

# Design Principles and Usability Heuristics - Part II

## The 9 Heuristics

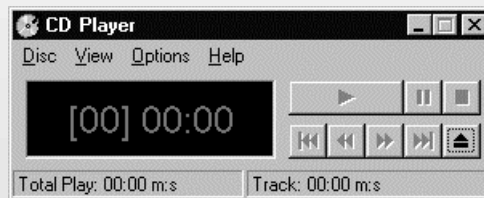
Ben Bederson / Saul Greenberg

### 1 Simple and natural dialogue

Use the user's conceptual model

Match the users' task in as natural a way as possible

- maximize mapping between interface and task semantics



Ben Bederson / Saul Greenberg

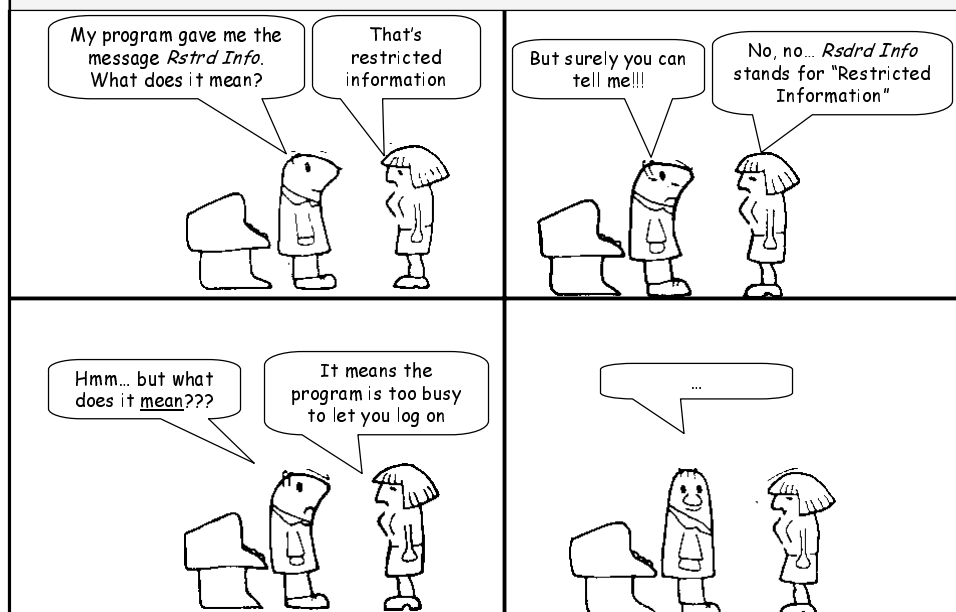
## 1 Simple and natural dialogue

### Present exactly the information the user needs

- less is more
  - less to learn, to get wrong, to distract...
- information should appear in natural order
  - related information is graphically clustered
  - order of accessing information matches user's expectations
- remove or hide irrelevant or rarely needed information
  - competes with important information on screen
- use windows frugally
  - don't make navigation and window management excessively complex

Ben Bederson / Saul Greenberg

## 2 Speak the users' language



## 2 Speak the users' language

Terminology based on users' language for task

- e.g. withdrawing money from a bank machine



Use meaningful mnemonics, icons, and abbreviations

- eg: File / Save
  - Ctrl + S (abbreviation)
  - Alt F S (mnemonic for menu action)
  - Open folder (tooltip icon)



