Building a Good User Interface

by Mike Callahan

Although many programmers have excellent ideas for new and different programs, they simply don't spend enough time on the user interface Old pros know the user interface is as important as the program itself A good interface can make the difference between users gladly registering a program, or grudgingly using it until something better comes along

The type of program I'm referring to is a highly interactive application, such as a wordprocessor The interface discussed in this article may not fit all programs, but before you think "This doesn't apply to my program," take a second look at how much interaction it requires

When designing a good interface five rules of thumb apply·

Rule No. 1: Don't be too clever.

Use existing standards whenever possible Remember customers will be using several different programs besides yours If your program's interface is too different, you'll confuse users

Many programmers feel they must make their program different to avoid the dreaded "look and feel" lawsuit. While the software industry is becoming increasingly "litigation crazy," if you use a text-based interface, you shouldn't have anything to fear Many software firms use the interface I'll be describing and I've never heard a company claim to own the design

Rule No. 2: Don't hide your guide.

Early PC programs were guilty of this — they would have different layers of menus, but users wouldn't know how to retrieve them from the main working screen The best solution is to have a menu bar at the top of the screen This idea should be the foundation of your design.

Rule No. 3: Never overwhelm users.

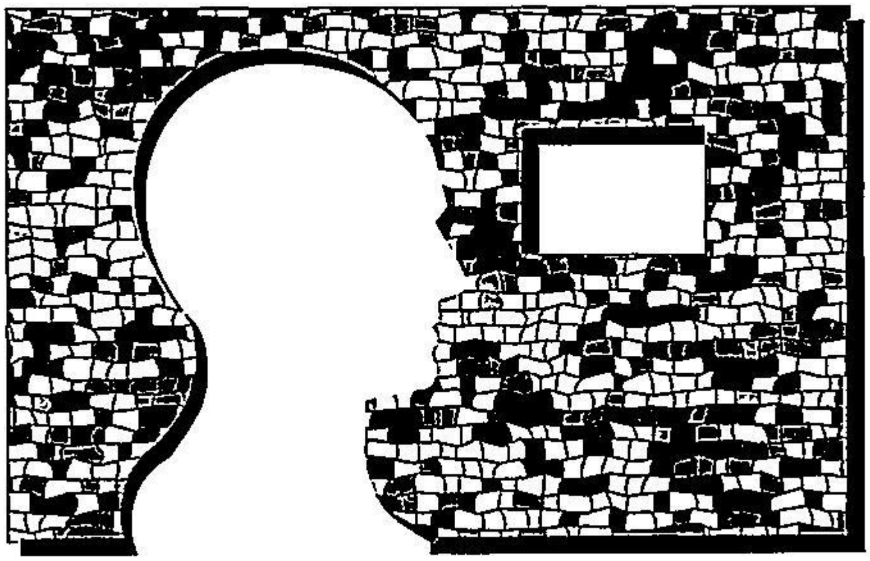
If your program is complex, hide the complexity in layers. This applies to both commands and help information Early PC programs placed important commands at the top of the screen, resulting in a menu bar several lines deep This was almost as bad as no help at all Use pull-down and pop-up menus to follow this rule

Rule No. 4: Only give users what they need.

It's a common mistake to overwhelm users with on-line help systems. When users ask for help, they're dropped into a large help index containing all the help for the entire program. Users may not know what index entry the information is stored under Also, adept users, in the middle of complex action, don't want to be bothered with going through help indexes to find the one bit of help they need at that point. Two methods that are better than the help index are context-sensitive help and hypertext.

Rule No. 5: Don't slow down power users.

This means that once users become familiar with a program, they shouldn't have to work their way through a maze of menus to perform common actions. There should always be shortcuts for seasoned users



Continued on page 52

PROGRAM (cont'd from page 41)

Now, to incorporate these ideas into a foundation for a user interface design Elements in the design will be familiar to many users

The first step is to write all the commands in your program, and describe them in one or two words. Using these descriptions, choose a unique selection letter (usually the first)

Next, group the commands into at least five, no more than ten, major categories (If you have more than ten, you may need to collect several commands into a subcategory) Make sure major categories are labeled with a single word and a unique selection letter Then write down the categories, subcategories, and commands, capitalizing the selection letter

After your commands are in a structure, choose frequently used ones and assign a unique shortcut key to them (Try to use the Control-key with the selection letter, if possible Function keys can be used, but it's easier to remember Control-S for Save rather than F3) Write this key down next to the command Control is usually abbreviated as ^ and Shift as s, so write ^S for Control-S, but sF3 for shift F3 Using this scheme, most users know to press ALT keys when selecting menus, and Control or Function keys when selecting shortcuts

It's also a good idea to see how similar programs label common commands and shortcuts Remember the first rule, users will feel right at home if you don't get too creative during this step. To be on the safe side, however, don't copy a program exactly

Here's how to arrange commands in a structure, using a wordprocessor as an example·

