



User Interface Design



GUIs: “bones” vs. cosmetics

- ▶ A lot of GUI design effort ends up focusing on frills
 - ▶ Colors, “cool” graphics and icons, fancy widgets
 - ▶ Online help systems
 - ▶ Animation, multimedia
- ▶ These are the “bells and whistles” that are added after the GUI foundation is in place (when appropriate).
- ▶ The basic experience is created by the underlying structure and concept of the interface
 - ▶ Start with the basic “bones” before you add the frills.



Understanding the users

- ▶ Need to focus on the profile and characteristics of the users
 - ▶ Level of expertise using software in general
 - ▶ Likely frequency of usage
 - ▶ Capabilities and constraints e.g. young users, users using dialup lines, using public terminals
- ▶ User background and expectations
 - ▶ How they think about this problem domain and this application
 - ▶ The specific expectations they have from the system: get work done quickly? Enjoy using it? Power – lots of different capabilities? Flexibility?



Understanding the tasks

- ▶ What do they want to do with the software?
- ▶ What operations will they perform frequently?
 - ▶ Design to make most frequent tasks most convenient
- ▶ Do they already have ideas about how to do the operations?
- ▶ What variations are likely in the way they perform tasks?

- ▶ E.g. Two contrasting models, each useful for their context:
 - ▶ Applications such as MS-Word
 - ▶ perform hundreds of different tasks from the same screen
 - ▶ Wizards
 - ▶ perform exactly one task, with minor parameters and variations!



User Interface - the Foundation

- ▶ Need a consistent concept underlying the entire UI
 - ▶ Games do a very good job of this: set up a “world” that users move into
 - ▶ Note how each application type has a model: browsers, spreadsheets, word processors, databases, calendar managers, messengers, mail programs...
 - ▶ Sometimes the model is a “metaphor” e.g. the desktop, folders and documents “worktable” metaphor
 - ▶ Sometimes the model is a standard for the application type e.g. graphics programs, first-person shooters, mail programs, ...
- ▶ Using the model consistently throughout makes it easier for users to figure out how to perform tasks



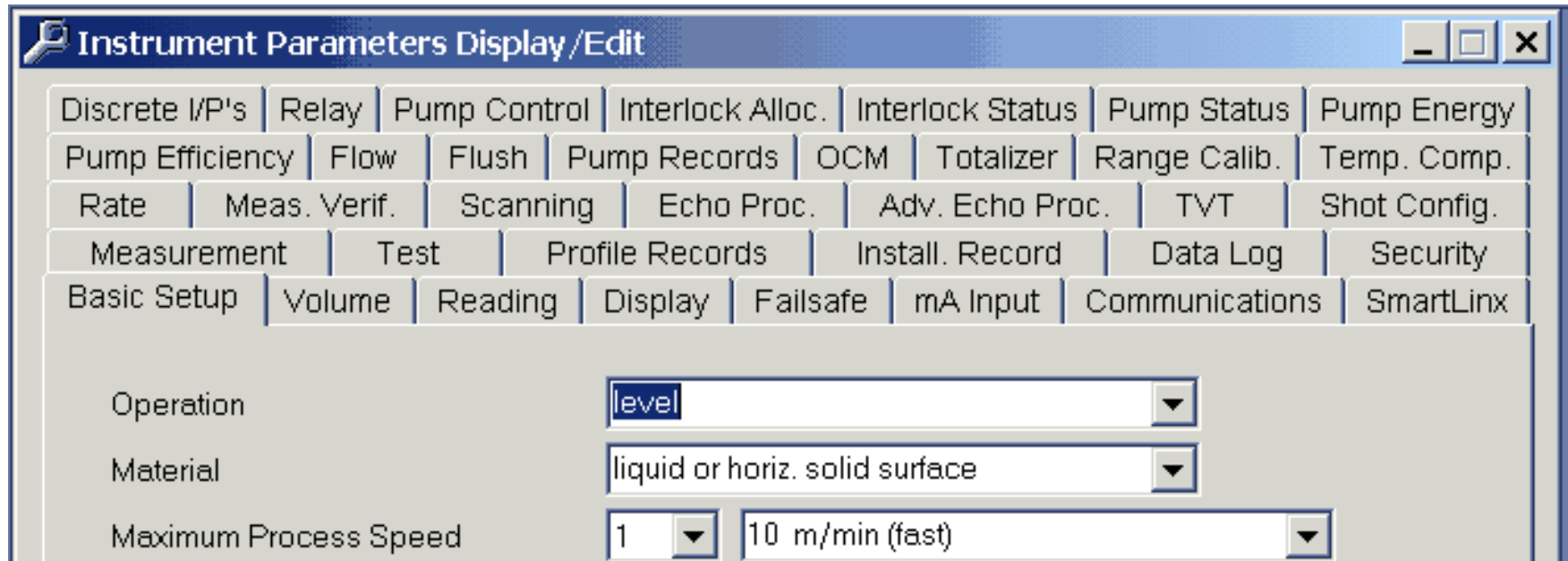
User Interface - Navigation Plan

- ▶ Group related tasks together
 - ▶ Grouping levels: Screens, tabs, menus
 - ▶ Acid test: How easy is it to figure out where to look for functions
- ▶ Have a consistent navigation scheme between screens
 - ▶ Match it to user workflow
 - ▶ Make frequently used tasks easy to access.
 - ▶ Tradeoff: avoid clutter – packing too much into one screen
- ▶ Be consistent in grouping and behavior
- ▶ State transition diagrams often useful to depict navigation schemes



