

Key Tips for User-Centered Design

— published in The X Journal, November/December, 1995

Eric Schaffer, Ph.D., CPE. and John Sorflaten, Ph.D., CPE

We interact with many developers when researching and designing GUI standards. Some of the recurring problems we find can be solved with knowledge of a few expert tips. The following design tips come from our 2-hour video course introduction to GUI design: "How to Design Usable GUIs for Corporate Applications." For information on the course call 1-800-242-4480.

PRINCIPLE	EXPLANATION
Windows	
Use a structure that controls window thrashing and window pile-up.	Anticipate user needs. Size and position windows to be useful immediately when opened. Use a UI architecture that avoids the need for window thrashing (E.g., folders, notebook, and context switch). It costs users time to activate, resize, reposition and orient to the new window. (See our GUI design column in this issue.)
Try to reserve pop-up windows for infrequent use.	A pop-up dialog box provides extra screen "real estate". But reserve it for occasions that occur rarely (5% of the time or less).
Use pop-up windows to keep context rather than to proliferate variety of tasks.	Help users keep track of their work flow with the main window in the background and an associated dialog box in the foreground. Have only one pop-up open at a time, where possible.
Avoid small windows.	Let users see as much context as possible. Limiting context is as bad as having your long-range view of the road reduced by fog.
Avoid frequent pop-ups.	Use an additional primary window instead of numerous pop-ups. Less fragmentation of data lets user see context better and consequently work faster.
Flag users about data in an unopened pop-up dialog box.	On the primary window, provide an "indicator icon" on the pop-up button. The icon informs users whether or not data exists in the pop-up. This saves the user from needlessly opening the pop-up to check for data.
Icons	
Don't expect to save much of user's time with icons.	At best, icons save about 300 milliseconds in speed over a label alone. That's about one third of a second!
Unless developing for expert users, use labels with icons.	Most icons must still be "learned" to be understood. Sidestep the learning curve by adding labels to your icons.
You can save space with an unlabeled toolbar or icon palette, but this requires training users to expert levels.	If you need to save space on labels, and the application is used often enough to permit easy recall of the icon meaning, then use a palette, toolbar, or ribbon.

©1996, Human Factors International, Inc.

Use icons primarily to increase “flash”, but don’t forget the costs.	Icons are a nifty way to enhance the visual attractiveness of an application. But consider the possible costs in increased development time, user training, and lack of keyboard access for activating an icon button.
Reinforce a metaphor with icons.	If your application uses a metaphor, like a paint program, make the controls more concrete and easier to learn with contextually related icons. They increase satisfaction and may reduce training.
Don’t make users work hard when selecting actions.	For a keyboard-oriented application, avoid toolbar icons for frequent actions. Users may find icons hard to select from the keyboard. Function keys give quicker keyboard access and they avoid typing errors associated with Alt and Ctrl modifiers. Also, keep functions positioned on the screen as part of a left-right, top-down task flow.

Mouse

Don’t make users switch between mouse and keyboard frequently.	Fast typists lose up to 8 keystrokes to reach for the mouse. Slow typists only lose about 3 keystrokes. Consider these costs when designing keyboard intensive screens like data entry. Consider using function keys, “hot keys”, or accelerator keys to access buttons for experts using the keyboard.
Design for a mouse when the task calls for occasional updates that use large cursor movements	When data changes are scattered across the screen, a mouse can let the user move quickly to the desired field. Use mouse-oriented widgets such as radio buttons to make the point-and-click selection easier. If space is limited, a drop-down list box accommodates mouse usage as well.
Remember the benefits of a mouse over typing	A mouse lets you draw using trial-and-error! Extend this to non-typing situations such as analog controls for setting valve opening and where approximate input values are fine. Make these systems easy for non-touch-typists.
Recognize practical user limitations for mouse usage	Some users may have difficulty with the eye-hand coordination required for the mouse, or users may be standing--a difficult position for mouse control. Sometimes the environment may not support a mouse (e.g., cluttered work space, corrosive fumes).

Pull-Downs

Don’t hide functions under pull-downs.	Use buttons and function keys for frequent options. Otherwise, new and infrequent users must search through the menu bar for the option they need. (See our GUI Design column in this issue.)
Don’t succumb to cryptic labels.	Pull-downs tend to use short labels that require interpretation and training. Buttons can offer more area for clear labels and larger mouse targets as well!

Avoid the extra motion required by some pull-downs.

Pull-downs require users to pick and then pull down a menu to finally pick the desired option. The user may need to explore the menubar as well. Buttons only need one action: a pick. Buttons save motion! (Selection by accelerator keys helps, but mistyping is easy with an Alt or Ctrl key combination.)

Use pull-downs for system-wide functions that are infrequently used.

Learn them once, use them everywhere. Low frequency use means the user won't be penalized too much for the extra physical work.

Solve screen space problems with pull-downs.

Use pull-downs when your task consumes the screen space for a major object like a text edit window (word processing) or large metaphor (spreadsheet), etc.

Metaphor

Reduce training time by using a metaphor.

