Changing the Scenery

A few thoughts on designing courseware for multiple-platform delivery.

BY JESSE M. HEINES, ED.D.

Dr. Jesse M. Heines is an independent consultant on CBT and an associate professor of computer science at the University of Lowell in Lowell, Massachusetts. Dr. Heines provides consultation and authoring workshops on CBT design and development, and develops custom CBT courseware.

About ten years ago, a colleague told me that he had written a program to allow PLATO lessons to run on an Apple II. "How wonderful!" I thought—Apple II users could now gain access to the vast PLATO library. Then he told me that, unfortunately, the graphics were inferior to the originals due to a lower screen resolution. And that they were squashed a bit, too, because the aspect ratio of the Apple screen did not match that of PLATO's. And of course, he lost the lovely PLATO fonts; you wouldn't expect them to port to another machine, would you?

"So what did translate completely?" I asked, "The lesson logic," he replied. Now, it's no mean feat to translate one language to another, but computer-based instruction is such a highly visual medium that translating the lesson logic just isn't enough. If a course loses its "look and feel," it may also lose much of its instructional value, or at least its visual appeal. So what issues should you consider when designing courseware to run on multiple delivery platforms, e.g., both Macintosh and MS-DOS machines?

Hardware Screen Considerations

There are three main hardware characteristics that affect how graphics and text appear on a computer screen: the display hardware's aspect ratio, resolution, and color capabilities.

If a course loses its "look and feel," it may also lose much of its instructional value, or at least its visual appeal.

Aspect Ratio: Aspect ratio refers to the relative horizontal and vertical spacing of the display's picture elements (pixels). Macintosh computers and MS-DOS models configured with the popular Video Graphics Array (VGA) have square spacing, i.e., an aspect ratio of 1:1. Other MS-DOS models use spacing that is from 1.37 to 2.4 times as tall as it is wide.

Courseware displays designed for screens with square pixels preserve their original appearance when transported to other square pixeled screens, but can appear vertically elongated when moved to models

having a different aspect ratio.

Aspect ratio distortion is very noticeable in graphic figures. Circles become ellipses, squares become rectangles, and isosceles triangles will be stretched. These distortions may go unnoticed in titles and other graphical elements, but may stick out like sore thumbs if the instructions tell students to "move the square until it covers the circle."

Resolution: Screen resolution commonly refers to the number of pixels available on a display. Some Macintosh models have built-in monitors containing 512x342 pixels; others support displays with resolutions of 640x480 or greater. MS-DOS-based displays typically range from the older Color Graphics Adapter (CGA) at 640x200, Enhanced Graphics Adapter (EGA) at 640x350, and the current VGA standard at 640x480. As with the Macintosh, MS-DOS-based computers can be equipped with displays having even greater resolution. Resolution controls the level of detail that can be presented in CBT designs. The greater the resolution, the larger the effective display area, and the greater the level of detail that can be presented. CBT content that requires very realistic, highly detailed displays should be delivered on high-resolution computers, while the older computers having less resolution may be adequate for simpler content. Courseware that depends on high-resolution displays cannot be easily transported to older hardware having smaller effective display areas.

It's important to note, however, that even though the hardware trend is to higher resolution, excessive detail is not always desirable in instructional situations. In courseware, we often want students to focus on one aspect of a problem; too much detail might be more of a hindrance than a benefit. The old adage that a "picture is worth a thousand words" is only relevant when one wants to say 1000 words. In other instructional situations, a simpler message can be more effective.

Color Capabilities: Another hardware consideration for authors designing CBT for multi-platform delivery is variation in the number and selection of colors that can be displayed at one time. Monochrome, 16-color, and 256-color support are common options, and some newer systems can display over 16 million colors.

When courseware is to be transported from one platform to another that supports fewer colors, color mapping effects must be taken into account. Hues that are distinctly different on a 256-color platform may map to the same selection when displayed on a 16-color device. Careful color selection can help minimize these effects.

A more subtle issue is color shifting—the effect of red or other colors not being the same shade on all systems, even among systems that support the same number of colors. We've all stood in the TV section of a department store and looked at the rows of televisions all showing the same channel and noticed that not all pictures look exactly the same. On some the colors are shifted more toward the green, on others, more toward the red. Similar effects can be seen in transporting CBT courseware across different platforms. If highly realistic use of color is desired, the courseware should be designed for use only on computers having the necessary color support.

It's important to note, however, that even though the hardware trend is to higher resolution, excessive detail is not always desirable in instructional situations.

Software Screen Considerations

While it's sometimes difficult for the CBT developer to exercise control over selection of student delivery hardware platforms, the developer can do much to control the look and feel of the software. Two important issues in this area are the use of fonts and the features of displayed windows.