Be Careful -- Extremes

- ▶ Tabs can be useful in organizing information, but having too many can lead to confusion



Screen Layout & Appearance

- ▶ Important stuff placed in the “prime” locations
- ▶ Obvious is good!
 - ▶ Use of appropriate shapes, colors, icons for your users and the domain
 - ▶ Match with look-and-feel of similar applications (standards)
 - ▶ Be consistent with these items as well.
- ▶ Fancy stuff is distracting and jarring – when used occasionally and carefully, fancy stuff can be eye-catching and attractive



Use of widgets

- ▶ Different GUI widgets are useful for different purposes
 - ▶ E.g. push buttons vs. radio buttons vs. checkboxes vs. drop-down lists
 - ▶ Each has semantics associated with it - educate yourself
 - ▶ Tradeoffs between different widgets e.g. drop-down lists save space, but less convenient than radio buttons
- ▶ Use widgets consistently throughout your application

8) Age:

9) ☒ Female
☒ Male


Enter your Social Security Number:

0	0	0	-	0	0	-	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---

Usability

- ▶ **Ease of learning**
 - ▶ Able to use software with little or no training
 - ▶ Use of software is faster the second time
- ▶ **Productivity**
 - ▶ Perform tasks quickly and efficiently
- ▶ **Minimal error rates**
 - ▶ Knowing whether operations worked, and if not, why it failed, and what can be done about it
 - ▶ If error occurs, provide good feedback so user can recover
- ▶ **High user satisfaction**
 - ▶ Users feel like they are driving – they issue the commands and software executes it, rather than it pulling them along and dictating to them
 - ▶ Includes the ability to customize environment.





User Interface Design - Some Don'ts

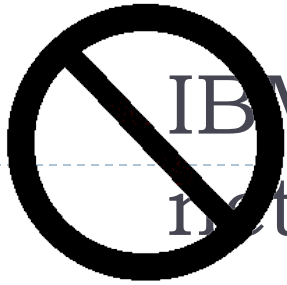


Keep the users busy doing unnecessary work.

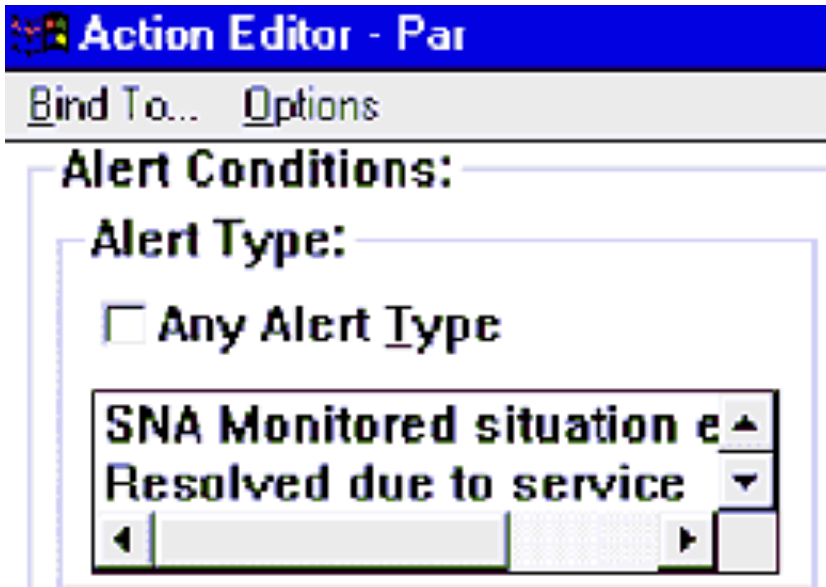


- ▶ **Example:** It's a "good" habit to let users enter data that the system already knows and could provide beforehand.
- ▶ **Example:** Let users put out a lot of effort to scroll around in text boxes in order to read information.





IBM's NetFinity (supervising networks & remote computers)



- ▶ Because of their desire to fit everything into a single dialog, IBM's designers made sacrifices.
- ▶ The designers sacrificed the user's ability to read some of the controls, such as the list of potential alert conditions, shown here.
- ▶ The user can scroll through the list one alert at a time, and will have to scroll both left and right to view the items.



