The best CBT programmer I ever worked with had red hair, a red beard, and a Cuban wife. When I first met her, she was shy, but then she didn't seem offended or annoyed. Anyway, the phrase became a kind of password between this superprogrammer and me, and many detailed code design sessions began with one of us walking into the other's office and chirping "Que pasa?" He never once said that an interaction had dreamed up

Jesse M. Heines is an independent consultant based in Chelmsford, Massachusetts. He develops custom CBT programs and provides training and consultation on CBT courseware development. Dr. Heine 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.

Some authoring systems will provide all of these options, while others will be much more rigid. The rigid systems not only stifle creativity and the development of crafted courseware, but they can also seriously affect the program's instructional effectiveness. Such rigidity forces course authors to adapt their teaching strategies to the capabilities of the authoring system, thus simplifying the range of interactions in the name of simplifying the program implementation. At their best, programs with restricted interactions are boring, at their worst, they're instructionally stagnant.

Not everyone is as fortunate as I have been to work with a good programmer who implemented interactions as I designed them rather than as dictated by the constraints of some authoring system. Response handling is the most visible of these constraints, but others exist, particularly in the display of graphics. For example, I become very frustrated when I know that my computer's basic input/output system provides sophisticated graphics functions that I cannot access from an authoring language. Alfred Bork, the reknowned pioneer of computer-assisted instruction in physics, and his colleagues at the University of Calirfornia at Irvine develop all of their courseware in Pascal. The reason for this choice, according to Bork, is that Pascal is a particularly good authoring language. It is that Pascal is simply a good language.

Remember that naive courseware authors, like naive computer users, stay naive for only about two weeks. Sooner or later they outgrow the capabilities of almost all authoring systems and want to do more sophisticated processing like parsing. Another old adage—"make it simple enough that any idiot can use it and only idiots will"—holds true for authoring systems as well as applications programs.

I am all for ease of use, but not at the expense of functionality. Instead of trying to shoehorn your instructional design into a restrictive authoring system, get yourself a real CBT programming language with a rich set of subroutines—and a red-hairied programmer if necessary—and you'll create CBT courseware that is Instructionally sound, visually stimulating, and comfortable for students to use.