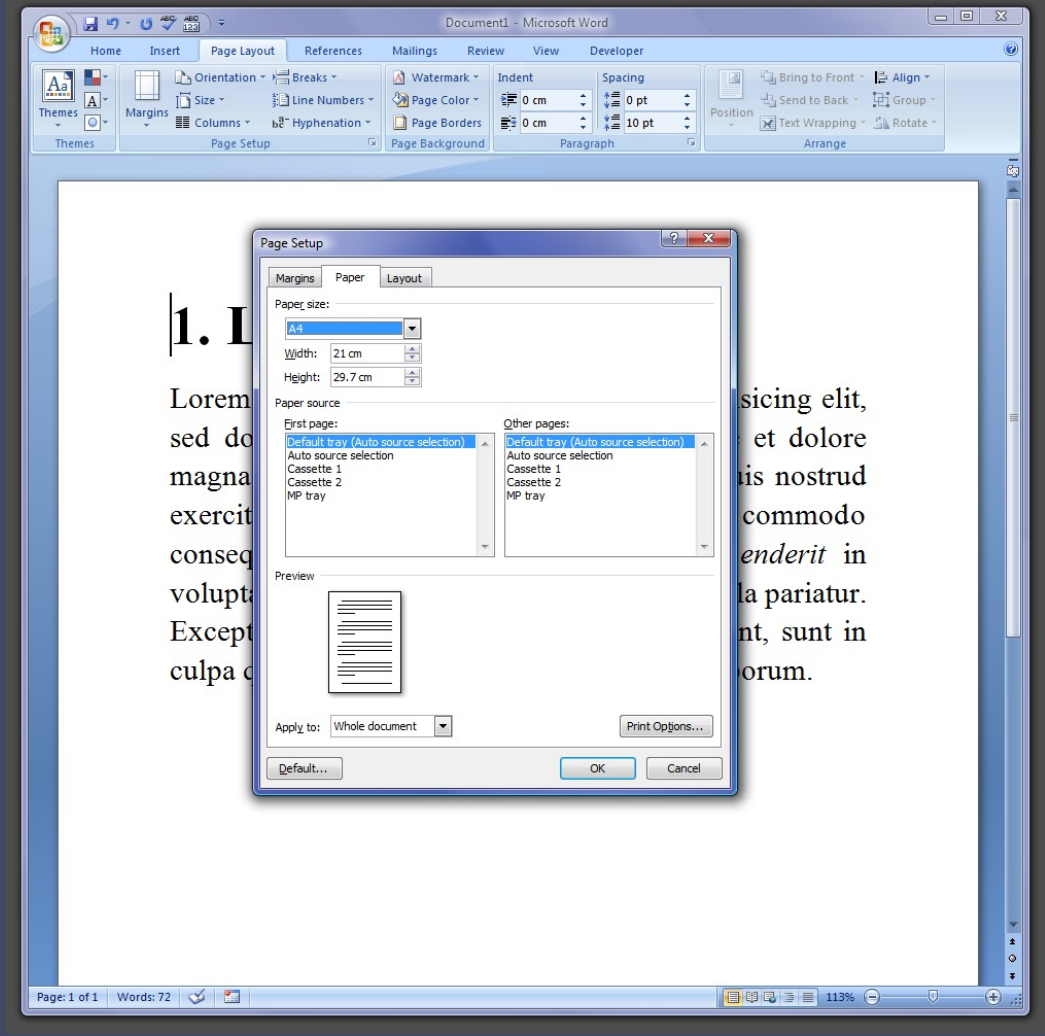
Scaling

Input:

Two touches

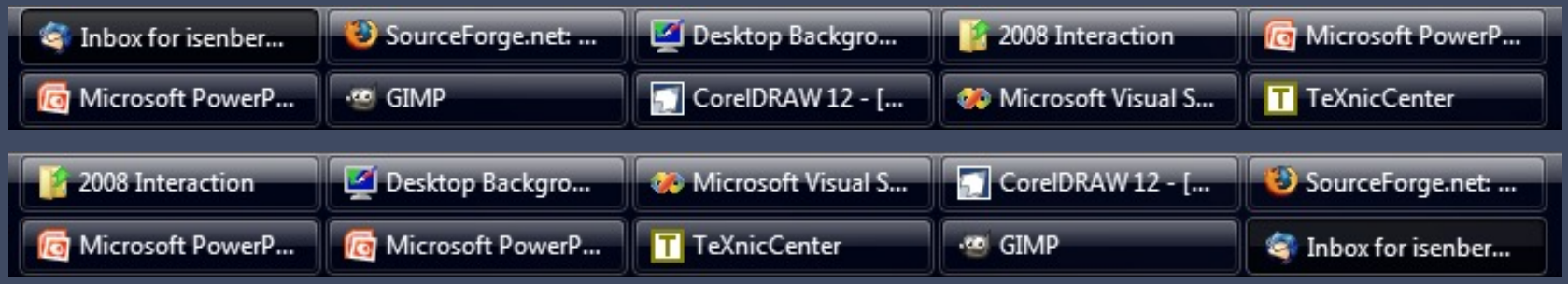
3) Structure the Interface

- group similar functionality
- structure the interface
- supports user's mental model
- new (and better) structure may initially perform worse → why?