Do not obey standards

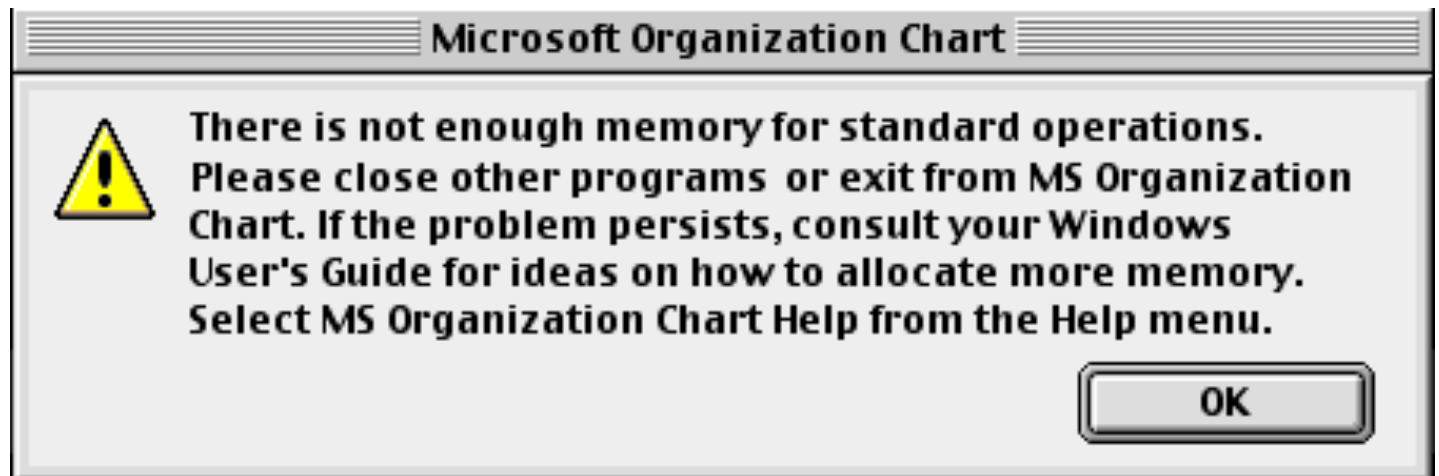
- ▶ **Example:** Do not use standard screen elements for a given purpose, such as single selection (e.g. use checkboxes instead of radio buttons because they look nicer).
- ▶ **Example:** Do not place menu items into the categories and locations they typically belong to (e.g. place "Save" in the "Edit Menu").