Ben Bederson / Saul Greenberg

## 2 Speak the users' language

**Ex:** Consider a virus detection program that may have to be occasionally turned off.

**One option would be to have an “override mode” that when activated would turn off the virus detection.**

**But this would be *on* when the user wanted the utility to be *off* – conflicting with the users' model**

**Alternatively, a checkbox that was on when the utility would be on would speak the users' language.**

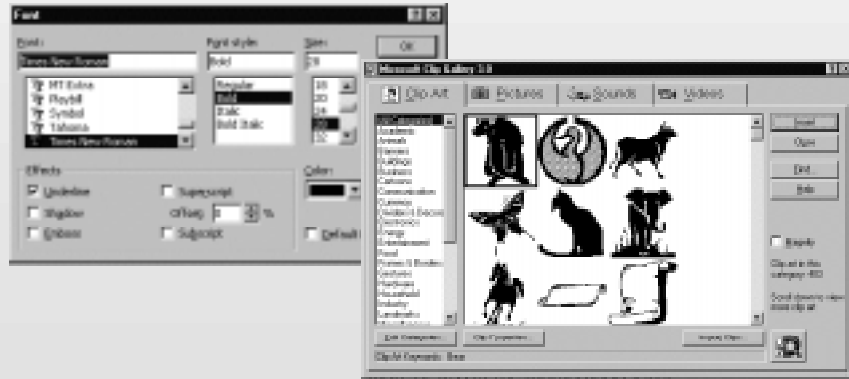
Ben Bederson / Saul Greenberg

### 3 Minimize user's memory load

Computers good at remembering things, people aren't!

Promote recognition over recall

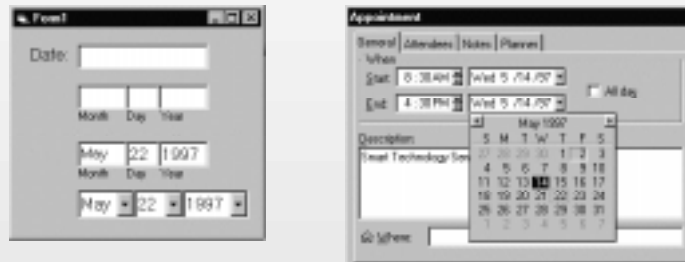
- menus, icons, choice dialog boxes vs command lines, field formats
- relies on visibility of objects to the user (but less is more!)



Ben Bederson / Saul Greenberg

### 3: Minimize user's memory load

Describe required input format and example, and default



Small number of rules applied universally

- generic commands
  - same command can be applied to all interface objects  
*interpreted in context of interface object*
  - copy, cut, paste, drag 'n drop, ... for characters, words, paragraphs, circles, files

Ben Bederson / Saul Greenberg

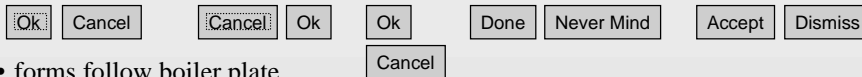
## **4: Be consistent**

### **Consistency of effects**

- same words, commands, actions will always have the same effect in equivalent situations
  - predictability

### **Consistency of language and graphics**

- same information/controls in same location on all screens / dialog boxes



- forms follow boiler plate
- same visual appearance across the system (e.g. widgets)
  - e.g. different scroll bars in a single window system!



### **Consistency of input**

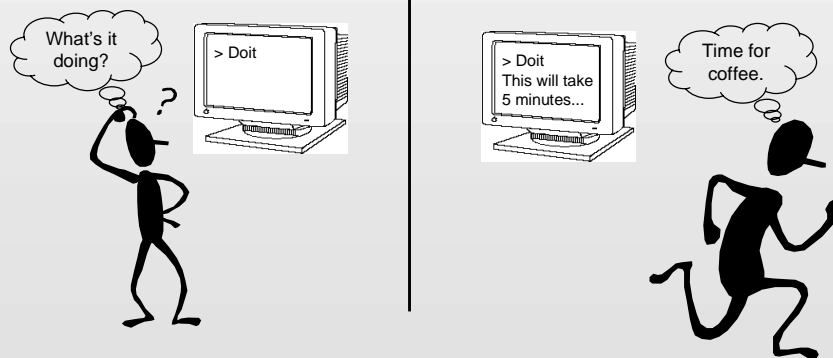
- consistent syntax across complete system

