

# Mousing Around: Tyrannosaurus Rodentia Plasticae

— published in *The X Journal*, May, 1996



Eric Schaffer, Ph.D., CUA, CPE, is Founder and CEO of Human Factors International, Inc. (HFI). He teaches, consults, and speaks on corporate and governmental Web and GUI interface design issues.



John Sorflaten, Ph.D., CUA, CPE, teaches and consults as a Project Director at HFI. With Eric, he initiated a usability curriculum at a local university in his home town of Fairfield, IA.

©1996, Human Factors International, Inc.

The following “game plan” missed proper shredding. The crumpled, top-secret paper fragment appeared in a plastic trash bag with brochures and paraphernalia linking it to a popular late-20th century CD-ROM game publisher. A small team of cultural anthropologists uncovered the document in the year 2056 while researching a old California landfill site (formerly a “garbage dump”). They were researching computer history, by the way.

“Story line, \$5 million budget MUD (Multiuser Dungeons type game). For release December 1996.

“With the advent of the mouse input device (Rodentia Plasticae), life on planet earth forever changed. Homo sapiens no longer depended solely on frail, fatigue-prone finger tips to control their savage computers. Limited memories no longer had to regurgitate long commands to feed their rapacious screen prompts. The advent of point-and-click heralded the dawn of a new age. Mousing around brought profit.

“However, against this new, bright sun stood one lone, shadowed, bearded figured. They called him “OB-1.” (Although it sounded like Obi-Wan, it signified “the Objective One”.) Tightly wrapped in robe and hood, this prophet (some said) dared speak out ‘Tyrannasaurus,’ against the revered Rodentia Plasticae. History later recorded that OB-1 alone was the first human to identify this most subtle guise of the eternal foe: cryptodesign.”

**THE MOUSE IN ITS PLACE: FUZZY WUZZY** Consider the icon. Some icons were meant to be moved around on the screen (see our GUI column for the January/February issue). We can drag the document icon onto the folder icon. We drop one folder on top another folder. We shove documents and folders into the trash or recycle bin. Likewise, take the “window” object. Yes, take it—and move it. Or point and click on a partially hidden window and bring it forward into full view. We interact with these objects almost as we do with paper and folders on our desks. These graphical interactions let us organize our work into hierarchical schemes unburdened with verbal commands. We eliminate learning new paradigms by using these metaphors. We take action non-verbally through direct manipulation. The mouse truly gives a fuzzy wuzzy feeling to computers.

**CRYPTOMOUSE SNEAK ATTACK** When preparing a Macintosh version of our GUI seminar several years ago, we explored some Mac appli-

cations. We found an anomaly: we could not TAB to a radio button (“option button”) or a check box! Afraid this may have been idiosyncratic to the application, we called an Apple human factors staff member. He laughed, and said “yes, we know that’s a problem. We need to enter data too. The Mac can’t tab to radio buttons or check boxes until we change some things at the core of the operating system. That’ll take a major revision.” Conclusion—the Mac is “mouse-centric.” Although we did get this recommendation: use “type select,” i.e., underscore one of the letters in the radio button label to create an accelerator key. Then, users could select the option with “Command key + letter key.” However, this solution lacks compelling motivation. Chorded key strokes (two at once) bring high error rates and slow down input speed when compared to pressing the tab key.

While we may laugh at the Mac situation, we find many applications in other popular operating systems remain mouse-centric when they shouldn’t. This fault arises from the stealthy work of cryptodesign. Recall, these are design ideas that work for certain situations, but get misapplied when thoughtlessly applied to tasks that are different than the original. We’ll cover the issues below in our “VIMM” model.

**SNEAKIER CRYPTO ATTACK BY SECONDARY MOUSE BUTTON** Now that Win95 has jumped into the future with the right mouse button (OS/2, OSF, etc. were there all along), we face grave challenges, i.e., developers and product managers may feel even more compelled to embrace mouse-centric design in order to get hypothesized benefits from the “other” mouse button. This bifurcation of mouse features appears to have some benefits. Pointing to a object and selecting the secondary mouse button brings up a context-sensitive menu. In the word processor used to type this article, a right-mouse menu lists three options for setting the

typing parameters for subsequent text. We can change the font, set paragraph style, or initiate bulleting and numbering schemes. If text were highlighted, we could also cut or copy it. Wow.