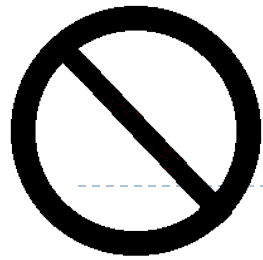




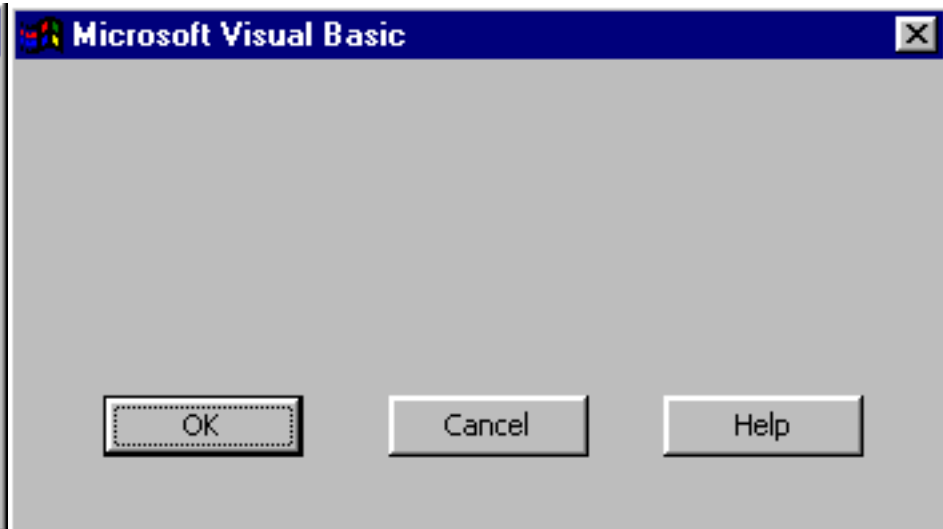
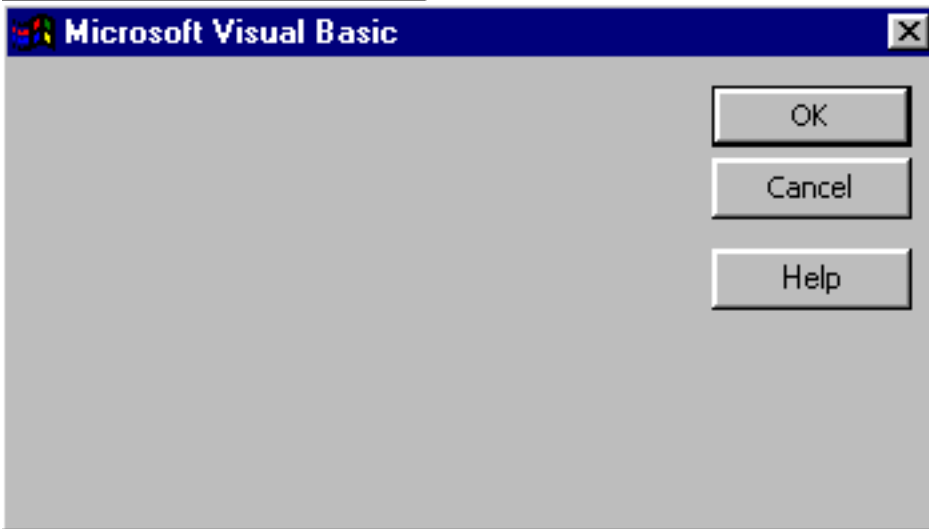
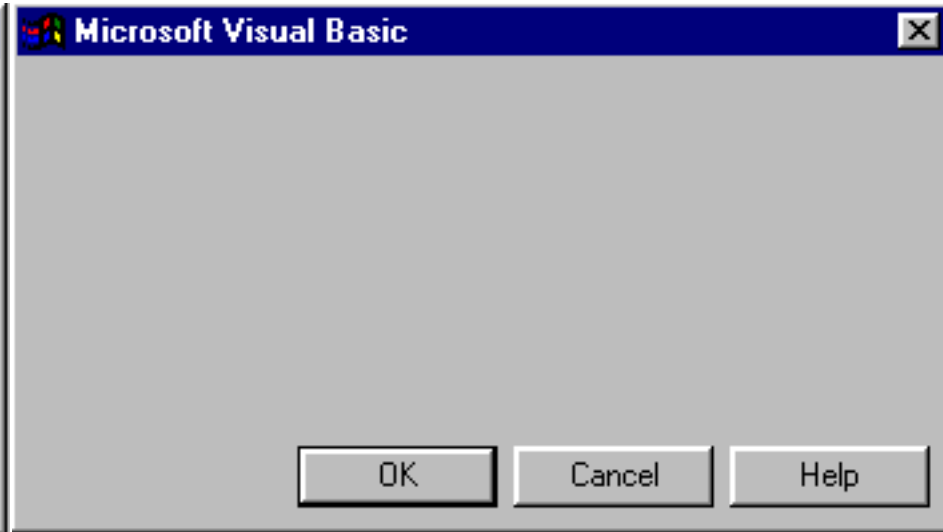
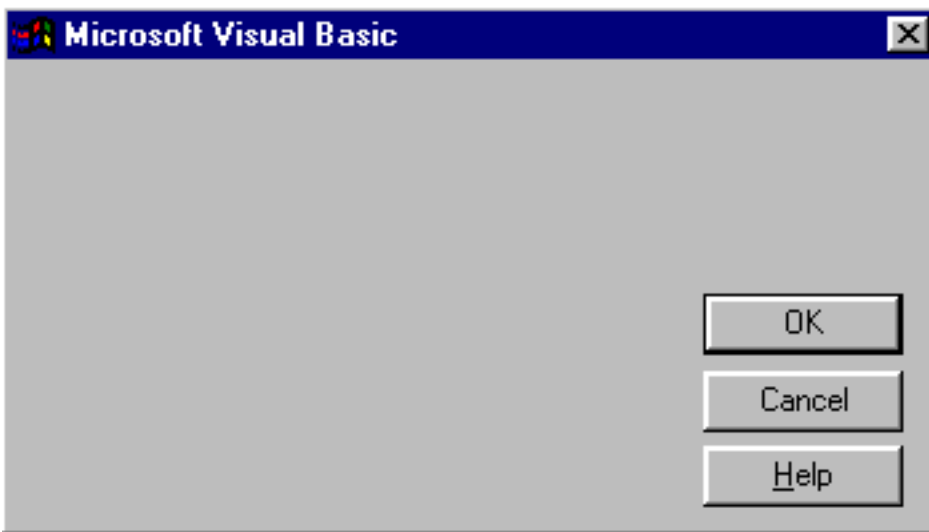
Word 98 for the Macintosh

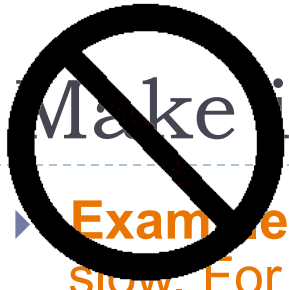
- ▶ This image is an error message received from Microsoft's *Office 98* on a PowerMac when attempting to add a flow chart to a Word document.





Visual Basic 5 (compilation)