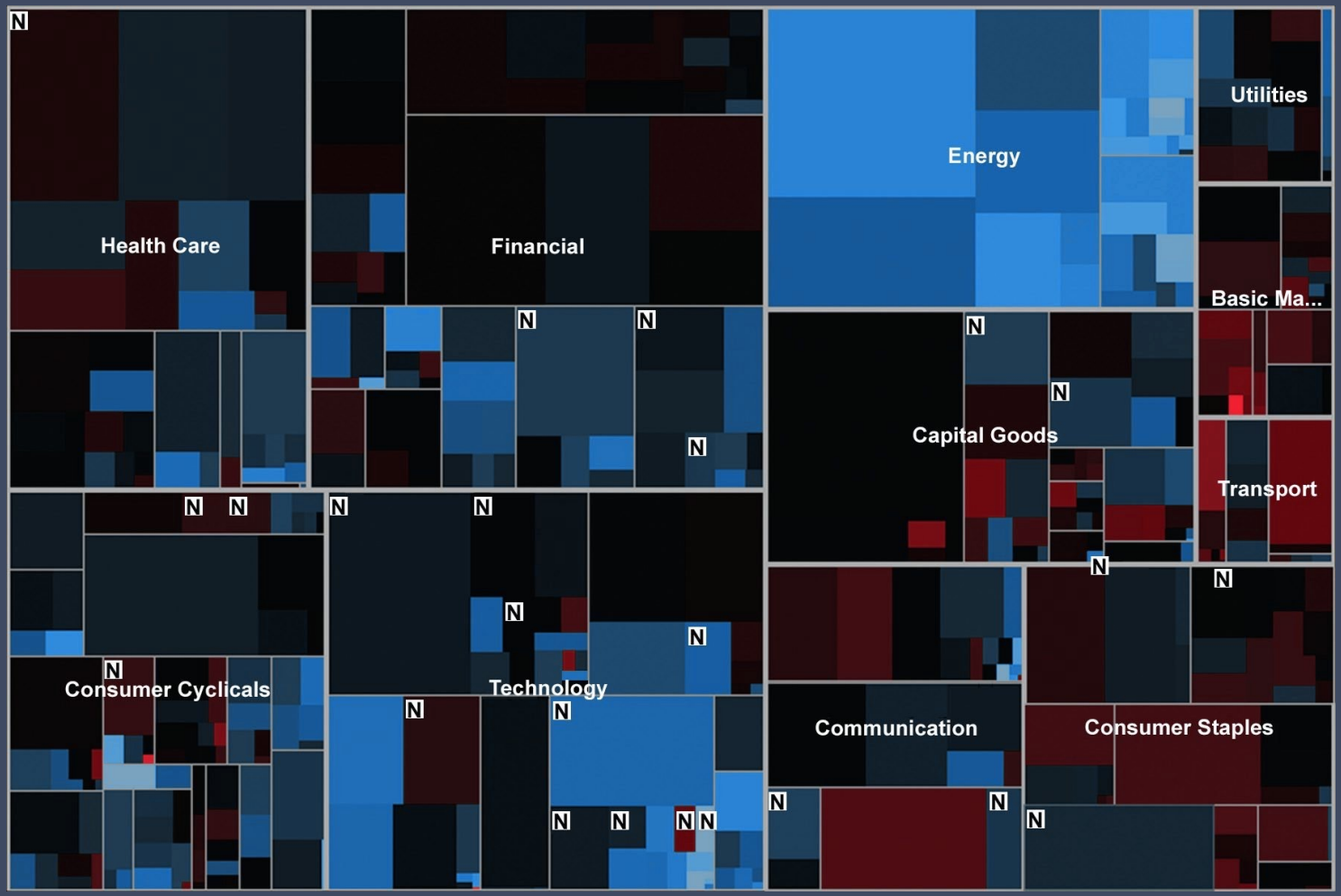
4) Adaptable Interface and Shortcuts

- people build the interface based on their own mental models



- classes of users: novices, occasional users, experts
  → different expectations of the interface
  → respect, respect, respect!
- different cognitive/visual abilities of users
  → different font sizes for visually impaired users
  → color schemes for users with color deficiency

