

# Window Layout: Cures for Cryptovision

— published in *The X Journal*, September/October, 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.

Among the incredible range of neuropsychological debilities, few are more puzzling than “prosopagnosia.” This strange malady removes the ability to “recognize” a familiar face! Yes, the victim still sees quite well. However, no dawn of recognition occurs when seeing the face of one’s spouse, child, or friend. The victim must examine the clothing and remember the voice to identify loved ones standing before them. The victim sees, but doesn’t understand. We call this “living with cryptovision.” No cure exists.

In screen design, your users may also experience “cryptovision.” They see the screen, but fail to recognize any meaningful task. Luckily, this illness seems to appear only on certain screens, typically those that other people designed. Never your screens! This symptom reflects the wily style of the eternal foe, cryptodesign. It strikes when we least expect. Let’s investigate the clinical prescription for combating this insidious neuro-cryptoagent, another enemy in our war against GUI’s from hell. Recall, cryptodesign manifests itself when a design that solves one problem gets misapplied to other, quite different circumstances.

**HOW TO READ A RECTANGLE** Different rectangles demand different viewing techniques. Imagine you’re at the Louvre and the rectangle on the wall contains a smiling female head-and-shoulders pose backed by mysterious, cragged and hazed rock formations. Notice how your gaze begins at a graphically “dominant” shape or color. Then it follows a path almost “prescribed” by the artist. Your eye traverses the mouth, the brow, the eyes, then back to the mouth again. Over and over. We’re charmed, enchanted, and transported by the ineffable smile of La Gioconda.

On the other hand, if this rectangle had textual commentary covering some 30% of the area, where would your eye begin? Would you begin in the middle, where you suspect an important or a bolded and italicized word resides? No. Most of us would exercise our convention of “reading” the text and thus begin at the leftmost side of the topmost line. (Readers of Arabic, Hebrew, Chinese, and Japanese text would follow their own directional conventions in their native language.) In conclusion, we use a different strategy to understand rectangles containing words compare to rectangles containing just pictures. Most screens display text. Therefore, we should expect users to “read” those text-oriented screens like we read a book.

The crypto-wounded designer has forgotten this convention and expects users to invent a task flow on the fly. The crypto-designer

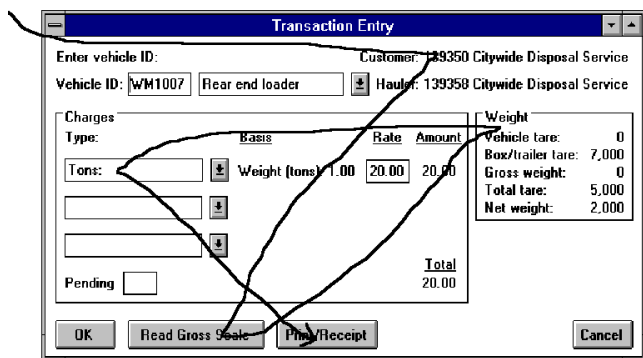


Figure 1. Cryptovision at work. Here the design hides the task flow. The user must work hard to understand the expected task sequence. The user must infer the task flow based on training or experience.

failed to build clear task flow into the screen design. See Figure 1 for an example. In defeating insidious cryptovision, the soul-designer quietly enlists our habit of reading. The soul designer creates meaning in the screen layout by enlisting our conventional reading pattern. See Figure 2.

Now let's see other methods by which cryptovision agents infiltrate screen designs and how you can mount a soul filled counterattack.

**CRYPTOVISION SNEAK ATTACK** Occasionally a designer will use a left-right, top-down pattern for the task flow, yet fail to tell the user what to do.

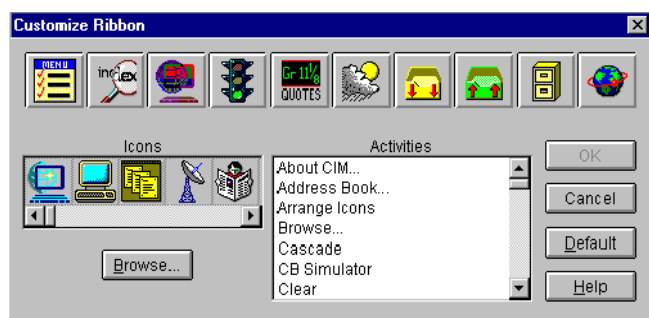


Figure 3. This window allows you to change your Compu-serve tool bar icons. Training is NOT an option for Compu-serve. Instructions would help. Apparently users can assign any command to any icon. Could you? Although the task follows the left-right, top-down model, we still get cryptovision meaninglessness. (PS, we never figured out what value you'd get from making the Mail Inbox perform the Exit command. But that's a topic for another column.)

