TrainingNews
Volume VII, Number 3 November 1985

THE CBT CRAFTSMAN

Copyright 1985
Weingarten Publications, Inc.
Reprinted with permission.

I Do Windows

There's more than one way to get lots of information on a small screen. Try opening a window.

Jesse M. Heines

Sooner or later, every CBT developer faces the problem of displaying a large amount of information on a small computer screen. This problem is particularly acute if you want to mix color graphics with text on the same screen. To do this on the ubiquitous IBM PC equipped with the standard IBM color/graphics card, you must work in "medium resolution mode" (320 pixels horizontally by 200 vertically), which provides 25 lines of only 40 characters each. (The 40 character restriction can be circumvented by defining your own, narrower character set as described in my September column.)

Now, I have often stated in my CBT design seminars that if you can't say what you want on 25 40-character lines, perhaps you are trying to say too much. But this isn't always true. There are numerous cases in which a graphic might take up half or even more of the display, leaving precious little space for explanatory text. If you try to include running heads to help students maintain their orientation in the course, you will surely find that the screen's the limit.

One way to solve these problems is to use more sophisticated graphics cards that provide higher resolution. This approach can add considerable cost to the system: a TecMar high resolution display card that provides 16 text *and* graphic colors with 640 × 400 resolution on the IBM PC retails for about the same cost as a basic IBM PC itself. These cards also often require the purchase of more expensive monitors to display the higher resolution. In addition, if you develop a CBT program that requires the student to have a system with such expensive additions and expect to sell your program on a retail basis, you will severely limit your market.

A much less drastic measure is to break large displays into two or more screens. In his book, ***Learning with Computers***, Alfred Bork points out that unlike printed media, "white space" on computer screens is "free." That is, while printing an amount of information on two pages of a book costs the printer twice as much as printing the same information on one page, displaying information on two separate screens in a CBT program costs no more than displaying it on one screen. (To be completely accurate, using two screens with some CBT systems might require a bit more disk space, but this isn't usually a problem.)

This multi-screen technique has three drawbacks. First, for logical reasons, some sets of information really have to be displayed on the same screen. Second, for aesthetic reasons, using two displays when you really want one tends to give your material a choppy, elementary-school feel. Third, for instructional reasons, the proliferation of multiple screens that students page through by pressing RETURN yields the boring, noninteractive CBT that I have frequently argued against in this column.

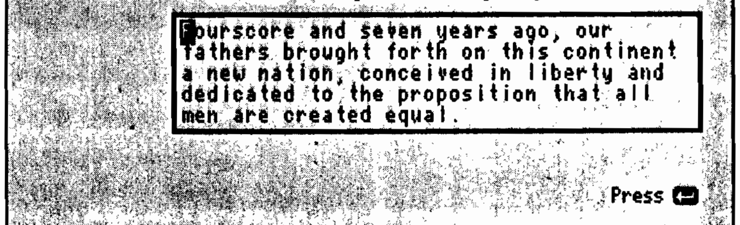
So what's to be done? The first step is to divide the screen into a number of different functional areas, one for graphics, another for explanatory text, a third for error messages, a fourth for orientation information, and the like. Then, if you must use two screens to display your explanatory text, you can go from one screen to another changing only the text area, leaving the graphic and other areas intact to maintain continuity.

Now, I don't believe it is necessary to keep the sizes of your functional areas the same throughout an entire course. Some graphics may be small, allowing large text areas, while other graphics may be large, encroaching upon other areas. As a rule of thumb, I try to keep the sizes of my functional areas consistent throughout a single module (approximately 30-60 minutes of instruction), but allow them to vary considerably in size from module to module. I find this flexibility necessary because I often change my instructional strategy from one module to the next, both to fit changes in the subject matter and to add variety to the interactions. Note, however, that small changes in the sizes of functional areas within a single module can usually be made without the student even noticing. If you key the areas in some way, using colored borders or varying fonts, the student will be able to identify each area easily even when the size varies considerably.

Another technique is to overlay displays using *windows*. Windowing has long been used in graphics labs to expand screen horizons, but has only recently found its way into the realm of CBT. I have seen the term window used incorrectly by a number of vendors, however, so it is important that I clarify my use of the term. Some CBT systems allow you to define an area on the screen in which subsequent text or graphics will be displayed. They allow you to erase this area independently from the rest of the screen and write into it without affecting other screen areas. Technically, an area with these features is not a window; it is what Alfred Bork has called a *viewport*.

Viewports are extremely valuable screen control devices, particularly when students are asked to enter a sizeable amount of information. By restricting input to a viewport, you can be sure that students do not destroy your screen setup, no matter how lengthy their responses. The key characteristic of viewports, as far as this discussion is concerned, is that they obliterate whatever was in their space on the screen before they were displayed. When you finish using a viewport and tell the computer to delete its definition, the area that it occupied becomes blank.

True windows act like viewports, but


Writing the First Draft — Word Processor

A MINI WORD PROCESSOR
Moving the Cursor

All functions in our word processor depend on the position of the *cursor*, a small rectangular block (█) that marks a position in the text. For example, the cursor below is marking the first character in the text.