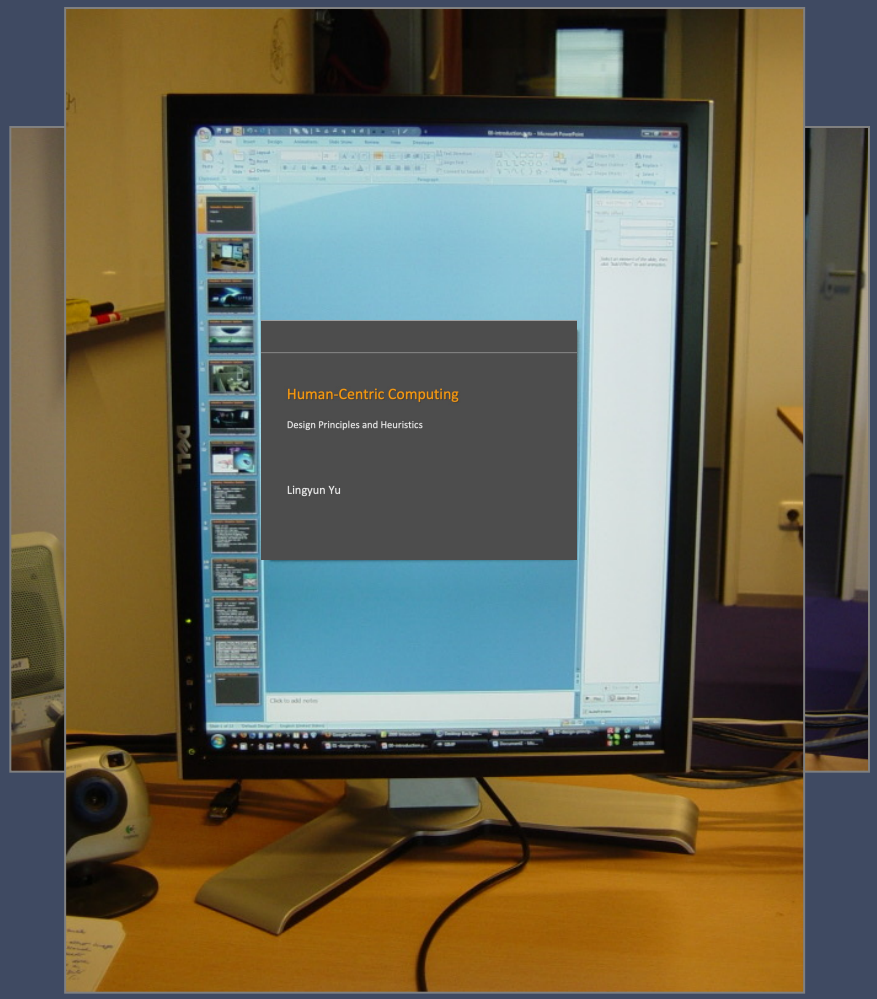
4) Adaptable Interface and Shortcuts

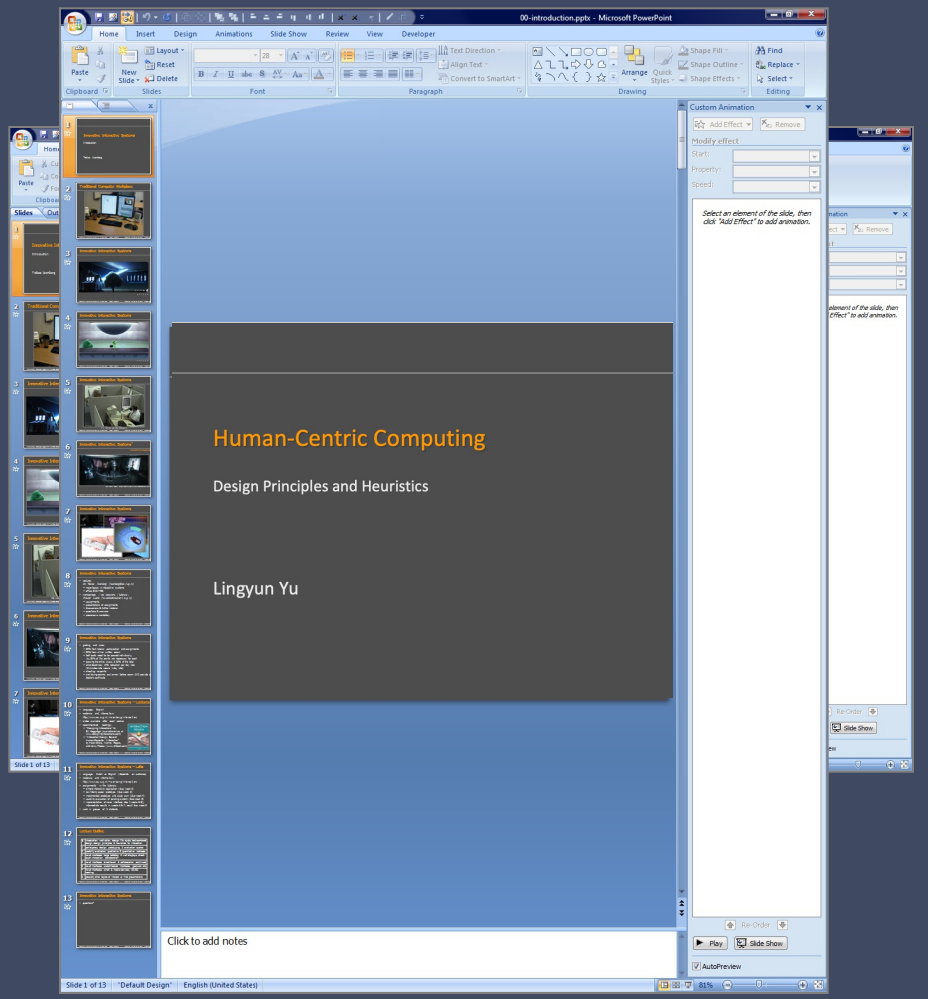


- provide options for color-deficient users

4) Adaptable Interface and Shortcuts

- different environments and specific hardware





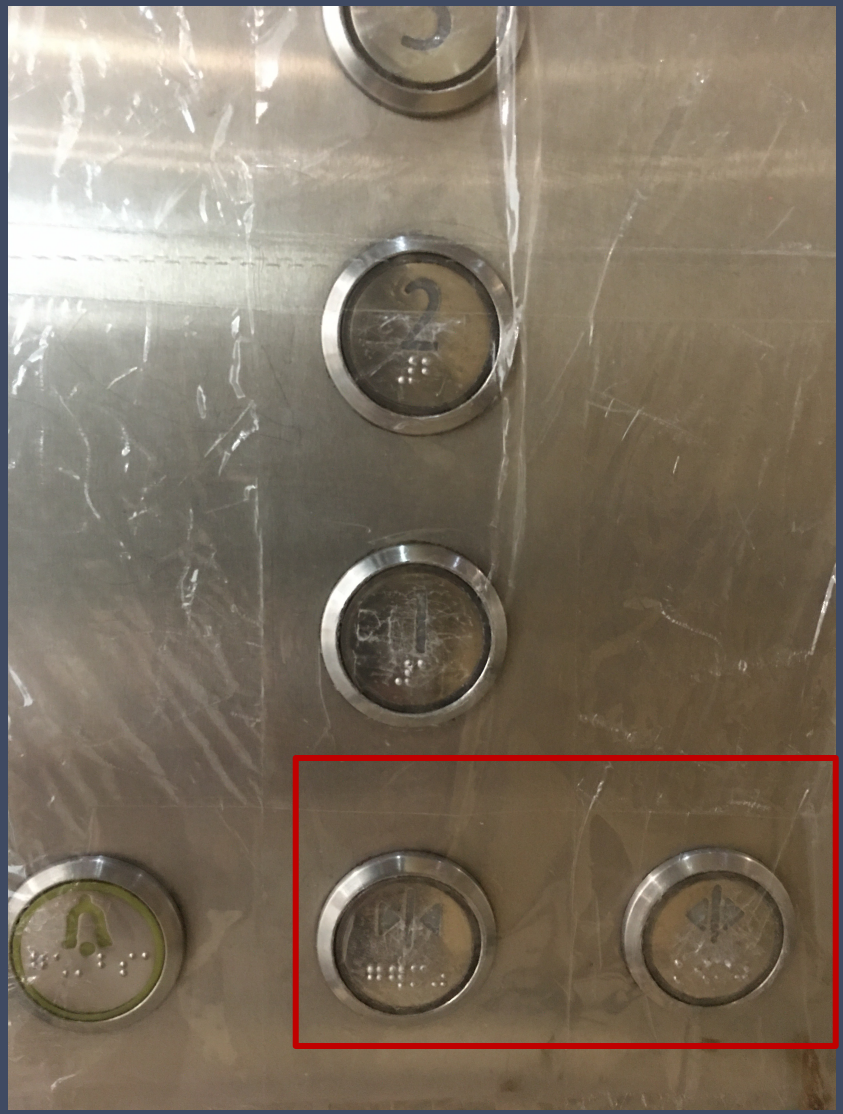
4) Adaptable Interface and Shortcuts

- different environments and specific hardware

4) Adaptable Interface and Shortcuts

- ***shortcuts***: expert users & frequent operations
  - mouse and keyboard accelerators
    - shortcut key combinations (Ctrl-S, Ctrl-Z, Ctrl-F4, …)
    - context menus (right-click to get specialized menu)
    - function keys (F1 for help, …)
    - double-clicking vs. single right click and menu selection
    - command completion (e.g., on a Unix or Win32 console: tab)

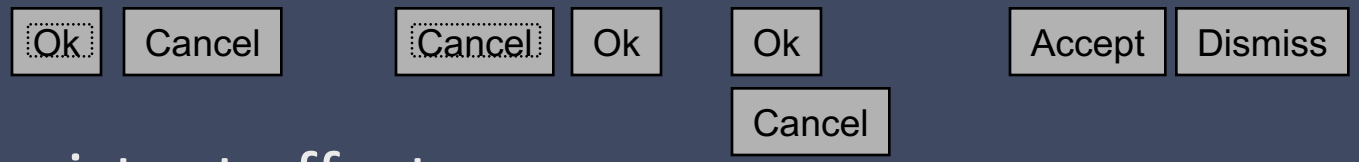
5) Be Consistent



5) Be Consistent

- consistent syntax of input
  - shortcuts

- consistent language and graphics
  - same visual appearance across the system
  - same information/controls in same location on all windows



- consistent effects
  - commands/actions have same effect in equivalent situations
    → predictability

5) Be Consistent

These are labels with a raised appearance.

Don't be surprised when people try to click on them …

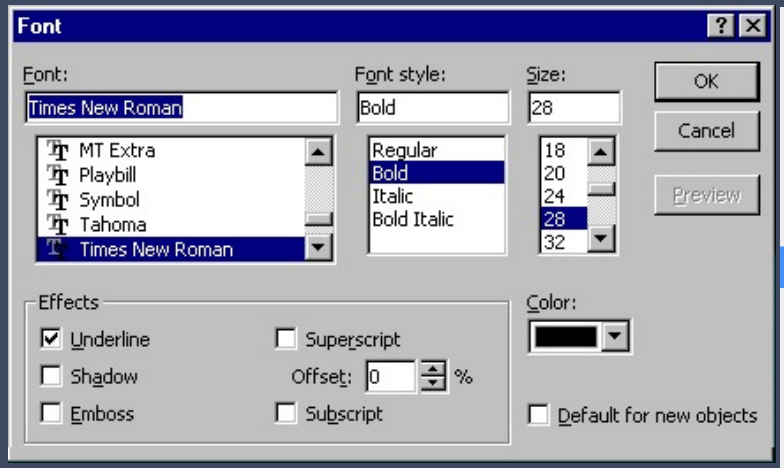
6) Speak the User's Language

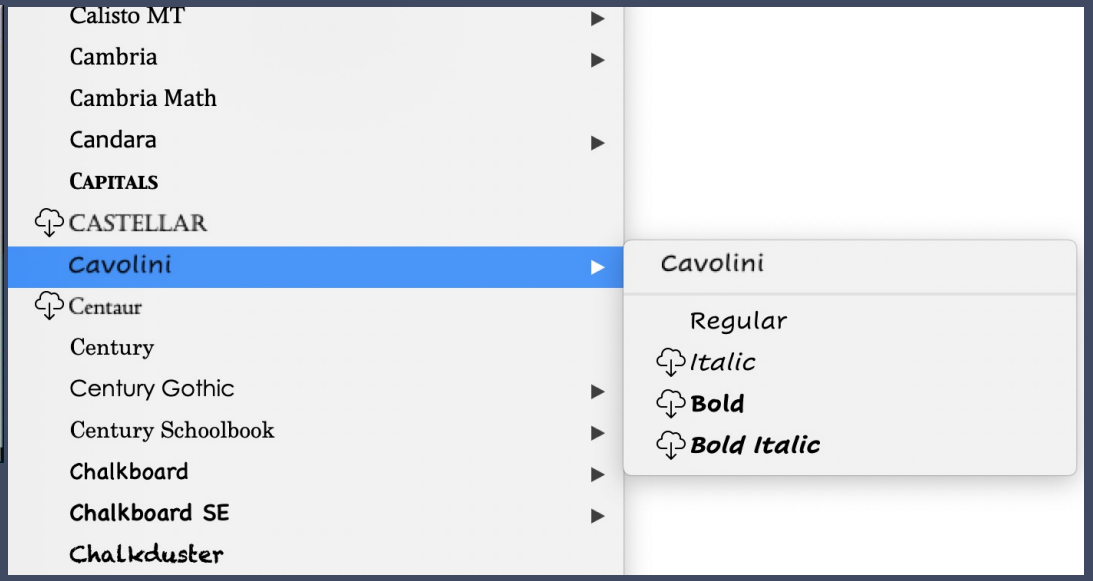
- use meaningful icons, abbreviations
  - Meaningful icons