Ben Bederson / Saul Greenberg

## **5: Provide feedback**

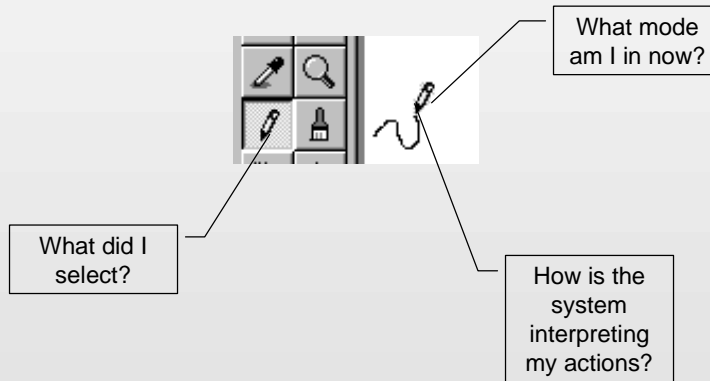
### **Continuously inform the user about**

- what it is doing
- how it is interpreting the user's input
- user should always be aware of what is going on



Ben Bederson / Saul Greenberg

## 5. Provide feedback



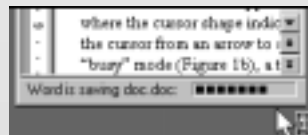
Ben Bederson / Saul Greenberg

## 5. Provide feedback

Should be as specific as possible, based on user's input



Best within the context of the action



Ben Bederson / Saul Greenberg

## **5. Provide feedback**

### **Response time**

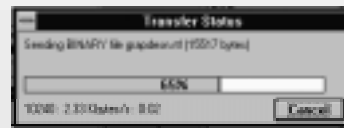
- how users perceive delays
  - 0.1 second max: perceived as “instantaneous”
  - 1 seconds max: user’s flow of thought stays uninterrupted, but delay noticed
  - 10 seconds: limit for keeping user’s attention focused on the dialog
  - > 10 seconds: user will want to perform other tasks while waiting

Ben Bederson / Saul Greenberg

## **5. Provide feedback**

### **Dealing with long delays**

- Cursors
  - for short transactions
- Percent done dialogs
  - for longer transactions
  - how much left
  - estimated time
  - what it is doing...
- Random
  - for unknown times



Ben Bederson / Saul Greenberg

## 6. Provide clearly marked exits



How do I get out of this?

Ben Bederson / Saul Greenberg

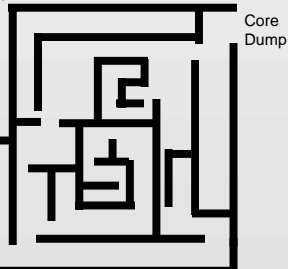
## 6. Provide clearly marked exits

**Users don't like to feel trapped by the computer!**

- should offer an easy way out of as many situations as possible

### **Strategies:**

- Cancel button (for dialogs waiting for user input)
- Universal Undo (can get back to previous state)
- Interrupt (especially for lengthy operations)
- Quit (for leaving the program at any time)
- Defaults (for restoring a property sheet)