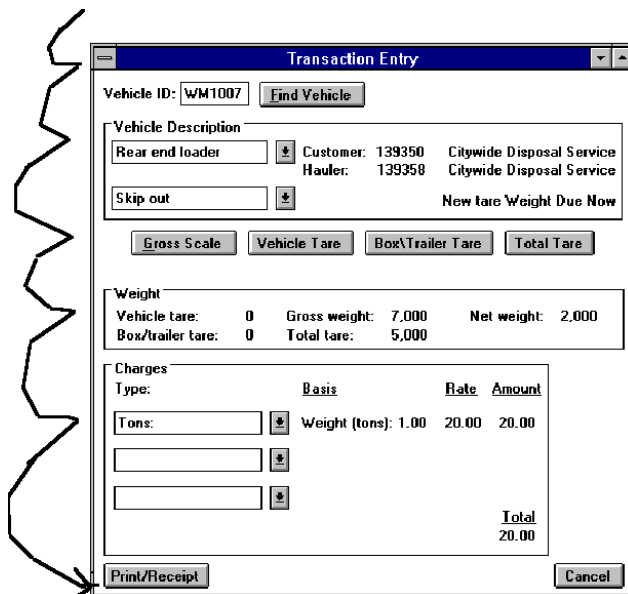


Figure 2. The task flow is implied by the leftright, top-down sequence of screen objects. Meaning emerges from the sequence with an implied then between each line. For example, “consider this first” then “consider this item next” then “consider this following item”, etc.

The design in Figure 3 causes confusion by withholding important instruction from the user. Check it out on Compu-serve!

Some developers think that instructions will insult users and clutter the screen for the expert users, but casual, infrequent users fail to remember what to do, thus need “just in time training” on the screen. Meanwhile, instructions never slow down experts. Experts easily ignore them, just like we ignore highway signs when driving on a familiar road.

In the Compu-serve example, the designer might reply that the Help button is nearby, with ready support. However, why call an ambulance, when only a band-aid is needed? The lesson is that layout can be correct but still lack meaning (the cryptovision thing). Layout can benefit from disciplined use of short, well-place instructions. Call it writing from the soul.

Discover the weak points in your design by asking potential users to think out loud as they do a task

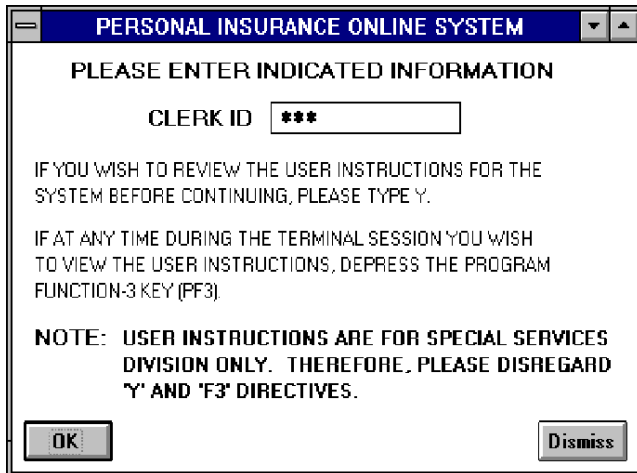


Figure 4. The warning “note” fails to capture the users attention even though the designer put it in bold. When tempted to highlight or emphasize text to get user attention, you probably lack the right layout sequence. Sequence guidance and other instructions in the task flow to eliminate the need for emphasis.

using paper printouts of your screen designs. This is called a *protocol simulation test*. If a user gets stuck, ask what they’re thinking. Don’t get defensive. If they don’t know what to do, and user training is not in the cards, then you can only blame the design. Get soul.

**THE TASK IS THE THING** What regimen can save users from attacks of misunderstanding arising from bad layout? First, we must understand what varieties of “work” designers create for users. Then we can see how to reduce that work. Once again we’ll take recourse to our familiar VIMM model (Visual, Intellectual, Memory, and Motor work) to construct soul design solutions.