  - Ctrl-S (abbreviation)

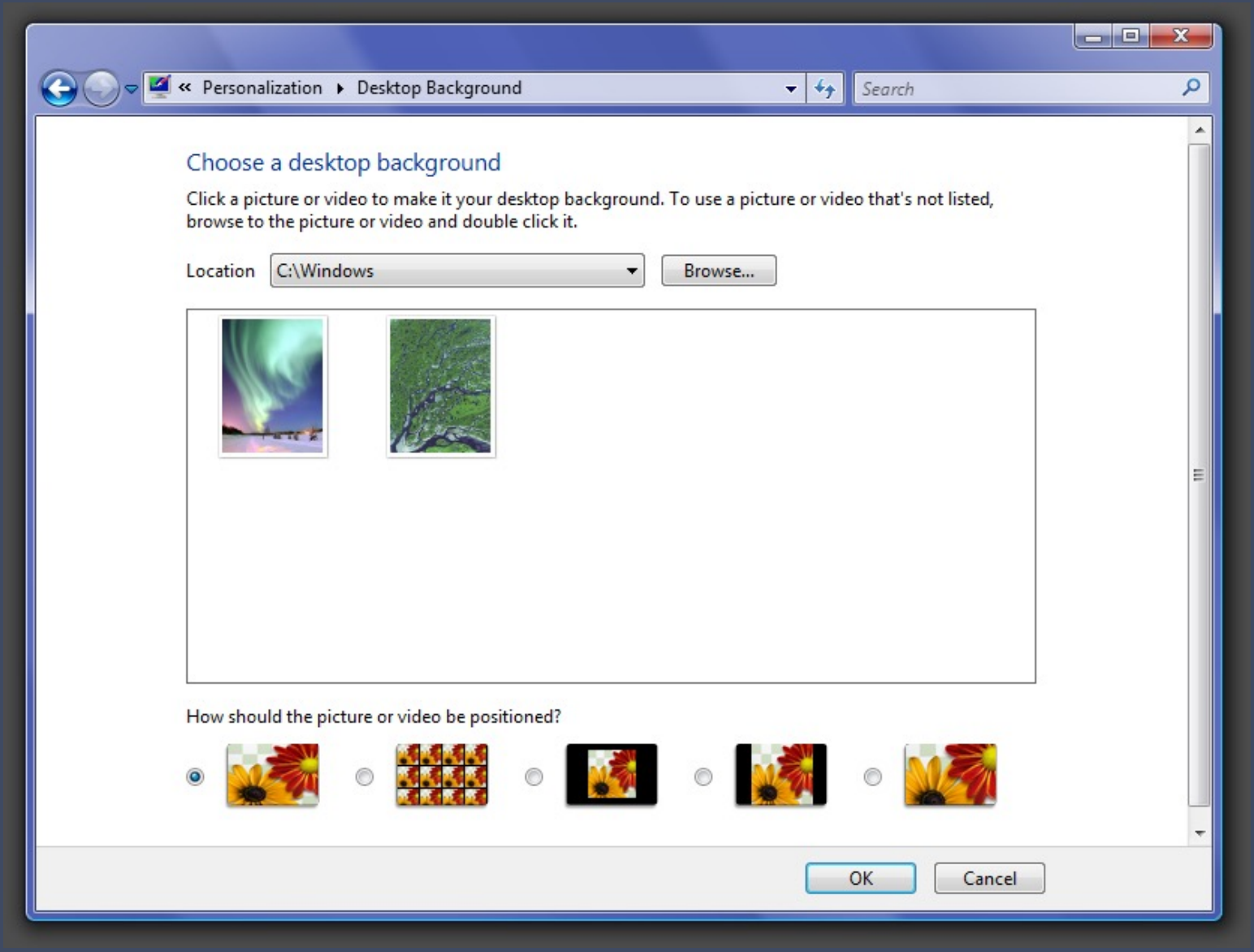
7) Minimize the User's Memory Load

- computers are good at remembering, people are not!
- promote recognition over recall
  - menus, icons, dialogs vs. commands, options, formats
  - relies on visibility of objects to the user (but less is more!)



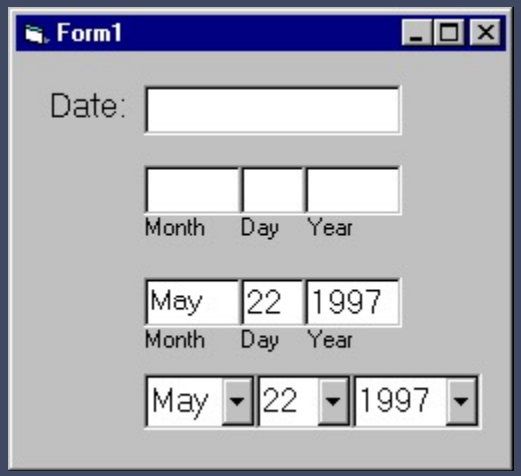


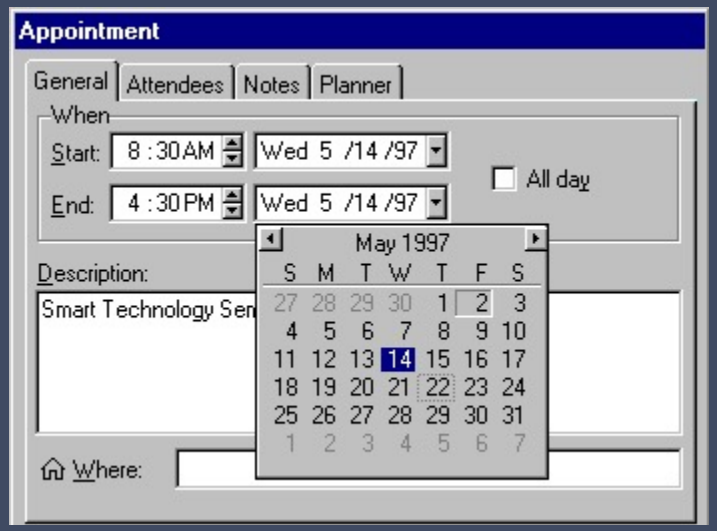
7) Minimize the User's Memory Load



7) Minimize the User's Memory Load

- give input format, example, and default



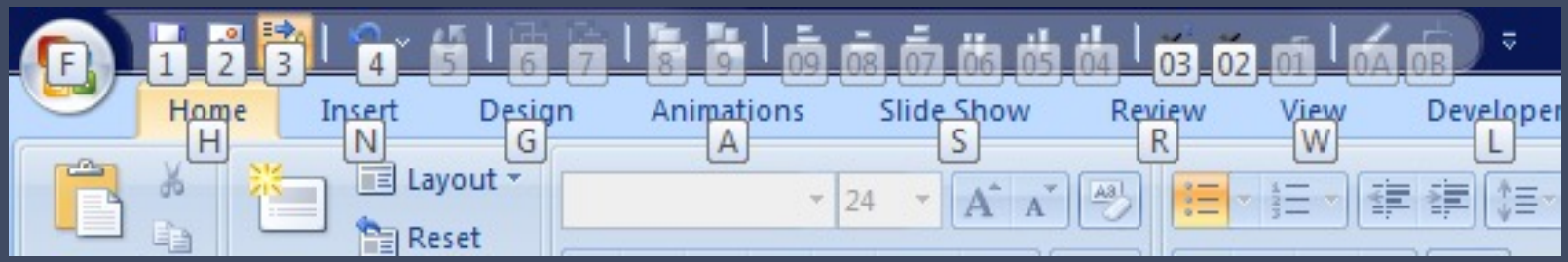


→ users do not need to remember format anymore, or even the day of the week

7) Minimize the User's Memory Load

- also show possible next steps/options
  or remind the user of shortcuts (help functionality)