Ben Bederson / Saul Greenberg

## 7. Provide shortcuts

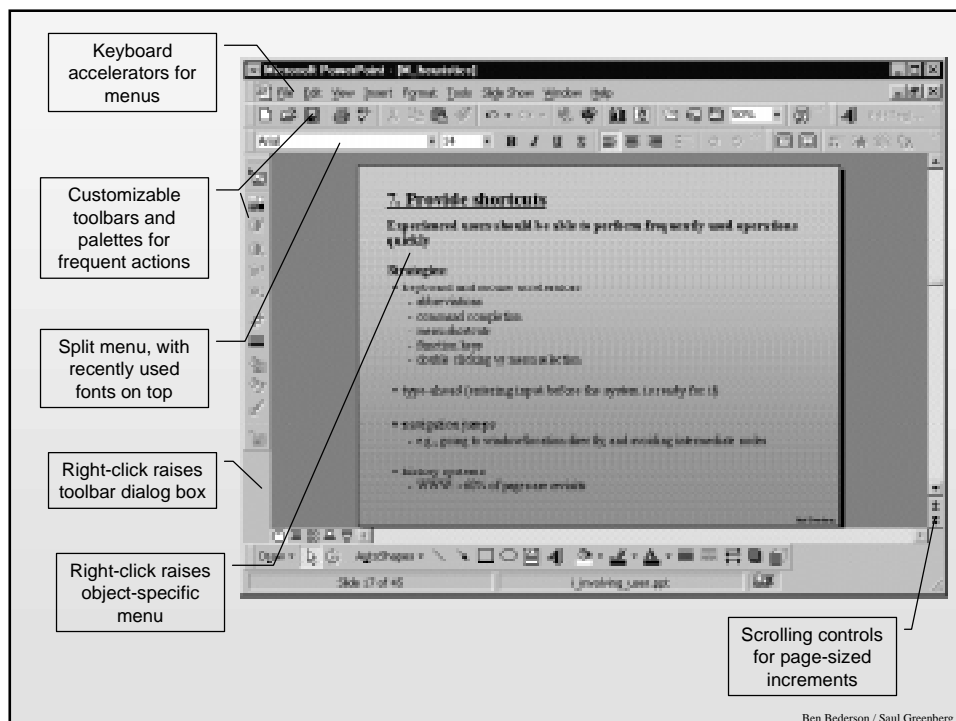
Experienced users should be able to perform frequently used operations quickly



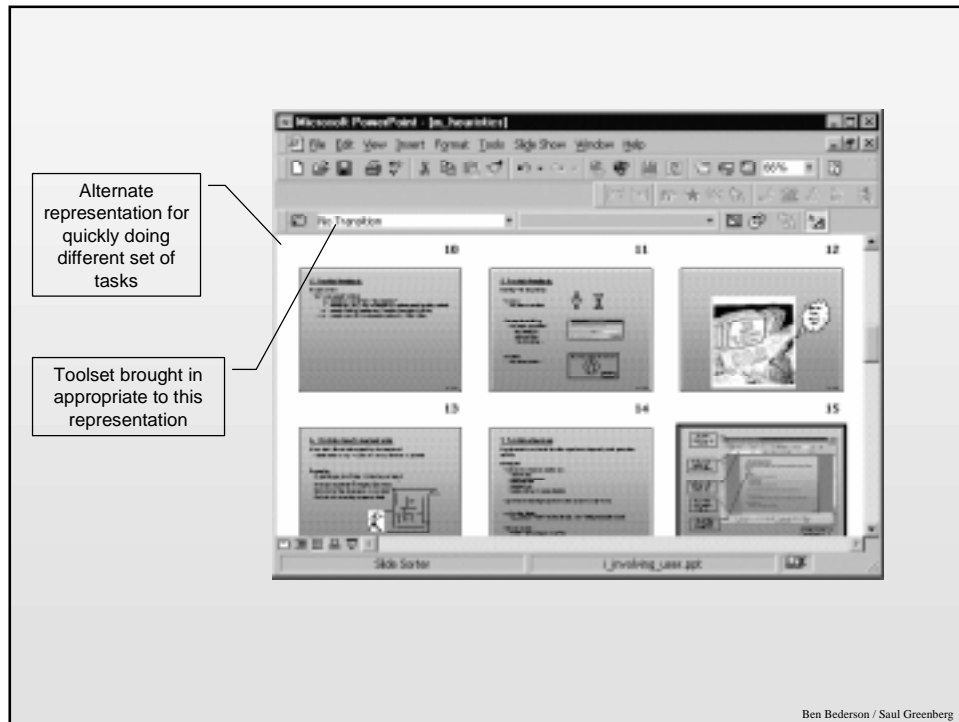
### Strategies:

- keyboard and mouse accelerators
  - abbreviations
  - command completion
  - menu shortcuts
  - function keys
  - double clicking vs menu selection
- type-ahead (entering input before the system is ready for it)
- navigation jumps
  - e.g., going to window/location directly, and avoiding intermediate nodes
- history systems
  - WWW: ~60% of pages are revisits

Ben Bederson / Saul Greenberg



Ben Bederson / Saul Greenberg



## **8: Deal with errors in a positive and helpful manner**

**People will make errors!**

### **Errors we make**

- Mistakes
  - arise from conscious deliberations that lead to an error instead of the correct solution
- Slips
  - unconscious behaviour that gets misdirected en route to satisfying goal  
e.g. drive to store, end up in the office
  - shows up frequently in skilled behaviour usually due to inattention
  - often arises from similarities of actions



Ben Bederson / Saul Greenberg

## Types of slips

### **Capture error**

- frequently done activity takes charge instead of one intended
  - occurs when common and rarer actions have same initial sequence
    - change clothes for dinner and find oneself in bed (William James, 1890)
    - confirm saving of a file when you don't want to replace it



Ben Bederson / Saul Greenberg

## Types of slips

### **Description error**

- intended action has much in common with others that are possible
  - usually occurs when right and wrong objects physically near each other
    - pour juice into bowl instead of glass
    - go jogging, come home, throw sweaty shirt in toilet instead of laundry basket
    - move file to trash instead of to folder

### **Loss of activation**

- forgetting what the goal is while undergoing the sequence of actions
  - start going to room and forget why you are going there
  - navigating menus/dialogs and can't remember what you are looking for
  - but continue action to remember (or go back to beginning)!

### **Mode errors**

- people do actions in one mode thinking they are in another
  - refer to file that's in a different directory
  - look for commands / menu options that are not relevant

Ben Bederson / Saul Greenberg

## Designing for slips

### General rules

- Prevent slips before they occur
- Detect and correct slips when they do occur
- User correction through feedback and undo



### Examples

- capture errors
  - instead of confirmation, make actions undoable
  - allows reconsideration of action by user
    - e.g. Mac trash can can be opened and “deleted” file taken back out
- description errors
  - in icon-based interfaces, make sure icons are not too similar,
  - check for reasonable input, etc.
- loss of activation
  - if system knows goal, make it explicit
  - if not, allow person to see path taken
- mode errors
  - have as few modes as possible (preferably none)
  - make modes highly visible

Ben Bederson / Saul Greenberg

## Generic system responses for errors

### Interlock

- deals with errors by preventing the user from continuing
  - eg cannot delete an object if none are selected

### Warn

- warn people that an unusual situation is occurring
- when overused, becomes an irritant
  - e.g.,
    - audible bell
    - alert box



Ben Bederson / Saul Greenberg

## Generic system responses for errors continued...

### **Do nothing**

- illegal action just doesn't do anything
- user must infer what happened
  - enter letter into a numeric-only field (key clicks ignored)
  - put a file icon on top of another file icon (returns it to original position)

### **Self-correct**

- system guesses legal action and does it instead
- but leads to a problem of trust
  - spelling corrector

### **Lets talk about it**

- system initiates dialog with user to come up with solution to the problem
  - compile error brings up offending line in source code

### **Teach me**

- system asks user what the action was supposed to have meant
- action then becomes a legal one

Ben Bederson / Saul Greenberg

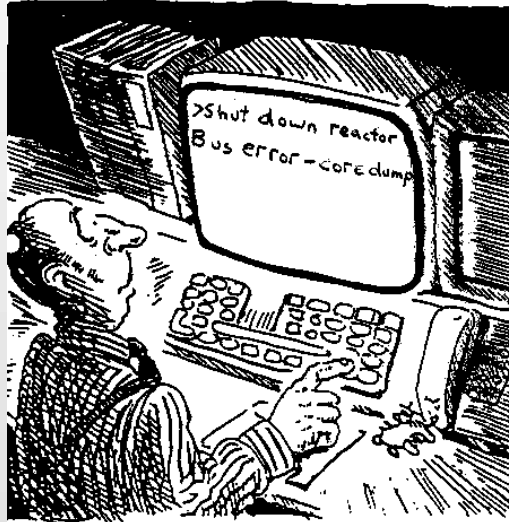
## 8 Deal with errors in a positive and helpful manner



What is "error 15762"?