### Reduce Visual Work

**Issue:** We find that some developers fail to recognize that their screens merit the text approach to reading the rectangle. They design with the artistic approach. They think that by making some area in the screen appear dominant, the user will automatically look there and subsequently figure out the task flow. Crypto-designers erroneously hope that

highlighting, blinking, and textual emphasis like bold or italics can override a poorly sequenced layout (see Figure 4). Here the “note” at the bottom of the screen really serves as a warning. It’s quite important. Although bold text may make it stand out, most users still fail to read the note prior to entering the forbidden Y value.

**Solution:** Sequence all your fields, menu options, and other screen elements to match a left-right, top-down task flow. If the task has no clear sequence, group items by subject or function. Lay out your screen so that important items like warnings (as in Figure 4) and critical guidance show up first. In other words follow the logical task flow, with warnings preceding the action.

Some corollary soul solutions to visual work in layout design:

- Avoid putting important items in the “status bar” typically used at the bottom of the window. The status bar is spatially too far removed from the other screen elements to attract the user’s attention for special cases. Important messages deserve special prominence such as a dialog box or placement at the top of the window. These include error and warning messages, system messages and informational messages. However, you can display field definitions in the status bar without compromising usability.
- Put field labels to the left of the field. Avoid placing the labels above the field or below it because it contradicts the normal habit of left-right reading.

### Reduce Memory Work

**Issue:** Cryptolayout lapses can increase memory demands. For example, the user may see a start date and an end date side-by-side. Can you compare the days, months, and years easily? (See Figure 5.) The eye cannot take both date elements

## Label and Field Justification

Follow justification principles

- Position edit field labels to **left**
- Position radio button and check box labels to **right**
- Position list box labels on **top**

Not like this:

EFF DATE	END DATE
12/31/89	10/31/92

What about this?

WSOI	DT	146029	42	1991	D
Tran	Co	WoNbr	Dv	IssYr	M

Figure 5. How compare dates easily? Here a single glance fails to get both dates at once. You must work hard to remember one date while reading the other. Solution: place the latter date below the earlier one. Next: How position labels? Solution: put labels on the left. But remember to reduce crypto memory demands in special cases. Match your screen layout to paper forms that may have labels below the line. Then users won't need to remember how to translate field names and sequences between paper and screen.

within the same glance. You must remember one and mentally compare.

**Solution:** When the user benefits from comparing two dates, then position one date above the other. The user can easily compare year, month, and day in a single glance, given their spatial proximity.

Some corollary solutions to memory demands in layout design:

- Avoid covering fields with their associated error messages or detail dialog boxes. Users won't remember what they did. To see the field, position dialog boxes below and to the right of the field. However, a single standard position for these dialog boxes won't work because fields can be anywhere on the screen. Therefore, build in some "smarts" to your application. Position the dialog according to field position on the screen. Use this algorithm: First choice: below and to the right of the field; second choice, to the

right; third, below the field; fourth, above and to the right; fifth, above the field; sixth, to the left of the field.

- Avoid slavish crypto-placement of field labels to the left of the field. (Although, normally we should place the label to the left of the field.) Consider the user's task! For example, exceptions occur when users read a paper form for data entry and the paper form has the label above or below the line. Ideally, the screen should mimic the form layout. This minimizes user memory requirements for mentally "translating" between the form and the screen layout (see Figure 5).
- Avoid putting focus on protected fields. These fields do not allow data entry, or they only display data. Don't let your user think they could enter data with a crypto-illusion data entry field. Dim (deactivate) your protected data entry fields. Meanwhile, put display-only data on a plain background, generally the same color as the window background. In rare cases when you need to draw special attention to the displayed data, use a rectangle around the data, but make the color of the rectangle background the same color as the window background. Don't color it like an input field.

### Reduce Intellectual Work

**Issue:** Let's assume that the screen displays a good task flow sequence. However, fields and controls may not be clearly grouped. Also, they may not be identified by some unifying concept (e.g., there is no grouping header). Plus, fields may lack left justification. It takes extra time to figure out the tasks amidst the clutter. The user must work to locate a particular field (see left side of Figure 6).