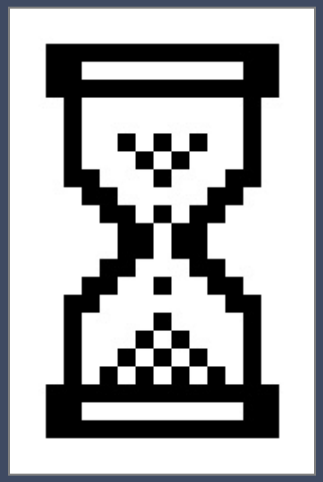
  ALT press:

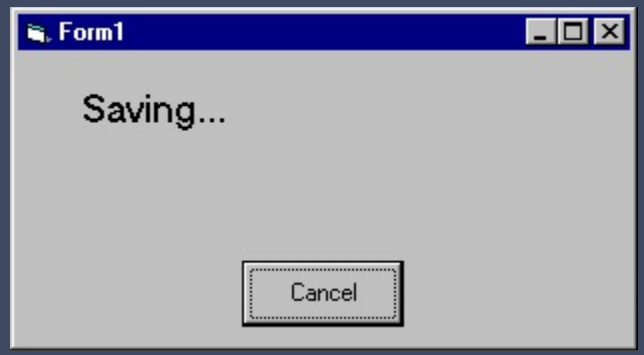


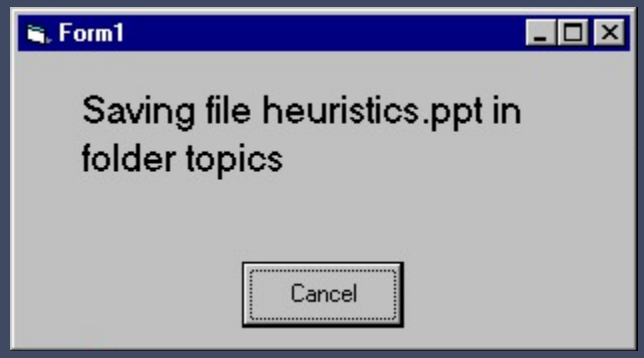
  mouse hover:

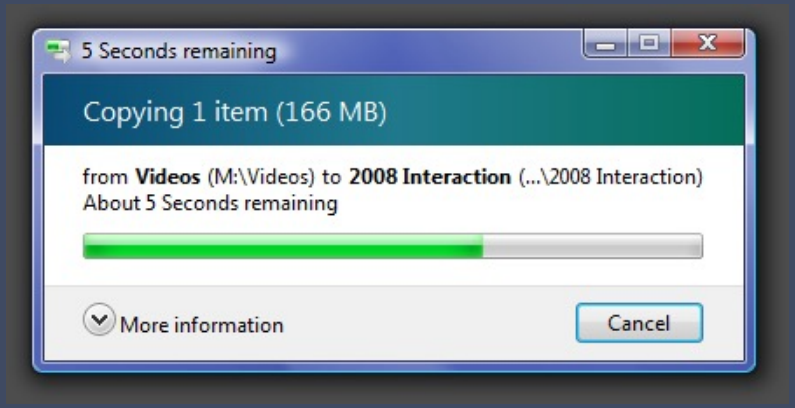
8) Provide Feedback

- user should always be aware of what is going on
- continuously inform the user about
  - what the program/interface is doing
  - how it is interpreting the user's input









8) Provide Feedback

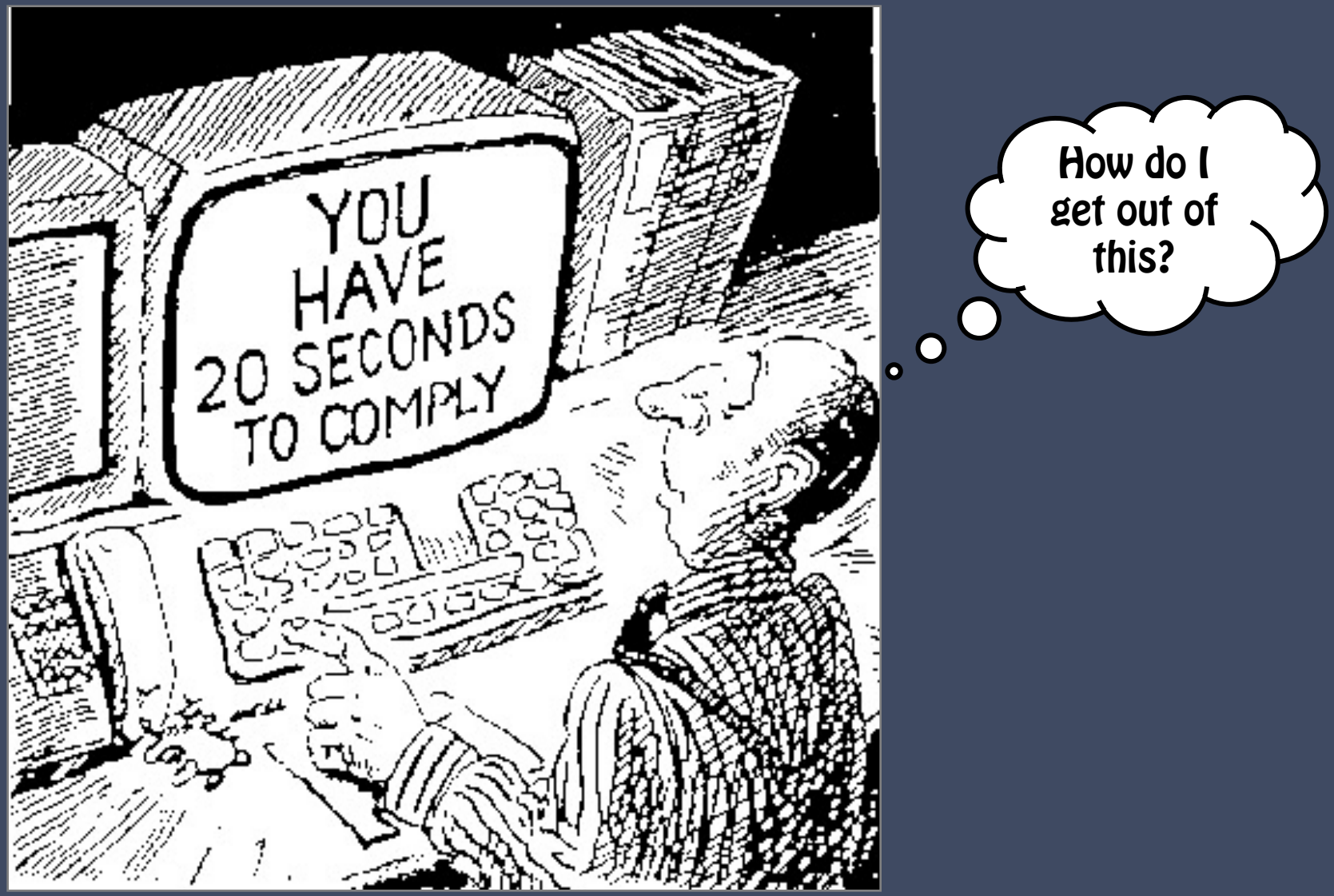
- response times: how users perceive delays