1 Load a disk file as the working document
2 Save the working document into a disk file
3 Rename the working document into a different disk file
4 Display a disk file on the screen without disturbing the working document
5 Send the working document to a device formatted for a printer
6 Display a directory of the disk
7 Mark or unmark a block of text
8 Copy a block of text to another location
9 Move a block of text to another location.
10 Delete a block of text
11 Read in a file as a block of text.
12. Write a block of text into a disk file

Here's a way to structure and label them.

```
FILE                  BLOCK
Save        ^S        Mark/Unmark    F2
Load        ^L        Copy           ^C
Rename                moVe           ^V
Examine               Delete
Directory^D           Read
Print                 Write
PRINT
Printer     ^P
Disk
Screen
```

In this example, you have a menu bar with two categories File and Block. The File pull-down menu has one submenu, Print. Shortcut keys exist for Save File, Load File, Directory, Mark/Unmark Block, Copy Block, Move Block, and Print to Printer commands

It's acceptable to repeat selection letters, as long as they aren't in the same category or subcategory Notice you can't use M as the selection letter for the Move command in the Block pull-down menu, since you've used it for the Mark command Instead, use V and capitalize it Although it may look strange, it's common in programs today Do not repeat shortcut keys, make sure all are unique

When you're done, the hierarchy of your command structure will be visible In fact, structure programming methods demand that you do this before a single line of code is written Take a good look at the structure and make any changes before proceeding

Place major categories in a menu bar at the top of the screen, commands and subcategories in pull-down menus, and submenus in pop-up menus A few expert users find the menu bar on the screen at all times distracting, but most programs I use do this. You can pop the menu bar in and out as users press a certain key, but that's distracting and users may forget what key to press.

The appearance of the menu bar and pull-down menus is extremely important. Colors and attributes should be selected carefully according to the type of display being used (MDA, CGA, EGA, or VGA) You might want to draw lines around the menus The width of the menus should accommodate the largest command along with its shortcut key, with the right edge under the first letter of the menu bar category, if possible.

Submenus will appear in pop-up menus either beside or on top of the pull-down menu, as long as the subcategory name remains in view

This is how the interface works users select a major category on the menu bar by pressing ALT-letter at any time. The category selected will then be shown in reverse-video, and the pull-down menu will appear, with the top command in the menu also in reverse-video

If users press the right or left arrow keys, the next or previous category will be highlighted and the associated pull-down menu will appear If users press the up or down arrow keys, the next or previous choice in the pull-down menu is highlighted in reverse video If users press the ENTER key, the highlighted command will be executed or the subcategory pop-up menu will appear Also, the command will be executed, or the subcategory pop-up menu will appear if they press any selection letter for the current pull-down menu.

If users press the ESC key, the program will back-up one level (another standard practice) Pressing F1 displays a help screen for that level (that will be explained later), and any other key press will be ignored.

Of course, if users press the Control key and a letter, or a function key, the shortcut command would be executed, bypassing all menus By listing all shortcut keys in the pull-down menus, users quickly learn which commands have shortcuts

WordPerfect (cont'd from page 16)

You then move to "Creating the Document," where you practice listing files, changing directories, retrieving files from disk, and typing in text You also see the embedded codes that are the secret of successfully mastering WordPerfect.

"Editing the Document" covers such useful tricks as centering, indenting paragraphs, deleting and inserting text, blocking text, and other common editing functions

There are also lessons on using spell check, the thesaurus, printing, columnar text, and math

Anyone who assiduously works through these lessons should be able to start using WordPerfect fairly productively, assuming they remember enough of what they read

One way to recall it is to use the workbook-style manual you receive when you register. It provides quizzes, additional practice sessions, and printed versions of much of the training material. Working through the manual along with the computer exercises improves your retention dramatically However, the manual is not essential to the program

It's Not Perfect

I only have a few minor quibbles with this program For one thing, I can't understand why the author failed to use color Only a few features are in color, even though the Bricklin demo program supports stunning color work A little more work could have produced a program that works well in both color and monochrome

Periodically, the lesson presents a screen one character at a time, with an audible click as each character appears. The words materialize laboriously, letter by letter, including at times a click for each space in a blank line. Sometimes a full screen takes up to two minutes to dribble out, which seems an eternity at the computer screen The speed of these screens doesn't depend on your computer's speed. they're no faster on my new 20-megaHertz Arche Rival 386 than on my old XT

Although the whole presentation needs the services of a good editor to remove inelegances, such as non-parallel construction and wordiness, the entire package is professional in appearance and content

Registering

If you try the program and like it, register At $79 95, it is competitively priced with commercial WordPerfect training packages and is every bit as good. You'll get a workbook, free telephone support, a free upgrade to the next version of the program, and a $20 commission on subsequent registrations traced to your personal code number There are also a series of materials for instructors who want to use this program in the classroom

If you ever need to learn WordPerfect, avoid any frustration by getting this excellent training package The incredible power of WordPerfect awaits you; all you need is a little help to learn how to make it work for you ❑

Richard O. Mann, CPA, CIA of Roy, Utah, is an internal auditor for The Church of Jesus Christ of Latter-day Saints and Contributing Editor of PC Today and PC Novice magazines

Using the wordprocessor as an example, look at the key presses necessary to print a document. Using the menus, press ALT-F, to display the File pull-down menu. Then press P to display the Print sub-menu and press P again, to print the file.