A metaphor mimics a familiar object or task. For example, one system used a grocery store metaphor to help users first identify screen items and then save the transaction. Putting an item in a "shopping cart" identified the items. Going through the "checkout counter" saved the transaction.

Don't overreach with a metaphor.

Metaphors can break down. If misused, like a trash can on a desktop, users may be puzzled and require training. If users understand their job, you may not need any special metaphor!

Don't be cute.

In corporate applications, avoid game-like or cute metaphors. They distract and may add to training time. Puns are hard to understand (e.g., an "axe" icon to signify "executing" the program.)

Drag-and-Drop

Make tasks physical with direct manipulation. the system.

Where possible, simplify tasks by letting users physically control This sidesteps training that may be needed for more abstract commands or functions.

Don't use direct manipulation if physical action or concepts become difficult.

Avoid making users work extra hard! For example, don't force users to drag 6 stick figures when they could type "6" faster and more easily.

Check the physical work for drag-and-drop.

Insure that your design provides shortcuts for Insure that your design provides shortcuts for drag-and-drop such as select-to-delete or group-and-drag. Avoid dreary, time consuming drag-and-drop situations like dragging a fax address from one list to the send list. For example, if only one send list exists, then, point-and-click is faster than drag-and-drop! If multiple destinations exist, then drag-and-drop may be best.

Spatial Storage

- Store data by spatial position. Users remember best when associations are made with a familiar location (cf., “Method of Loci” as a memorization technique.) For example, position OK and Cancel buttons and menubar options consistently. This capitalizes on the user’s natural skill in spatial recall of often-used items.
- Avoid “sterile” or “featureless” spaces. A blank file folder doesn’t aid spatial memory very much. For example, give users an on-screen office to help locate a file, document, or application module. A structure such as the “Controlled Task Panel” helps the user perceive relationships between objects. (See our GUI Design column in this issue.)

Presentation Graphics

- Use presentation graphics. Meet the needs of users who must quickly find trends, patterns or relationships in their data. Charts show these better than tables.
- Don’t clutter the screen. For example, avoid 3-D charts when two dimensions are enough to describe the data. (In other words, don’t be misled by 3-D advertising claims made by charting programs.)
- Don’t mislead users. Keep x and y axis measurement scales authentic. Avoid broken scales. Maintain a constant aspect ratio between the x and y axes. Don’t allow “rubber band” changes in aspect ratio.

Large Screens

- Feel good about blank space on the window. Use blank space to group and organize your fields and data. Don’t feel obliged to completely fill the screen with only fields and data. 70% white space is very acceptable. For strongly organized data (e.g., tables) 30% white space is okay.
- Put a complete logical unit of work on the screen. Large screens allow the luxury of giving users the “big picture” of their work. This is better than mentally piecing together the work from different windows or pop-ups.

High Speed

- Provide 1/2 second delay before displaying an error message. Most users get surprised and disoriented when an error message appears instantaneously. Match expectations by adding a delay.

Color

- Don’t commit basic color errors like chromostereopsis (stereo “layers” of color). Avoid letting one color, like red, “float” on another color, like blue. Plus, these colors, in particular, can be hard to read. Adjust the level of grey in the color to solve these problems.

Use color to aid eye scan.	When scanning across columns, if you can't skip every fifth or sixth line, then use a subtle color change between groups of lines. Or, use color to identify particular rows in a list that meets special criteria.
Unless you test users for color vision, design for monochrome, then add color.	8% of males and about one half a percent of females have some form of color blindness. Ensure that your design is meaningful in black and white, then add color to aid discovering relationships and aid scanning for highlighted items.
Don't expect users to notice subtle color differences.	Even users who can see all the colors fail to remember subtle color coding schemes that require memorization of more than 1 or 2 colors. Do not use color as the sole code discriminator. Use color and shape or color and location to reinforce the code.
Use color for aesthetic appeal and to convey meaning.	Users prefer color. Also, remember cultural associations for each color: such as red for stop, green for go, and yellow for caution. Don't fall prey to any unintended meanings. Special industries have different codes. Red, green, and yellow have different meanings in a nuclear power plant, for example.
Complex Keyboard Control	
Don't use shifted key entries.	Pressing two keys at once invites keying errors. Try typing “*. *” ten times in a row without a mistake! Avoid forcing users to type with Alt and Ctrl keys for frequently used functions. F-keys are much better.
Avoid creative key mapping.	While it may appear straightforward to make “&” mean one thing with Alt and another thing with Ctrl, it can be confusing; and most users find it difficult to keep it all straight. They require extensive training and practice to memorize it all.
Use “hot keys” when users are experts.	When users interact with an application often, “hot keys” (with Alt and Ctrl) are worth the learning effort by speeding up the selection of menu and button options. Help make them easy to remember with mnemonics, like Alt/H for Help.
Design cursor movement.	Look for instances of speeding cursor control, such as mapping the down arrow key to navigate to the field below the current field. The up arrow would also be mapped to go up a line. You may need to modify your widgets to allow this.