Ben Bederson / Saul Greenberg

## 8 Deal with errors in a positive and helpful manner



*A problematic message to a nuclear power plant operator*

Ben Bederson / Saul Greenberg

## 8 Deal with errors in a positive and helpful manner

### **Provide meaningful error messages**

- error messages should be in the user's language (preferably task language)
- don't make people feel stupid

Try again, bonehead!

Error 25

Cannot open this document

Cannot open "chapter 5" because the application "Microsoft Word" is not on your system

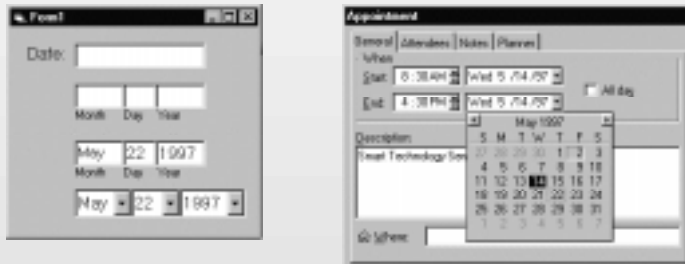
Cannot open "chapter 5" because the application "Microsoft Word" is not on your system. Open it with "Teachtext" instead?

Ben Bederson / Saul Greenberg

## 8 Deal with errors in a positive and helpful manner

### Prevent errors

- try to make errors impossible
- modern widgets: only “legal commands” selected, or “legal data” entered



### Provide reasonableness checks on input data

- on entering order for office supplies
  - 5000 pencils is an unusually large order. Do you really want to order that many?

Ben Bederson / Saul Greenberg

## Consumer Manuals...



## **9. Provide help**

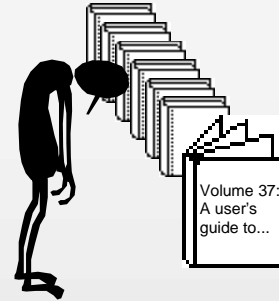
**Help is not a replacement for bad design!**

### **Simple systems:**

- walk up and use; minimal instructions

### **Most other systems:**

- feature rich
- some users will want to become “experts” rather than “casual” users
- intermediate users need reminding, plus a learning path



Ben Bederson / Saul Greenberg

## **Documentation and how it is used**

### **Many users do not read manuals**

- prefer to spend their time pursuing their task

### **Usually used when users are in some kind of panic, need immediate help**

- indicates need for online documentation, good search/lookup tools
- online help can be specific to current context
- paper manuals unavailable in many businesses!
  - e.g. single copy locked away in system administrator's office

### **Sometimes used for quick reference**

- syntax of actions, possibilities...
- list of shortcuts ...