< 0.1 s	perceived as “instantaneous”
1 s	user’s flow of though stays uninterrupted, but the delay is noticed
10 s	limit for keeping user’s attention focused on the dialog
> 10 s	user will want to perform other tasks while waiting

9) Do not Surprise the User

- programs should behave in a predictable way (mental model)
- results of actions should be foreseeable
- e.g., provide automatic or manual previews

9) Provide Clearly Marked Exits



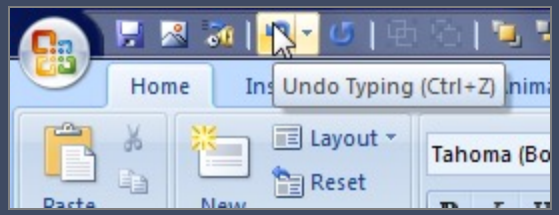
9) Provide Clearly Marked Exits

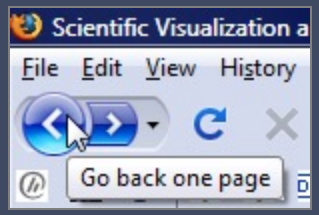
- users do not like to feel trapped by the computer!
  - interfaces should offer an easy way out of as many situations as possible

- strategies:
  - cancel button (for dialogs waiting for user input)
  - pause/continue (especially for lengthy operations)
  - quit (for leaving the program at any time; ≠ cancel)

10) Provide Undo (and Redo) Functionality

- interactive systems: designed to support a creative process in which users try out things: trial-and-error
- actions need to be reversible:



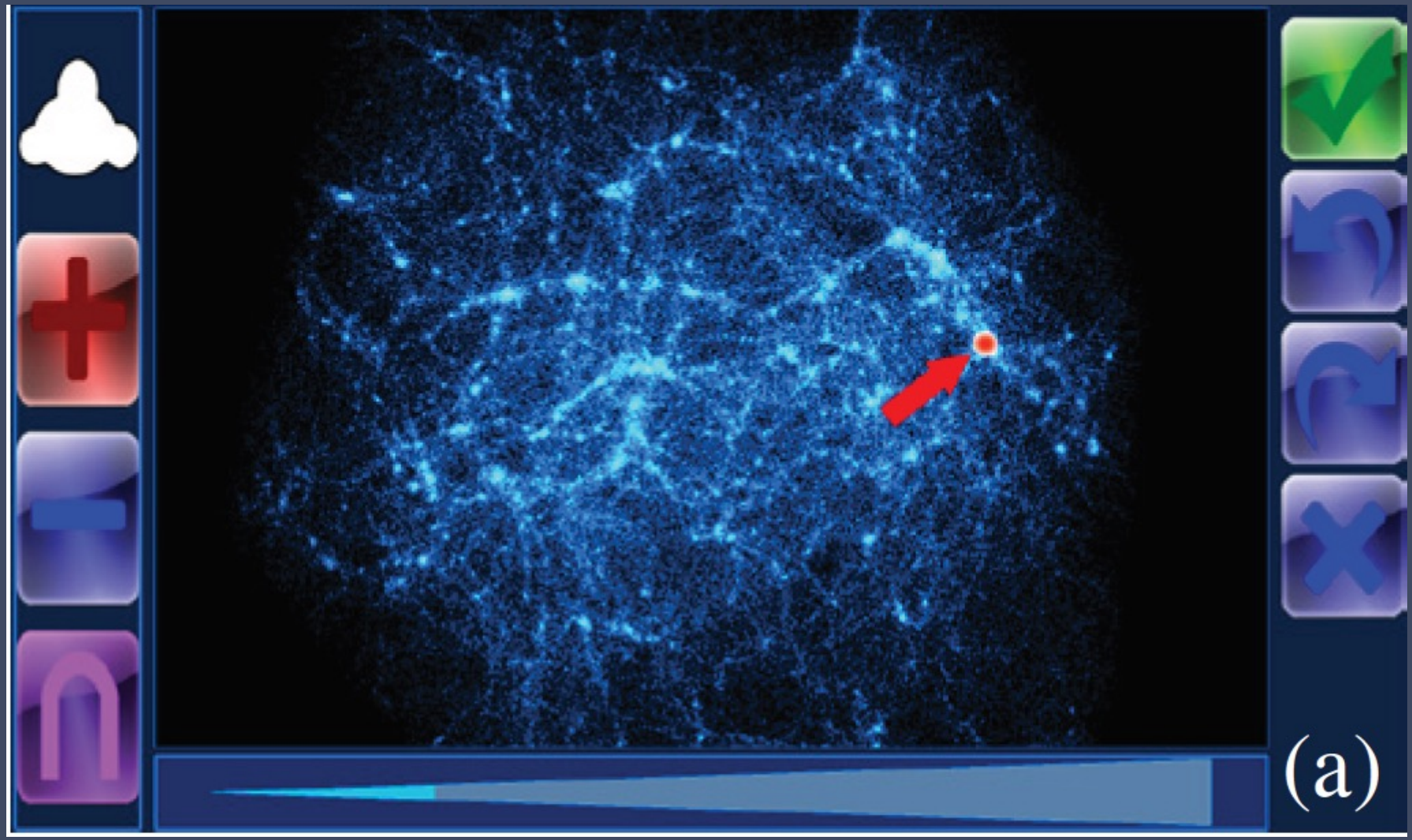


Web browsing is also often a trial-and-error interaction

- undo functionality needs to be easily accessible:
  - most Windows programs (consistency): Ctrl-Z
  - accessibility depends on major interaction technique: e.g., Web browsing uses the mouse, so back button and Ctrl-Z are not ideal

10) Provide Undo (and Redo) Functionality

- several steps of undo (and redo) should be possible
- undo should use logical and predictable steps
- not all actions need to be undone (scrolling?)
- not always clear what “an action” is:
  - actions initiated by clicks or keyboard commands
  - typing (letter? word? paragraph? typed text between two longer typing pauses?)
- visualizations of what could be undone possible (interaction history)
- if undo impossible: warn user!


(a)

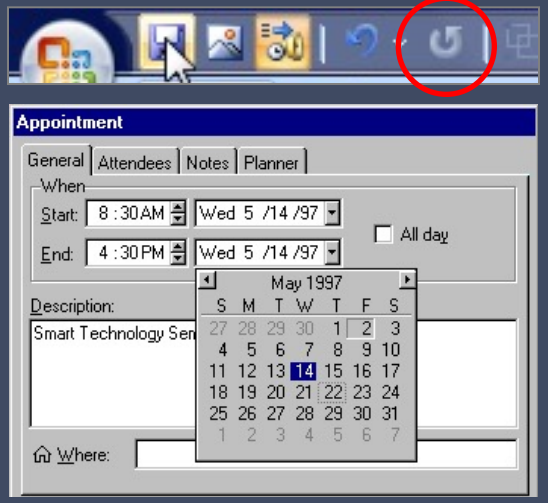
11) Deal with Errors in a Positive Manner

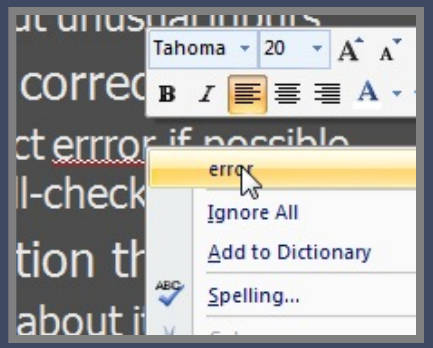
- people will make errors!