♦ Use the four arrow keys to move the cursor around.
Press ⏎ when you are ready to go on.

```
Fourscore and seven years ago, our
fathers brought forth on this continent
a new nation, conceived in liberty and
dedicated to the proposition that all
men are created equal.
```

Press ⏎

(Copyright 1985 by KJ Software, Inc., reproduced with permission)

Figure 1. The above display is a coordinated set of graphics and text which appears to be a single display but is actually several components constructed piece by piece.

Jesse M. Heines, Ed.D. is an independent consultant based in Chelmsford, Massachusetts. He develops custom CBT programs and provides training and consultation on CBT course development. Dr. Heines is also an assistant professor of computer science at the University of Lowell in Lowell, Massachusetts, and the author of Screen Design Strategies for Computer-Assisted Instruction.

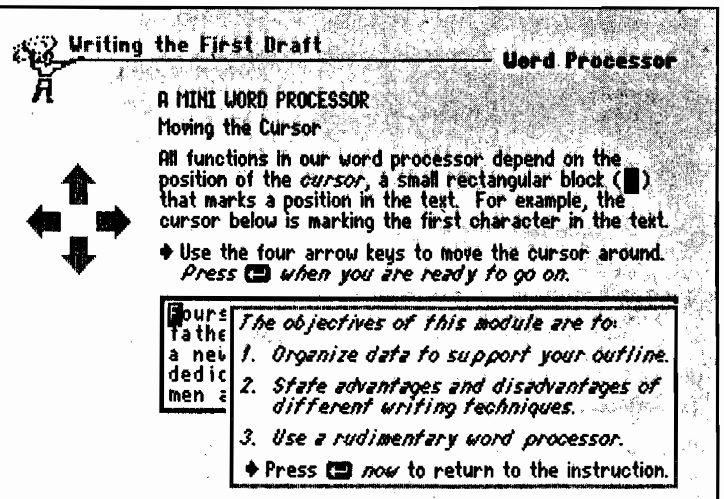
THE CBT CRAFTSMAN

with an added feature: the computer remembers what was on the screen before the window was displayed. When you finish using a window and tell the computer to delete its definition, the area that it occupied is restored to what it looked like before the window was defined. Implementation of windows requires enough memory to store multiple copies of the screen. For this reason, most systems limit the number of windows that can be overlaid simultaneously to a small, single-digit number.

Windows have many applications in CBT. Besides allowing you to extend your screen arbitrarily, they allow you to create display structures that are otherwise nearly impossible. Consider, for example, the display in Figure 1. This figure shows a rather complex display with several functional areas. The display itself is not stored as a single entity, but rather as a coordinated set of graphics and text that are displayed one at a time (but very quickly) to form the complete screen. It is important to understand that this screen builds from the previous screen in viewport fashion. That is, the "David" trademark (as in David and Goliath) and the main title "Writing the First Draft" were not redrawn from the previous screen, but the text and arrow graphic were added, obliterating the previous information in those functional areas.

The course from which this display is reproduced contained a "review objectives" feature which allowed students to see a restatement of the current module's objectives at any time. When this feature is invoked, the objectives are displayed in a window as shown in Figure 2. Since this functional area is a real window, pressing the RETURN key with the objectives displayed restored the original screen exactly as it was before the student invoked the review objectives feature.

The use of windows for this feature—as opposed to erasing the screen, displaying the objectives, and then regenerating the screen—has three main advantages. First, it eliminates the need to regenerate the screen. This would be a tedious programming task due to the way in which each screen builds from the previous one. Second, it makes it possible to write a single routine for displaying the objectives, because the screen state is saved before the objectives window is defined. The actual screen state is irrelevant to the routine; it is simply stored as a series of memory locations. Thus, the objectives window can be displayed on *any* screen, and the windowing software assures that when the objective window definition is deleted, the screen will be restored to the way it was before the window was defined, regardless of the complexity of that original state. Third, the use of a window maintains the instructional continuity because it simply overlays a part of the screen temporarily without destroying the visual context of where the students were before they invoked the review objectives feature.


Writing the First Draft — Word Processor

A MINI WORD PROCESSOR

Moving the Cursor

All functions in our word processor depend on the position of the *cursor*, a small rectangular block (█) that marks a position in the text. For example, the cursor below is marking the first character in the text.

◆ Use the four arrow keys to move the cursor around.
Press ⏎ when you are ready to go on.

```
Fours
fathe
a new
dedic
men a
```

The objectives of this module are to:

1. *Organize data to support your outline.*
2. *State advantages and disadvantages of different writing techniques.*
3. *Use a rudimentary word processor.*

◆ Press ⏎ *now* to return to the instruction.

(Copyright 1985 by KJ Software, Inc., reproduced with permission)

Figure 2 shows a window overlaid on the graphics and text of Figure 1. Because it is a true window, pressing RETURN would bring the user back to Figure 1.

The implementation of windows, like the implementation of character fonts I discussed in my last column, is not easy if it is not provided by your authoring system. The programming job is not within the realm of typical lesson programmers, but it is within the realm of typical systems programmers. As I pointed out in my very first column ("Anybody Can't Do CBT," *Training News*, March, 1985), building CBT with these levels of sophistication requires a team approach. CBT authors must not limit their instructional designs to those types of interactions that they can program themselves. There is much more that can be done on a computer than is dreamt of in most authoring systems' philosophies. Demand a system that provides power—even above ease of use—and demand the assistance of a programmer who knows how to use that power. Then let your imagination soar and your courseware shine. □