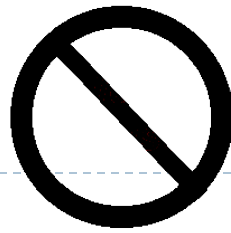




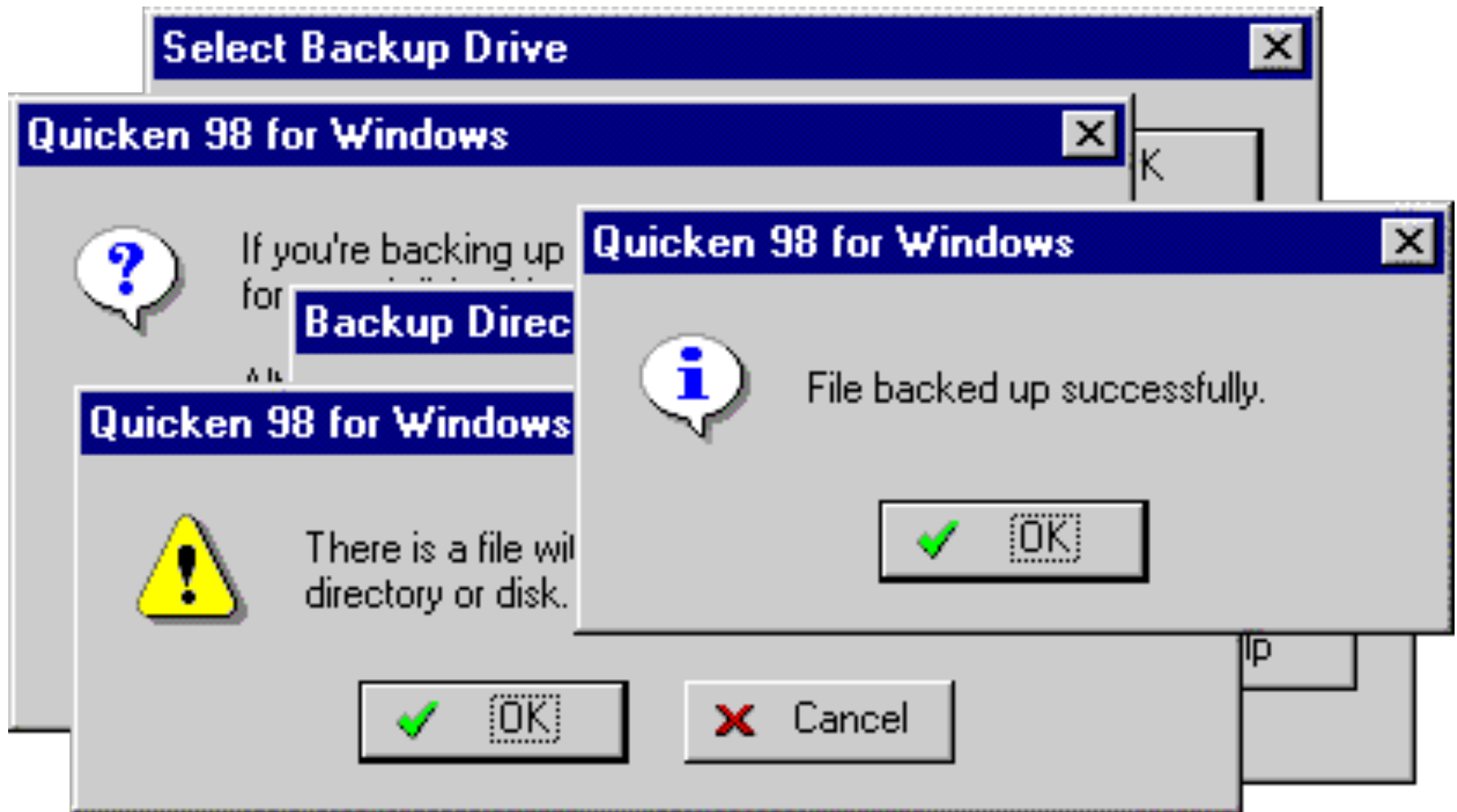
Make it slow!

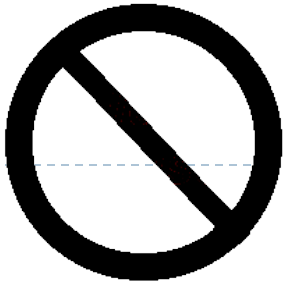
- ▶ **Example:** There are nearly unlimited possibilities of making software slow. For example, you can include long lasting checks or roundtrips after each user input. Or you can force users through long chains of dialog boxes.