Contextually related commands are wonderful. However, the secondary mouse button can easily be misemployed. Note that the word processor also provided the same functions in a more traditional way (pull-downs in our word processor, or buttons in other applications). Designers misuse the right-mouse button when they fail to provide navigation or functions in these other, more obvious ways.

Problems arise because this “new” pop-up menu, like pull-downs, is hidden. Avoid using it as the sole source for navigation or functions. Reserve it only for short-cut commands that experts would like to use. Frequent users accept the effort required to memorize the functions and recall their availability. Their high volume workload makes the effort pay off. Meanwhile, casual users and new users must have access to commands through more traditional, visible means such as buttons and menubar options. We suspect that after another 5 to 10 years of computer experience in the workforce, this recommendation can change. Currently, however, we must remain conservative while still in the early stages of computer-literacy.

**THE TASK IS THE THING** As in each of our columns, we resurrect soul-design by analysis of the various kinds of work involved: Visual, Intellectual, Memory, and Motor (the VIMM model). Let’s look at how Rodentia Plasticae can be misused and what you can do to decrypt its influence.

### Reduce Visual Work

**Issue:** Finding a window to click on requires visual work. The window title or other clues to distinguish the desired window may be hidden. This problem arises when the designer depends on the mouse as a

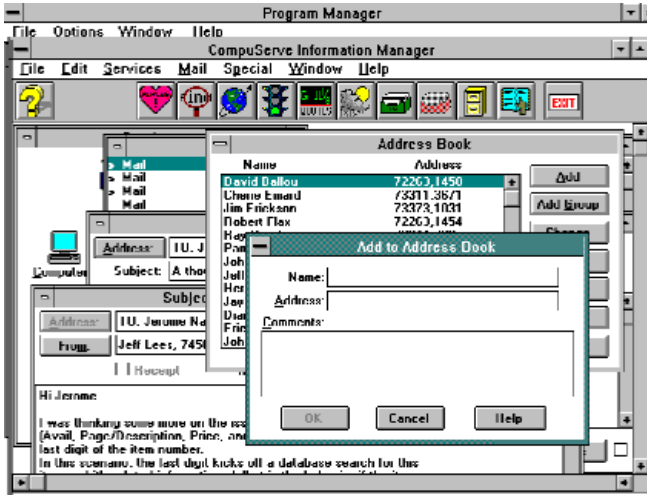


Fig 1. The mouse fails to maintain productivity and ease of use when designers use multiple free-form windows.

cryptonavigational tool and thus tolerates the burden of “window thrashing” (see Figure 1).

**Solution:** Design applications to greatly reduce or eliminate the proliferation of windows. For example, as discussed in previous issues (September/October and November/December, 1995), the folder metaphor allows users to open one tab while simultaneously closing another tab. This approach does not exclude using the mouse. But it reduces the tyranny of mousing around! Note that keyboard access is quite easy if we label the tab with a function key or use an accelerator key (Alt + letter). (The function key is less error-prone than accelerators.)

**The Method of Loci:** There are many other ways to reduce the visual work in searching for “clickable objects”—be they windows or buttons. All the solutions depend on maintaining a fixed spatial location for the pointer target. Tabs remain in a fixed location. Another example involves placement of the OK and Cancel buttons in locations that remain consistent throughout an application or your user environment.