**Solution:** Align field labels on the left side. Avoid justifying field labels on the right, typically on the colon. This creates a "ragged left" margin that cre-

## Left-Justify Edit Field Labels

Problem: Poor justification

Solution: Left-justified fields

Figure 6. Avoid right justification on the colon. Left justification looks more organized. Also, indenting labels under a grouping header helps users in finding the desired field on a screen. Find the “1099\$” field in each illustration. Which was easier?

ates crypto-problems. First, the ragged left margin creates a subjective sense of clutter, according to rigorous usability research. Second, it’s hard to discern a grouping header when the designer fails to evenly indent and align the labels below the header (see right side of Figure 6). Some designers stricken with cryptovision think right justification makes it easier to scan a set of entry fields. For an expert-only population will be true because experts can ignore the field labels; they memorize the screen. But soul design recognizes that casual, infrequent, and new users need to make sense of the screen first. They need clear visual cues that create intellectual understanding for them.

Here are some corollary solutions to intellectual demands:

- Left-align text edit fields as much as possible. It’s all right to tolerate four to seven extra spaces between the label and the rectangle to get the alignment. However, you need not be compulsive and align absolutely every field! The example on the right side of Figure 6 is a good balance of alignment and variation. The variation reinforces the grouping of fields. In summary, pull the text edit field towards the label when the spacing becomes extreme. Meanwhile, pick up

Figure 7. Avoid run-on fields. The eye fails to “read” the bottom items in a single glance. If you can’t use a grouping header, at least put a blank line every 5 or 6 lines.

alignments from items higher in the screen whenever possible.

- Place a blank line every five or six rows of fields. This avoids visual “run-on” that fatigues the eye. Physiological research shows that a typical user can handle five or six lines of text in one glance. (This constitutes the preferred maximum of 5 degrees visual angle.) Consider Figure 7. Notice that even though the fields are aligned and grouped, the abundance of run-on fields slows comprehension. You can logically put a blank line above the Room rate and the Phone fields for improved grouping.
- When you need a set of nested menu options to meet the user’s task requirement, consider the classical hierarchical menu. This layout puts two levels of menu on the same screen as shown in Figure 8. Research shows that the single screen with many options is faster and more easily understood than nested menus. Sequence the groups and option by task flow as much as possible. People get to know their job from the menu groups and sequence. Otherwise, group by

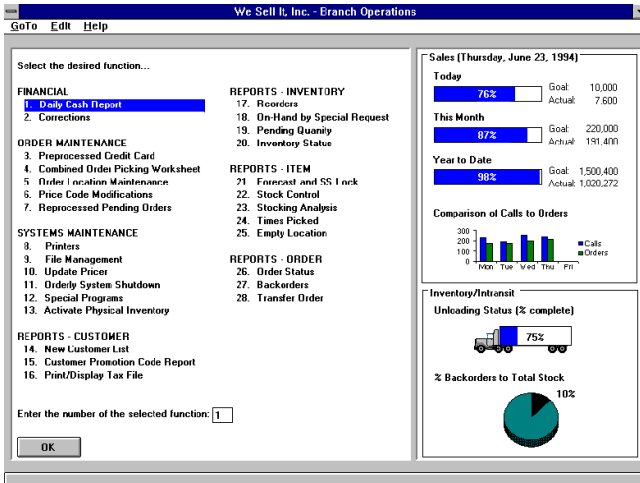


Figure 8. Just because you’re doing GUI design, don’t eliminate the classical hierarchical menu for particular situations. If you have modal task flows that always lead back to a “hub”-use the hierarchical menu. Here’s a GUI version with charts and hot spots for mouse selection. For best comprehension, keep it to 6 groups or less, no more than 10 items in a single group, with a maximum of 18-24 items total.

subject or function. Avoid alphabetical sequences. Users may have a different understanding of which option name to look for compared to yours. Should they look under *N for New Employee* or *H for Hire Employee*?

**Reduce Motor Work**

**Issue:** Have you ever scanned a 160 column print-out and jumped a line? How can you avoid clicking on the wrong line in a scrolled list with several columns? Look at Figure 9. Locate the phone number of Abraham Edwards. Did you appreciate the blank line?