Fonts: Can fonts from one courseware platform be used effectively on another? This question rests both on the availability of the currently used fonts on the new platform and the desirability of using them there.

Screen fonts commonly used on the Macintosh may not be available on MS-DOS computers and vice versa. For example, the standard screen fonts provided in the Macintosh and *Microsoft Windows* systems are two distinctly different sets. Fonts not normally present on one system may have to be transported to, or re-created for the other system using software tools.

Assuming fonts can be transported to or matched on the targeted system, developers should consider the desirability of doing so. On the one hand, using the same fonts can help minimize effort required to revise text displays and allows the courseware to retain its look and feel. On the other hand, it may be desirable to change from fonts typically used in the original environment to those most frequently used in the other. For example, many Macintosh developers love the

Chicago font, while most MS-DOS developers prefer Times Roman or Helvetica. While I feel that any clearly readable font is acceptable for the bulk of one's instructional text, I contend that a course delivered on an MS-DOS system using the Chicago font will have a distinctly Macintosh look.

> While it's sometimes difficult for the CBT developer to exercise control over selection of student delivery hardware platforms, the developer can do much to control the look and feel of the software.

If you opt to use different fonts in each environment, you will want to design screens with sufficient white space so that the text may shrink or grow without impacting the overall screen design. I would add that this is always a good posture to take even if one is designing for a single platform—it improves readability and allows programs to be translated into other languages that require more space than English.

Windowing Features: All windowing systems have many features in common, but there are some distinct differences, too. When one clicks on the "close" box in a window in a Macintosh application, the window closes immediately. The same action in *Windows 3.0*, on the other hand, opens a control box that allows users to select from a list of commands with "close" as one of them (Double clicking on the "close box" in the upper left corner closes the window directly).

Common actions on most windowing systems include clicking the mouse pointer in the window to make it active and dragging the scroll bar to view previous and subsequent text. Techniques for resizing a window and moving it to the background, however, may be accomplished differently.