Several rules can be invoked. First, these buttons should consistently appear either at the bottom of

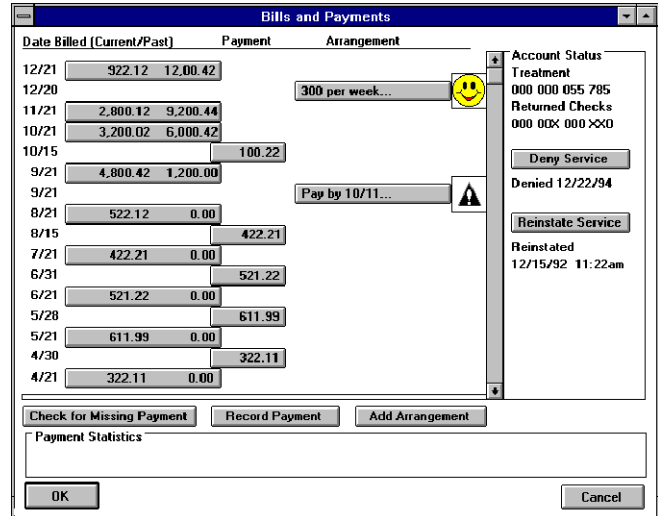


Fig 2. In this customer service window, the rows and column names provide a consistent schema for locating mouse pointer targets. Also, the default (OK) and Cancel buttons remain in the lower left and right corners in all the windows of the application. In both cases, the fixed layout (“method of loci”) eliminates much of the visual work in determining click targets.

the screen or on the right across all windows in an application and not switch from window to window. (The bottom is usually best, since users finish “reading” the screen at the bottom.) Second, within the bottom or side, the buttons should always have the same physical positions, and not “move” from window to window as other options are added. For example, [OK] [Cancel] might appear together in the middle of the bottom row in one screen. However, this layout becomes spatially inconsistent with another window that may have the additional options, resulting in a row like [OK] [Search] [Add] [Delete All] [Cancel]. To eliminate such “wandering buttons” we recommend placing the OK or default button on the far left and the Cancel button on the far right. Other buttons can come and go, but these elements remain consistent. Why do this? It’s based on the “method of loci” as follows.

Note that you probably prefer keeping items in fixed locations (in your desk, in your home). This

regularity of habit lets you devote your mind to other, more interesting activities. We particularly experience this when driving and talking. If the rules of the road suddenly changed, such as “drive on the left side for this next mile,” you would reduce your conversation drastically for a while. Using this penchant for fixed location, you can memorize a list of groceries (or reasons for salary increases) quite easily. The 5th century BC Greek poet Simonides taught his students to associate each item in a list with the pieces of furniture located in their bedroom, living room, or elsewhere. It was then easy to recall the list in sequence. The student only had to recall the clockwise position of the furniture located in the room! Thus, the technique is called the “method of loci.” and yes, we tap into the same mental process when we fix the position of the default button (e.g., OK) and the Cancel button. We can extend this design approach to more challenging tasks, such as accessing numerous historical records for a telephone customer (see Figure 3).

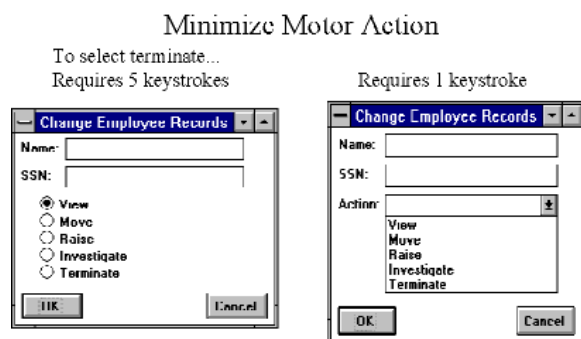


Fig 3. On the left, the choice of radio buttons fools the user into reaching for the mouse. Note that the user COULD figure out to use the down arrow. But that costs extra keystrokes. On the right, use of the drop down list box allows a single keystroke to select the item.

### Reduce Memory Work

**Issue:** Jumping back and forth between keyboard and mouse costs time and effort. People who type rapidly will lose more keystrokes than slow typ-

ists. We estimate it costs three to eight keystrokes for each jump. Cryptoclick design will fool users into reaching for the mouse when they should stay on the keyboard to maintain efficient input speed (see the left side of Figure 3). Users must do extra memory work to remember how to keep the fingers on the keyboard (use down arrow to select “terminate”). Even then, users get penalized with extra keystrokes compared to the better solution, next.