- types of errors:
  - mistakes:
    conscious deliberations lead to an error
    instead of the correct solution (see undo-redo)
  - slips:
    unconscious behavior that gets misdirected route to a satisfying goal, e.g., drive to store, end up in the office
    - shows up frequently in skilled behavior,
      usually due to inattention
    - often arises from similar actions

11) Deal with Errors in a Positive Manner

- designing for slips – general rules:
  - prevent slips before they occur
    - do not allow illegal actions or inputs
    - use masks and selection lists for data input
    - have as few different modes as possible
    - warn about unusual inputs
  - detect and correct slips when they do occur
    - self-correct error if possible (e.g., spell-check while typing)
  - user correction through feedback and undo
    - "let's talk about it": system initiates dialog to solve problem (e.g., compiler error)





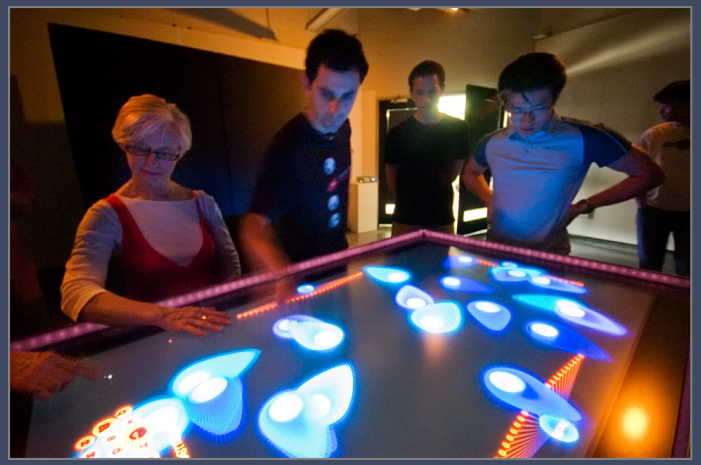
12) Deal with Errors Positively: Provide Help

- provide meaningful error messages
- error messages in the user's task language
- don't make people feel stupid
  - Error.
  - Error 25.
  - Try again!
  - Cannot open this document.
  - Cannot open "00-introduction.pptx" because the application "Microsoft PowerPoint" is not on your system.
  - Cannot open "00-introduction.pptx" because the application "Microsoft PowerPoint" is not on your system. Open with "OpenOffice" instead?

13) Provide Help

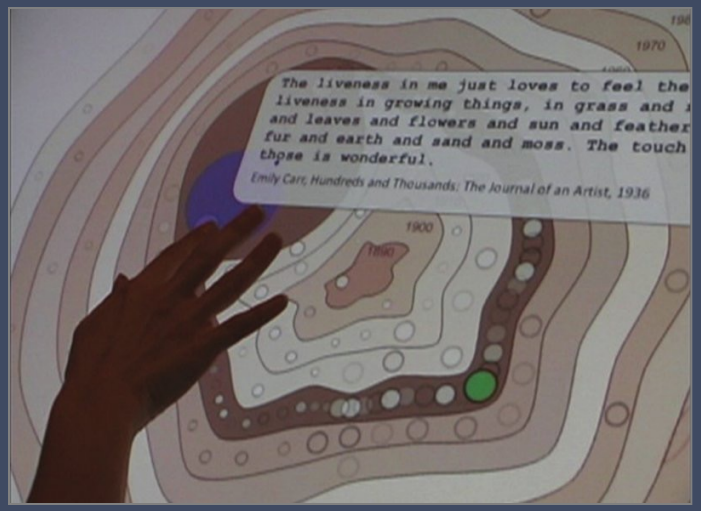
- help is not a replacement for bad design!

- simple systems:
  - walk-up-and-use
  - with minimal instructions



Hinrichs et al., 2007

- most other systems:
  - feature rich
  - simple things should be simple
  - learning path for advanced features



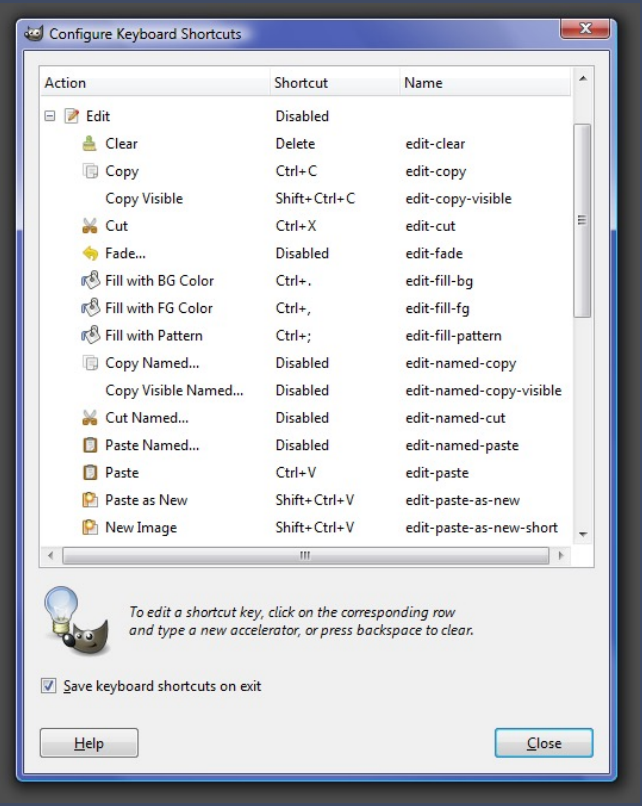
Hinrichs et al., 2008

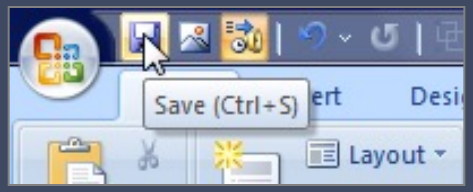
13) Provide Help: Documentation

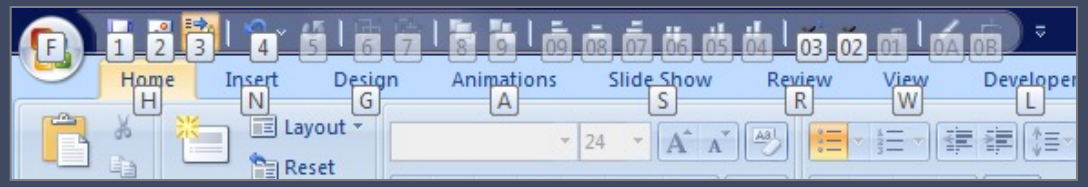
- many users do not read manuals
  - want to spend time on task rather than reading about it
- usually used when users are in some kind of panic
  - paper manuals often unavailable (locked/thrown away)
  - online documentation better
  - good search and lookup tools
  - online help specific to current context
- sometimes used for quick reference
  - syntax of actions, possibilities, list of shortcuts, etc.