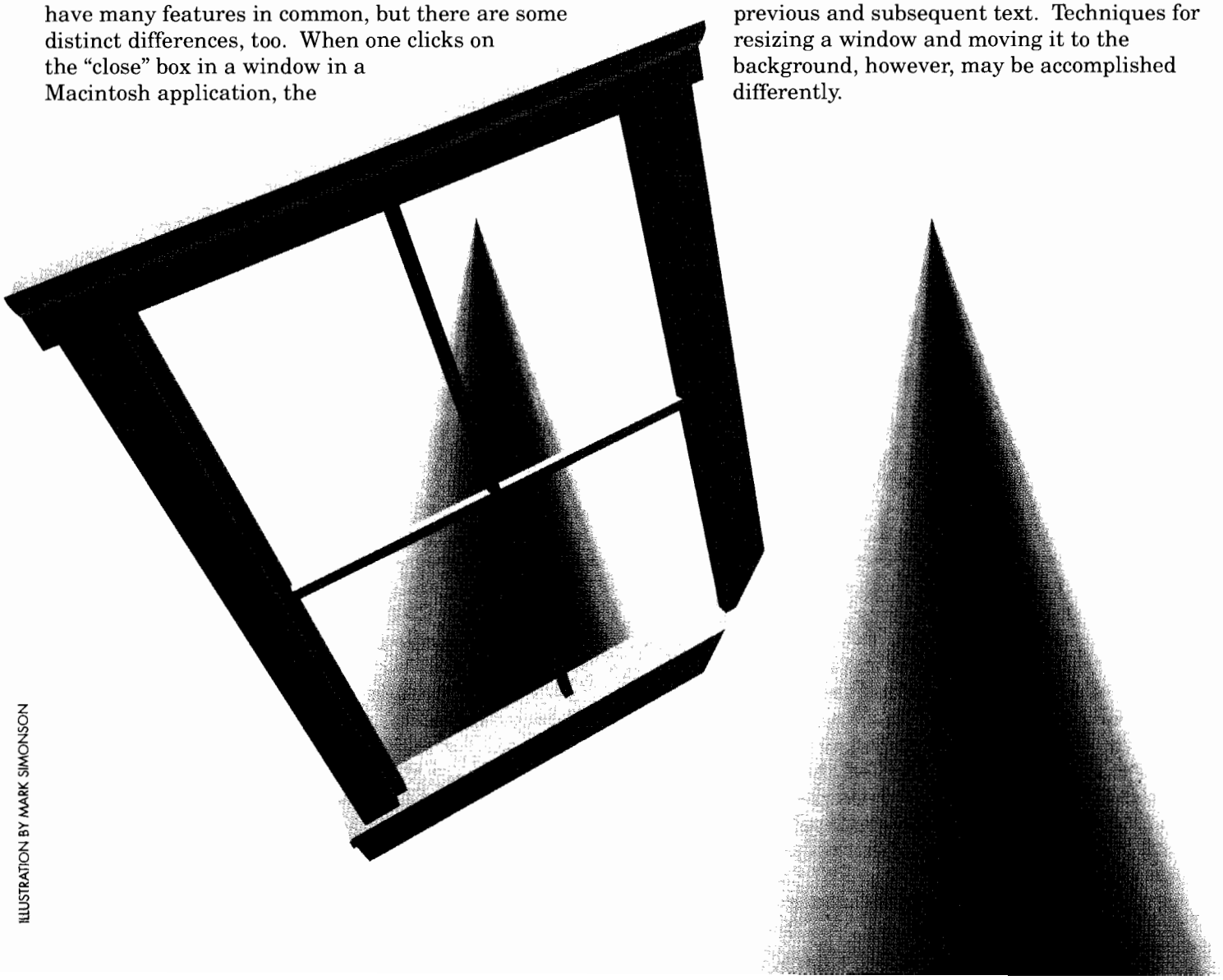

ILLUSTRATION BY MARK SIMONSON

Differences in windowing environments may not cause any significant problems for either the developer or the student, but developers should be aware of them when giving students instructions for working within courseware. One may either conditionalize the course to give different instructions on different systems or word instructions that concern the environment in ways that are meaningful on all targeted platforms.

Interaction Considerations

Users of different systems vary in their expectations about how their systems should react under certain conditions. Macintosh users know that if they happen to pull down the wrong menu, they can make it go away by moving the mouse pointer off the menu and releasing the mouse button. A *Microsoft Windows* pulldown menu, on the other hand, would stay on the screen until the user took specific action to remove it (usually a mouse click outside the menu or a keypress).

Macintosh users really couldn't live without the mouse, even though there are command-key equivalents for almost all menu functions. MS-DOS users are accustomed to being able to use the keyboard for all functions if they don't want to use or don't have a mouse. And not every MS-DOS user likes to use a mouse. I know of one who so hates the "rodent," as he calls it, that he keeps it shackled in the biggest mouse trap I've ever seen.

When dragging objects on the screen, Macintosh users are accustomed to the entire object moving under their control. MS-DOS users, however, typically expect a rectangular outline of the object to track their mouse actions and the full object to snap to the new position once the mouse button is released. These minor differences may go unnoticed to many users, but like the windowing features described previously, they must be considered when giving instructions on dealing with the CBT interactions.

A more important issue involves the differences in mice typically used on Macintosh and MS-DOS systems. The Macintosh uses a one-button mouse. MS-DOS and most other workstation systems use mice with at least two buttons, and some popular models have three buttons. Thus, while Macintosh users must press the shift or command keys to access different mouse functions, MS-DOS users may be accustomed to pressing different mouse buttons to accomplish these functions.

A user's expectations of how the mouse should react are shaped by his or her first experience. No platform's mouse strategy can be called right or wrong, but users might become confused when courseware on their platform uses an unfamiliar mouse strategy. So CBT designers planning to develop for multiple platforms must be sensitive to these differences and provide appropriate instructions.

User Considerations

Whatever the delivery platform, it may be best to adhere to the general standards of that platform, be they explicitly stated or merely *ad hoc* standards. As a courseware developer, I've worked with numerous authoring systems, and have always found it easier to work within a system's view of the world rather than trying to impose another paradigm on the system. There are certainly times when special course designs or my own creativity dictate that I depart from this stance. But if you don't have a significantly better way to do something, do it the way everyone else does it. And the best way to find out how everyone else does it is to look at a lot of courses and applications (something very few of us do sufficiently).

> It's easier to work within a system's view of the world rather than trying to impose another paradigm . . . if you don't have a significantly better way to do something, do it the way everyone else does it.

I've said in my workshops that if you have to write a separate module to teach students how to take your course, your design is too complex. On the other hand, today's proliferation of multiplatform delivery demands that some orientation be given for students who are used to other systems. To address these issues I recommend embedding extensive help sequences within the course flow rather than asking students to go through a separate introductory lesson or module, which they may choose to skip anyway.

Your delivery platform is nothing more than a medium through which students gain access to information. The most important part of any course will first, last, and always, be its content. Delivery environments must help students gain access to that content regardless of the platform on which a course is running. A truly excellent environment is one that is so transparent to end-users that they seldom think about it.