**Solution:** The objects used for a screen design must match the task. If the task requires fingers to be on the keyboard to enter characters, then choose objects to support keyboard entry like the drop down list box (see Figure 3, above). Further reduce memory requirements by using list boxes that are always “open.” The user can see the choices immediately. Use drop down lists only when space on the window is extremely tight.

Note that list boxes should support “autocomplete” by immediately displaying the best match to the keystrokes entered (as found in Quicken or Lotus cc:mail). Windows 3.1 allowed a match only on the first letter. Better yet, Win95 allows a match on all keystrokes typed within the time limit. Research shows that autocomplete speeds data entry by 100% or more for expert users.

Some windows might require both keyboard and mouse usage. For example, a user may type in search criteria, then use the mouse to scroll the long list of found items and select one. In such cases, design the task flow so that keyboard objects are grouped together, resulting in only one “jump” to the mouse. By the way, we can eliminate even more mouse clicks in search-and-list functions. If the search results in only one item, consider displaying that record immediately, bypassing display of the list box and subsequent mouse click (see Figure 4).

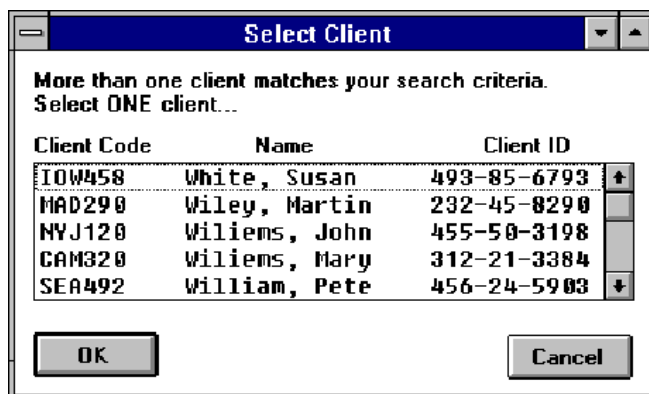


Fig 4. Avoid requiring the user to mouse-click on a list that displays only one item after a search. Consider showing the item record immediately. Display the list only when more than one item results from the search.

### Reduce Intellectual Work

**Issue:** Although using the mouse to select a menu option appears easy (after all, it's Rodentia Plasticae), we can still create mind-bending problems for the user (see the left side of Figure 5). How sort out the many options, even though one click will do it?

**Solution:** Figure out how to “unpack” the choices into “bite-size” chunks. We estimate that 25% of our work as screen designers involves unpacking complex logic into simple steps. Note that unpacked work allows the user to take an action after reading a short instruction. (Win95 wizards do this well.) Yes, it's an extra mouse click, but certainly less intellectual work (see the right side of Figure 5).

### Reduce Motor Work

**Issue:** Motor work is physical work, like moving the mouse pointer to a target or reaching over to move the mouse in the first place. How can we efficiently reduce such work “expenses” for the corporation and the user?

**Solution:** Reduction of physical work has a micro- and a macrosolution. The microsolution involves

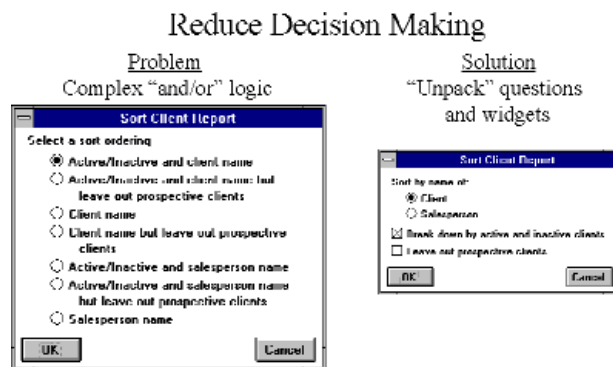


Fig 5. In seeking to minimize mouse clicks, do not make the intellectual work overwhelming. Fit the mouse clicks to the work, not the work to the mouse click. Avoid Tyrannosaurus rodentia plasticae.