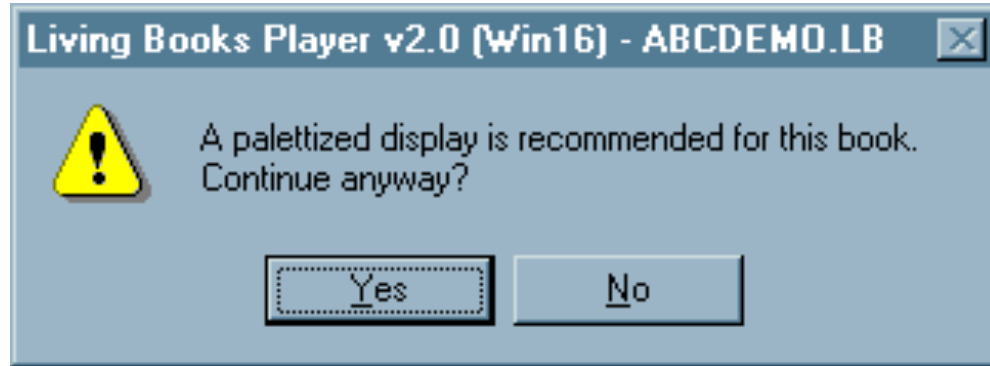


The Backup Process in Quicken98



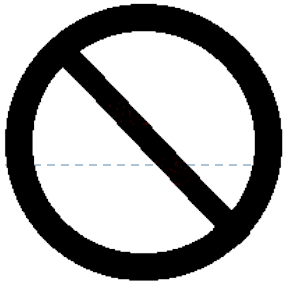


Dr. Seuss' ABC

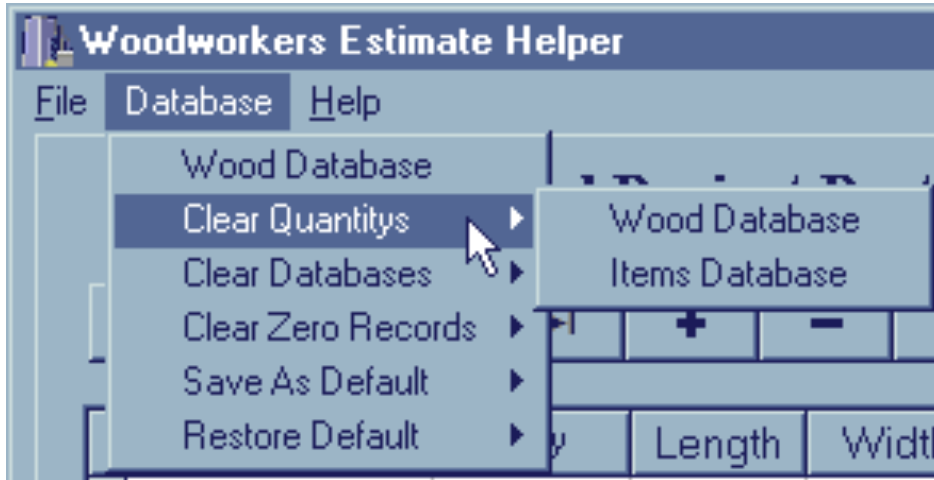


- ▶ While some programmers might readily understand this message, we would consider it beyond the understanding of most computer users, especially for the target population of the particular application that generated the message.
- ▶ The game is intended for **3 to 5 year-old children**. Funny thing though, the message is completely unnecessary, since the program works just as well at any typical display setting.

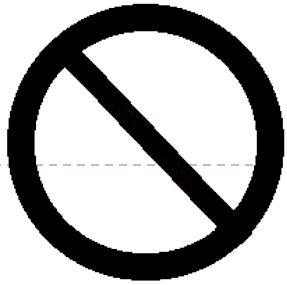




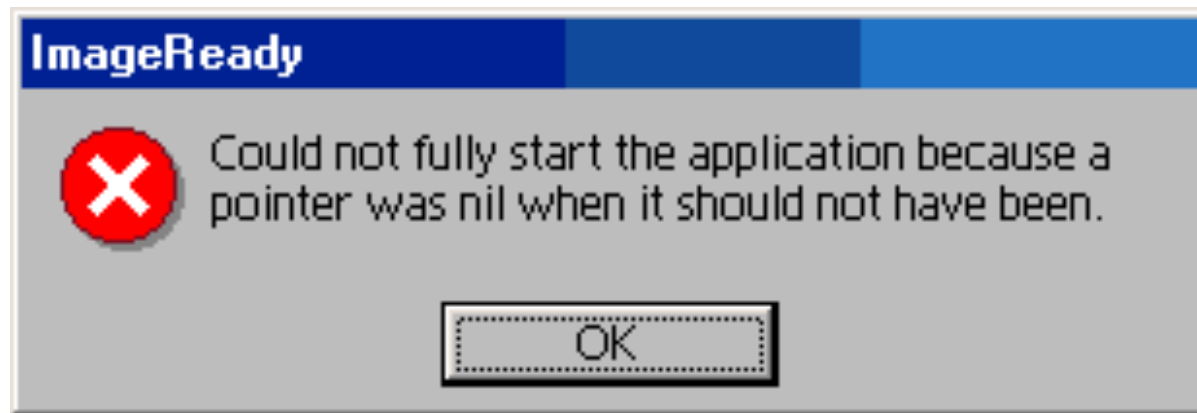
Woodworkers Estimate



- ▶ The program is "designed for woodworkers and cabinet makers", and purports to assist in the process of calculating price quotes for their projects.
- ▶ Unfortunately the program uses such esoteric programming terminology as "Databases", "Records", and, if the user attempts to enter a duplicate part name, presents the message "Key Validation Error".



Adobe ImageReady

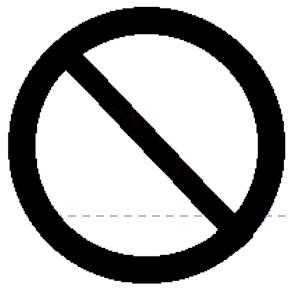


Hide Important and often-used functionality from the users' view



- ▶ **Reasoning:** This strategy stimulates users to explore your application and learn a lot about it.
- ▶ **Example:** Hide important functions in menus where users would never expect them.
- ▶ **Example:** Hide functionality in cryptic icons.

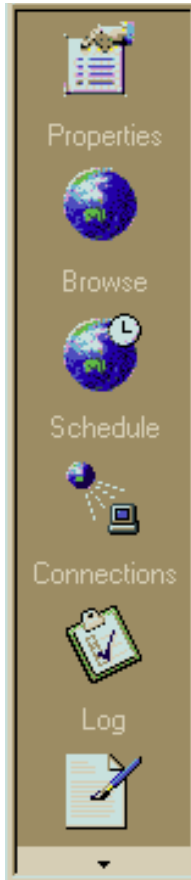




ccMail



- ▶ While some of the images are immediately recognizable (e.g., the printer, and the trashcan), the functions of many of the images are completely unclear.
- ▶ To make matters worse, *ccMail* does not provide ToolTips ("bubble help") for the toolbar images.



- ▶ *WebZip* utilizes a navigation toolbar.
- ▶ The navigation toolbar is scrollable, but the designers essentially hid this fact from the user.