Ben Bederson / Saul Greenberg

## Types of help

### Tutorial and/or getting started manuals

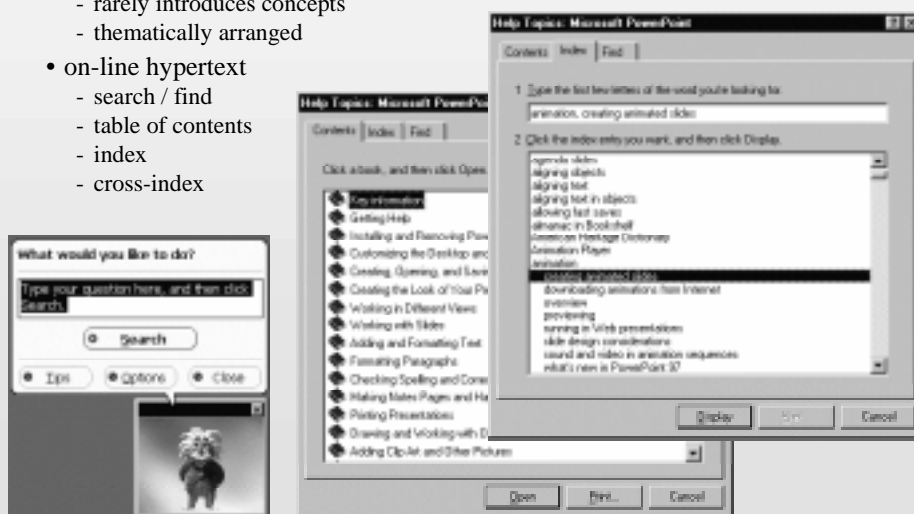
- short guides that people are likely to read when first obtaining their systems
  - encourages exploration and getting to know the system
  - tries to get conceptual material across and essential syntax
- on-line “tours”, exercises, and demos
  - demonstrates very basic principles through working examples

Ben Bederson / Saul Greenberg

## Types of help

### Reference manuals

- used mostly for detailed lookup by experts
  - rarely introduces concepts
  - thematically arranged
- on-line hypertext
  - search / find
  - table of contents
  - index
  - cross-index



Ben Bederson / Saul Greenberg

## Types of help

### Reminders

- short reference cards
  - expert user who just wants to check facts
  - novice who wants to get overview of system's capabilities
- keyboard templates
  - shortcuts/syntactic meanings of keys; recognition vs. recall; capabilities
- tooltips
  - text over graphical items indicates their meaning or purpose



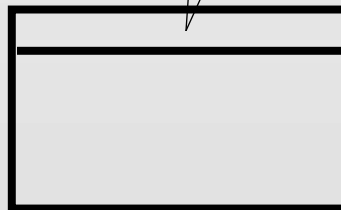
Ben Bederson / Saul Greenberg

## Types of help

### Context-sensitive help

- system provides help on the interface component the user is currently working with
    - Macintosh "balloon help"
    - Microsoft "What's this" help
- brief help explaining whatever the user is pointing at on the screen

**Title bar**  
To move the window, position the pointer in the title bar, press the button, and drag it to the new position

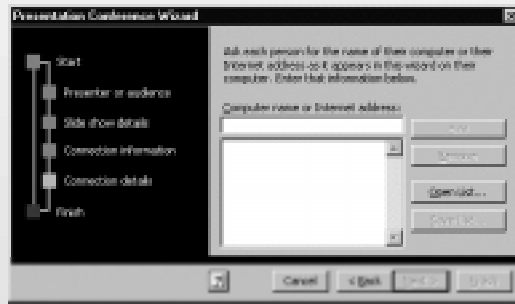
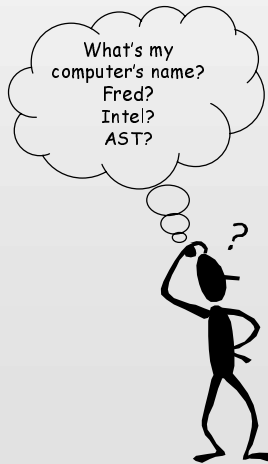


Ben Bederson / Saul Greenberg

## Types of help

### Wizards

- walks user through typical tasks
- *but* dangerous if user gets stuck

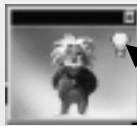


Ben Bederson / Saul Greenberg

## Types of help

### Tips

- migration path to learning system features
- also context-specific tips on being more efficient
- must be "smart", otherwise boring and tedious



Ben Bederson / Saul Greenberg