knowledge of “psychophysical” findings such as given by Fitt's law. Professor Fitt studied the mathematical relation between the time required for a subject to move a stylus to a target and the accuracy to hitting the target. His research found that the movement time decreased when the target was larger (see Figure 6). Doubling the width of the target effectively cut the time in half! Our lesson is clear. Bigger buttons are faster! We can add here that larger targets also provide more space for clear button labels. Many commercial applications use short button labels and we may feel obliged to imitate that style. However, in corporate applications, users interact with many more screens, and have less time to “learn” the cryptic button labels. Thus, you should feel comfortable in creating longer button labels if it speeds learning and reduces errors. For example, a button that reads “Employee charitable contributions last month” could be far more productive than “Contributions”.

The macrosolution to reducing motor work requires standardization. Designers who work in a group on a given application must coordinate their design choices in order to provide consistency for their users. We recommend use of “widget selection rules” to guide the group (see Figure 6). Note that the rules account for keyboard vs. mouse

If the user operation will be...	And access frequency is...	And space is...	Then use this widget	Example
Keyboard	High	→	Combo box	Phone <input type="text" value="(416) 348-6776"/> (416) 348-6776 (515) 472-4480 (212) 544-6235 (617) 863-9023
	Low	Adequate		
Tight		Drop down combo or list box*	Phone <input type="text" value=""/>	
Mouse	Very low	Tight	Drop down list box.* Use spin button if a clear up/down sequence (e.g., length of mortgage)	Drop down list box <input type="text" value="(416) 348-6776"/> (416) 348-6776 (515) 472-4480 (212) 544-6235 (617) 863-9023  Spin button Mortgage (yrs.) <input type="text" value="5"/>
			Adequate	Radio button
	Low - High	→		

Fig 6. A sample page from our “Widget Selection Rules” used in our GUI standards. It coordinates the work of developers and provide interface consistency for users.

tasks, available screen space, and list length, among other parameters. We include a complete set of custom selection rules in all the standards we create for clients. Another plus--the rules limit the developers to a subset of all widgets. This means users get a lot of practice on the subset. Try to save users the challenge of constantly learning new widgets and confronting strange-but-true interactions of widgets with unique behaviors (see Figure 7).

**CONCLUSIONS** Using the mouse appears simple. Even kids can use it. In fact, the apparent simplicity has been the source of many of the problems

## Widget Interaction Problems

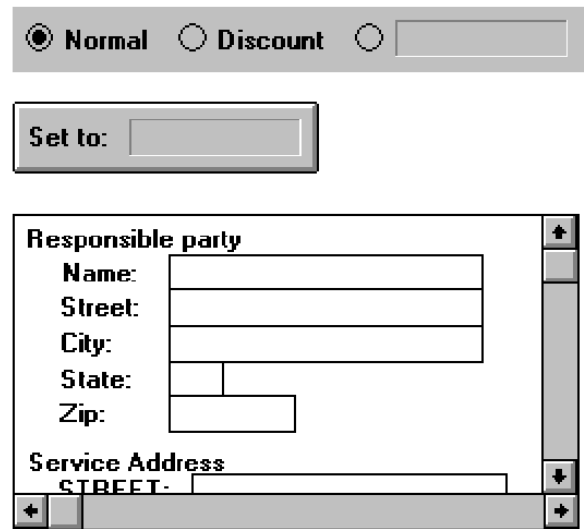


Fig 7. A sample of strange siamese widgets we have seen. Although each may have solved an apparent design problem, the agony of interpreting them afresh is too much to inflict on innocent users. Tyrannosaurus run amok.

we pointed out in the column. Designers forget that cryptofactors can influence the work required for mouse usage such as extra visual search, forgetting to use keyboard methods, and overwhelming user with intellectual demands. These burdens add the Tyrannosaurus to the innocent Rodentia Plasticae.

By the way, the OB-1 game turned out to be the all-time best selling CD-ROM during the “age of information.” That age ended in 2006 with the advent of quantum computing and the accompanying solutions to problems of sustenance, political instability, and general craziness. Then we got the “age of wisdom.” People found they could lead much more enjoyable lives than sitting in front of a terminal or even talking to one.