Make your application mouse-only - do not offer any keyboard shortcuts

- ▶ **Reason 1:** This will make your application completely inaccessible to visually impaired users. Therefore, you can leave out all the other accessibility stuff as well. That will save you a lot of development time.
- ▶ **Reason 2:** This will drive many experts crazy who used to accelerate their work with keyboard shortcuts. Now, they will have more empathy for beginners because they are thrown back to their speed.



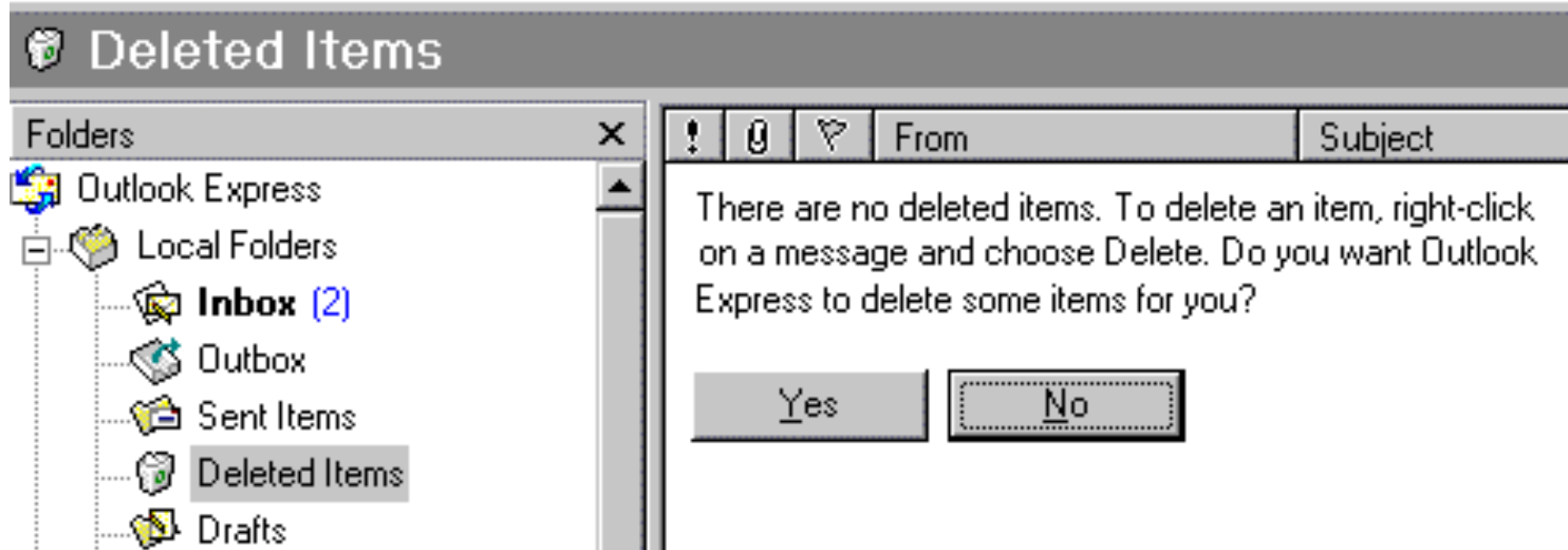
Make Using Your Application a Real Challenge!

- ▶ **Reasoning:** This teaches people to take more risks, which is important particularly when they are under pressure or have other things they could be doing with their time.
- ▶ **Example:** Do not offer an *Undo* function.
- ▶ **Example:** Do not warn users if actions can have severe consequences.
- ▶ **Note:** If you want to top this and make using your application like playing Russian roulette, change the names of important functions, such as *Save* and *Delete*, temporarily from time to time...





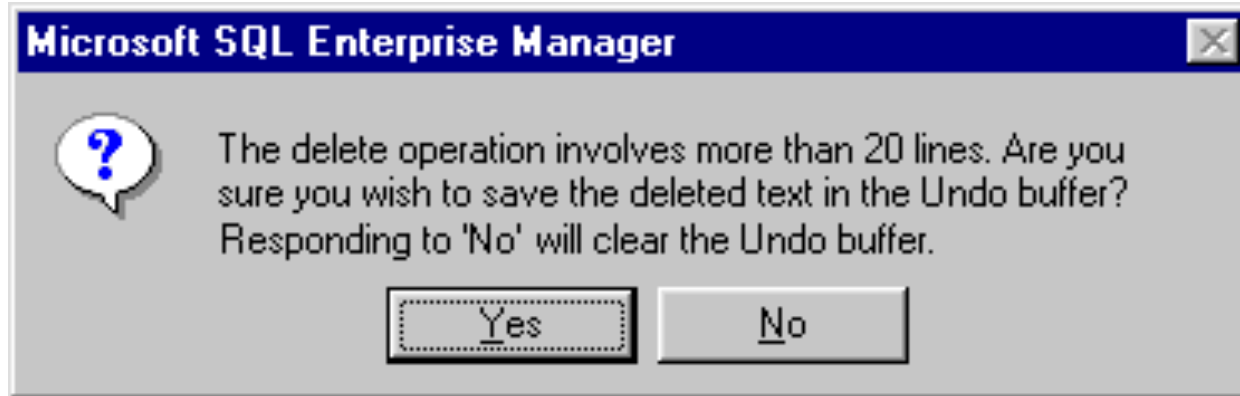
Microsoft Outlook v.5



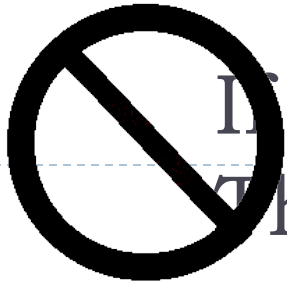
- ▶ The message states that there were no items in the Deleted Items folder and do I want Outlook to delete something.
- ▶ What sort of a message is that?!? Why would you want Outlook to delete a randomly-selected piece of mail?