13) Provide Help: Types of Help

- tutorials and/or getting started manuals
- reference manuals for lookup
  - large ones for detailed reference
  - short ones for quick lookup
- reminders, tooltips, context-sensitive help (e.g., syntax of function calls in programming)

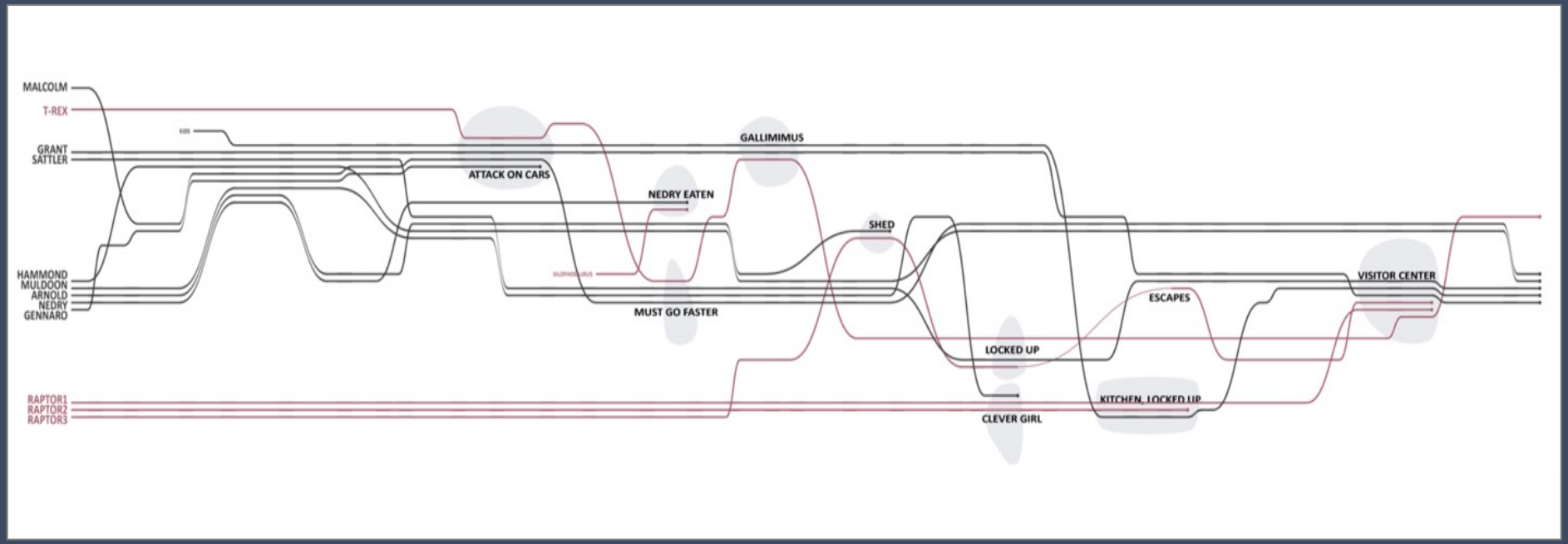






Customized Design Principle

- iStoryline
  - **D1** Lines in the same group should appear next to each other.
  - **D2** Otherwise, lines must be far away from each other.
  - **D3** A line must remain straight unless its group changes

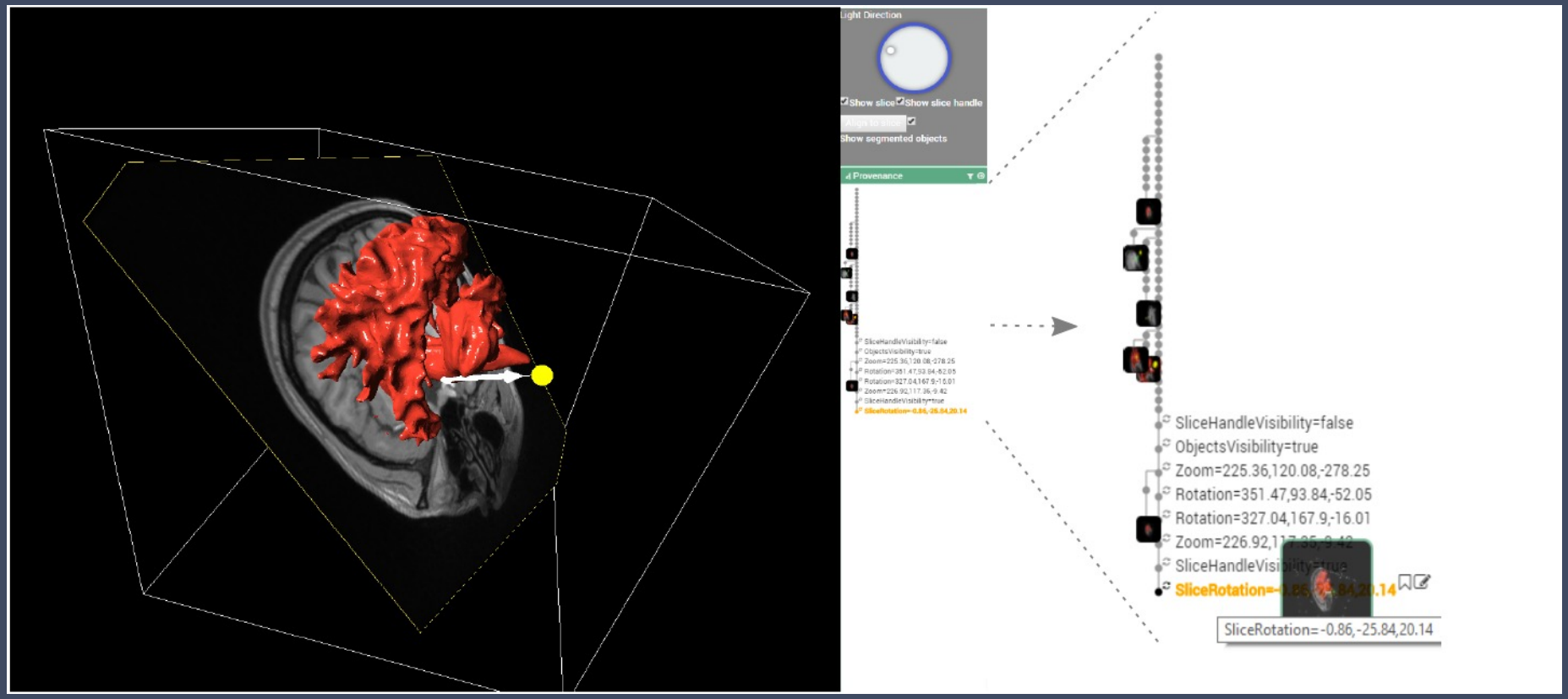


Summary

- several (heuristic) principles of design
  - Simple and Natural Dialog, Let the User Develop a Mental Model
  - Make System Modes Easy to Recognize and Distinguish
  - Structure the Interface
  - Provide Shortcuts, Make the Interface Adaptable
  - Make Possible Actions Easily Visible
  - Be Consistent
  - Speak the User's Language
  - Minimize the User's Memory Load
  - Provide Feedback
  - Do not Surprise the User
  - Provide Clearly Marked Exits and Undo Functionality
  - Deal with Errors in a Positive Manner
  - Provide Help and Documentation

In class activity

List Five Design Principles that we can use to improve the usability of the system.



References

- **Improving a Human-Computer Dialogue**, Nielsen and Molich, March 1990, *Communications of the ACM* 33(3), ACM Press.
- **Chapter 5: Usability heuristics.** Nielsen, J. (1993) In *Usability Engineering,* p115-163, Academic Press.

An Interface Design Process