**Solution:** If a list requires that the user scan between columns, add a blank line after every five or six lines. The blank line serves as a ruler to prevent the eye from accidentally jumping up or down a line. Alternative soul solutions include use of a thin hairline every five or six lines, or you may alter the shade of the background every five

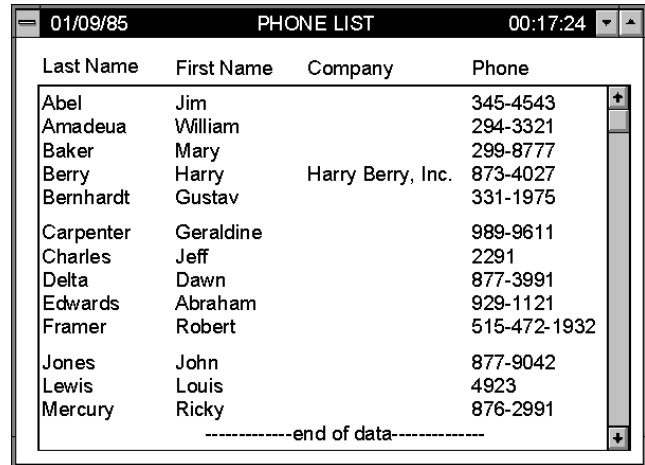


Figure 9. When the user must scan across several columns, place a blank line every 5 or 6 lines to prevent jumping between rows. Be consistent in the number of rows in a group. Also, reduce scanning requirements by putting the most frequently used columns close together, if it’s logical. In this example, consider putting the phone numbers closer to the name if you know there will be a lot of empty space in the list.

or six lines. Use a consistent number of lines to define a group.

Some corollary solutions to motor demands in screen layout:

- Group command buttons to simplify target identification. Buttons with familiar functions such as OK, Help, and Cancel should appear in a consistent position across all windows. (We recommend the bottom left of the window for OK and bottom right for Help and Cancel.) Users use their visual-kinesthetic memory to locate those buttons rapidly. This frees attention and targeting time for the new, task specific button. See figure 10.
- When a screen has many fields, speed up keyboard navigation with “super-tab” jumps between groups. Make the groups obvious. White space makes grouping clear. At least 70% white space is best. If data is highly organized as in a table, you can get by with 30% white space.

If you need to indicate columns more clearly, consider using a single vertical line between them. If you need to clarify the horizontal sections of your screen try to get by with a single horizontal line. If these approaches don't work, the final backup method is to use a frame or group box. The extra lines in the frame add some clutter, but can aid in determining the super-tab group. Standardize on the super-tab key for all your corporate applications. Note that OS/2 uses the regular Tab key as the super-tab and arrow keys for the regular tabs. This forces the user to reach over for the arrow keys to navigate within a group. This is especially awkward since navigation within groups (and to the next group) is far more common.

**WRAPUP AND CONCLUSIONS** How avoid “prosopagnosia” on your interface when your users stare at your creative layout oeuvre? Remember that you can test your screens for that moment of layout meaninglessness. We mentioned the protocol simulation test. Create a sequence of screens that contain a given task, and use print outs for starters. Find people with the job skills and system knowledge you expect your users will have. Then, working with one subject at a time, ask them to think out loud as they complete the work (or struggle with it). Don't chicken out and give them hints. Let them talk and experiment. See how they

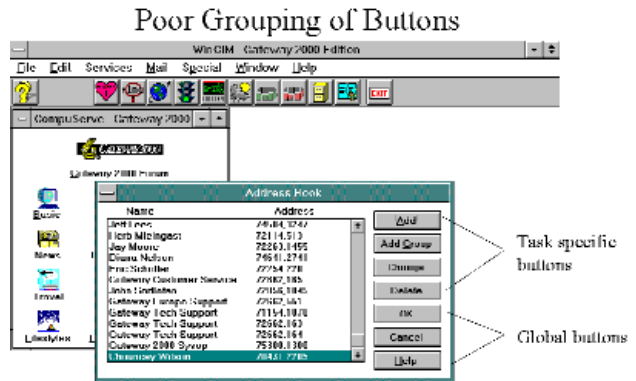


Figure 10. Keep OK, Help, and Cancel in familiar positions throughout your application. Segregate the task buttons to clarify the task groups and minimize hitting the wrong button.

see the screen. Tell them how to move on when they've exhausted their patience. Get through the whole task, if you can. Then you can revise it based on those insights.

Another technique involves placing your layout handiwork on the wall. Put it up where potential users or other knowledgeable friends will see it. Put post-it paper nearby so they can stick comments on your work! You'll find out from your colleagues where the cryptovision hit them. Each of these techniques solves the problem of getting too close to your work. Objectivity is a precious commodity. We have to work for it.