Microsoft SQL Manager



- ▶ If you select more than 20 lines of text from the query window, and then accidentally hit delete, you get this apparently helpful error message, so that if you change your mind, you can hit "No" to avoid the delete. However, closer inspection of the message reveals a glaring inconsistency. Hitting "No" actually causes the Delete operation to continue, but destroys the Undo buffer in the process, leaving no means of canceling an accidental delete.



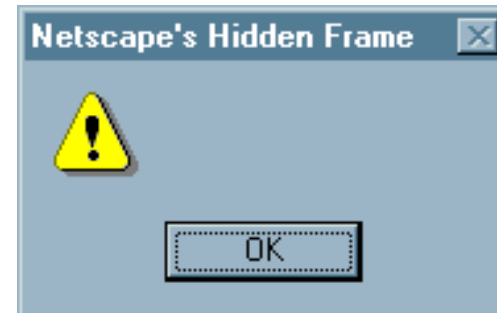
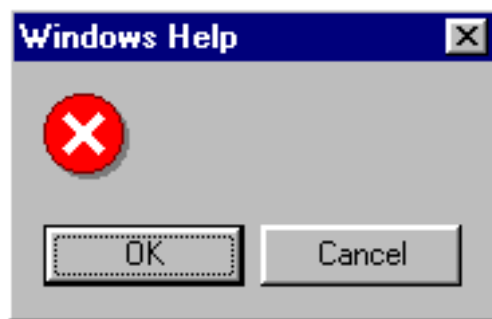
If You Can't Say Anything Useful, Then Don't Say Anything At All

- ▶ **Reasoning:** This teaches people to be more self-reliant and to pay more attention to what they are doing in the first place.
- ▶ **Example:** Do not give users feedback when they deleted something.
- ▶ **Example:** Do not offer status when copying files.





Various Applications

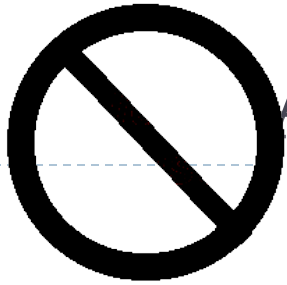


Insult Your Users Rather than Provide Useful Information

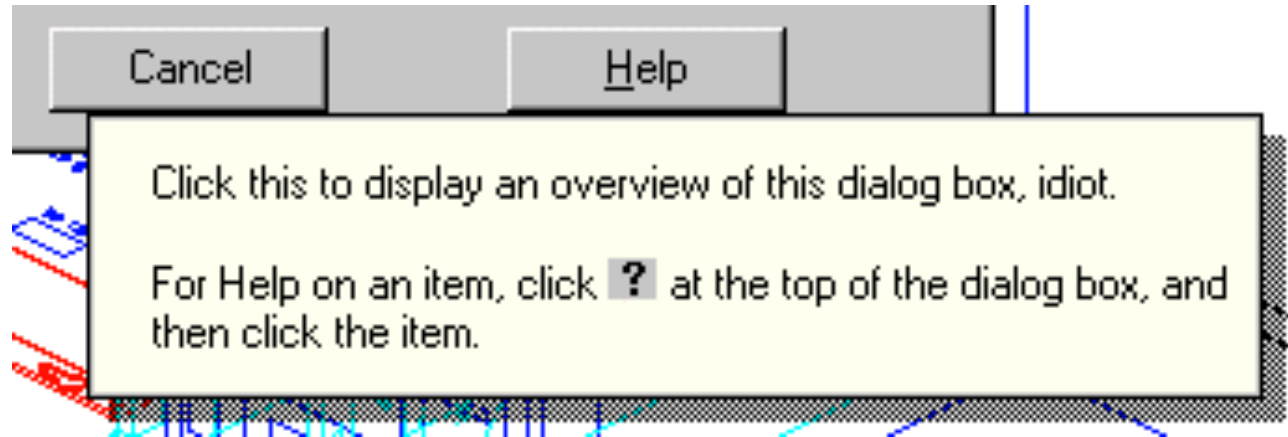


- ▶ **Reasoning:** This teaches people to keep from making the mistake again by not using the feature at all and will keep customer support calls low (as well as your customer base).
- ▶ **Example:** Calling them names
- ▶ **Example:** Being general with errors rather than providing useful information.





Autocad Mechanical



- ▶ The author of the message considered it a joke that would never be seen by the users.
- ▶ As a result of the joke, the employee is now a *former* employee, and AutoDesk has sent thousands of letters of apology to its customers.