The arrow keys can be used also. After pressing ALT-F, press the down arrow five times to highlight Print, and then press ENTER. The Print sub-menu is now displayed, with the Print command already highlighted. Press ENTER to print the file. While this method may seem clumsy, many beginning users like to use the arrow keys.

A seasoned user would simply press Control-P to use the Print shortcut and bypass all the menus.

When designing help screens always make the F1 key the help key, almost every program uses it. Help should always be context-sensitive. Using the wordprocessor designed above as an example, suppose you've pressed the ALT-F key and the File pull-down menu is displayed. If you press F1, you'll see a screen that lists and defines the Save, Load, Rename, Examine, Directory commands, and explains that Print is in a submenu. If the Print sub-menu was displayed, pressing F1 would show a screen with an explanation on how to print to the printer or screen.

This method works well, but including hypertext help is even better. Hypertext links help screens by words. When a help screen appears, linking words are either in bold text or different colors. The first hypertext link will be in reverse video. Using arrow keys, users highlight the word they want to know more about, and press ENTER. Another screen appears explaining the selected word with more hypertext links. The process continues until users find out everything they want to know.

The disadvantage to using hypertext help in a program is the amount of disk space the screens use. With a good hypertext design, however, users soon will find the program manual more practical as a paperweight.

Notice these ideas obey all the rules. The design is similar to those found in popular programs, help is always on the screen, a limited number of choices are on the screen at any one time, users get help only when they want it, and the power user has shortcuts available.

Remember this design is a foundation for an interface and not state-of-the-art. Add to these ideas as you see fit. A popular item you might want to support is a mouse. Although you shouldn't make a mouse mandatory, because some people just don't like the little rodents.

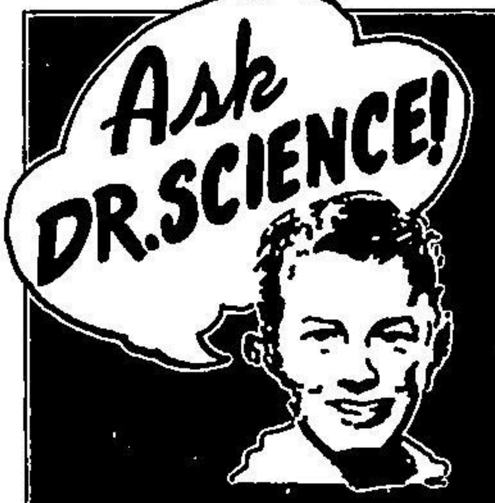
If you make it possible for users to customize the screen colors, window sizes, and even shortcut keys, save each configuration in a unique file so users can have a different setup.

You might want your program run under some of the popular graphic user interfaces (GUI) such as Tandy's Deskmate, Microsoft's Windows, or Digital Research's GEM. This will make some of your design choices easier, but you need to obtain a programmers development kit. You'll be a pioneer in the shareware field, however, since few shareware programs use commercial GUIs.

These interfaces are a lot more complicated than the programs with menus, but PC users today expect this level of sophistication. And, fortunately, you don't have to create all this code without help. If you check the PC-SIG Library, you'll find quite a few routine collections for most of the popular compilers that make menu bars, pull-down menus, pop-up menus, help screens, and mouse functions easy. There are even several hypertext editors in the library. Don't be afraid to experiment a little, just don't get too creative (remember Rule No. 1).

With a little extra care in designing the interface, you can turn an average program into one that is a joy to use. If your program is unique, and it makes your users happy, you should have plenty of registration checks soon heading your way. ❑

Mike Callahan is a hydrologist for the National Weather Service, who lives in Indiana but doesn't play basketball very well. He enjoys programming and trying out new software whenever he can get his wife and four-year-old daughter off his homebrew AT clone.



Ask Dr. Science is a popular radio show on 150 American Public Radio stations.

It is for people who are tired of intellectual pretentiousness.

Q: A friend of mine told me my hard drive crashed. What does this mean? Why can't I find any dents on my hard drive?

—*Bill Gates, Bust, Colorado*

A: Chances are your friend owns a disk drive body shop and is looking for business. Sure, there are probably some microscopic dents in your hard drive, but so what?

As for the poor thing crashing, that's a $50 word describing a five cent phenomenon. So you lost all your data, big deal. That sort of thing happens all the time in this business. There was no crash, no screeching of tires, no crunching of metal. All the electrons decided to take a nap at once, that's all.

©Duck's Breath Mystery Theatre

If you have a question for Dr. Science, for this column, you can send it in care of Shareware Magazine. For a free Dr. Science catalog of goodies (tee-shirts, pins, books, cassettes, and other sordid duckphenalia), write to: Ask Dr. Science, c/o Duck's Breath Mystery Theatre, P.O. Box 22513, San Francisco, CA 94122.