THE CBT CRAFTSMAN

Not Just Another Pretty Authoring System

The talk of the town at the recent ADCIS conference was a startling new authoring system that lets CBT course authors structure programs much more easily than ever before.

Jesse M. Heines

When the Association for the Development of Computer-Based Instructional Systems (ADCIS) held its 28th International Conference in Washington D.C. this past November the talk of the conference was a new authoring system called *Course of Action*. As my title implies, this system is not just another programmer's view of what authors should want to do in CBT. *Course of Action* comes from a new company called Authorware, Inc., whose CEO just happens to be Dr. Michael Allen. Allen, who has served as president of ADCIS and editor of the *Journal of Computer-Based Instruction*, possesses a panoramic understanding of CBT shaped by years of courseware development and research project management on the PLATO system at Control Data Corporation.

When word got around that Allen and his

Jesse M. Heines is an assistant professor of computer science at the University of Lowell in Lowell, Massachusetts. He is the author of Screen Design Strategies for Computer-Assisted Instruction *as well as numerous articles on courseware development. Dr. Heines provides training and consultation on computer-based training, develops custom training programs on contract, and writes "The CBT Craftsman" every other month.*

partners, Carl Philabaum and Steve Birth, were showing their new product in a suite on the ninth floor, ADCIS stalwarts expected the system to be worth the wait for elevators. I daresay that few, however, expected to see such a polished system focused so precisely on the major bottlenecks in courseware authoring productivity. People (including yours truly, "The CBT Craftsman") returning from a visit to Authorware's suite looked as if they had just had a number of their assumptions about courseware development seriously challenged.

Course of Action is implemented on Apple's Macintosh. To appreciate it fully, one must first understand the basic difference between the Macintosh and other personal computers. The secret to the "Mac," as it is often affectionately called, is *integration*. True Macintosh software never stands alone as a single application. Rather, it uses (and allows itself to be used by) the rich variety of software tools provided by the Macintosh operating system. These tools include text and graphic editors, facilities for moving displays from one application to another, and a common user interface built around the mouse and on-screen windows. Provided that software developers follow the Mac conventions, the question "Can I Call You?" (see my July 1986 column) practically becomes moot. A separate call does not have to be made; the user simply opens up another window.

Course of Action is beautifully integrated into the Mac environment. You may use the authoring system's own graphics editor, or you may use MacPaint. You may use its own text editor, or you may use MacWrite. Fonts are built from the standard Macintosh font facilities. *Course of Action's* unique contribution is a powerful and flexible course *structuring* tool that helps authors pull together screen displays, interaction strategies, and response analyses into a cohesive program that is highly maintainable. This tool lets authors work at a high level that is difficult to achieve with authoring languages and is often so restricted by other authoring systems as to be virtually useless.

Figure 1 shows the author's top level view when *Course of Action* is initially invoked. Courses are structured by using the Macintosh mouse to move the course component icons shown at the left onto the "course flow line" that appears to the right of the icons. The eight basic course component
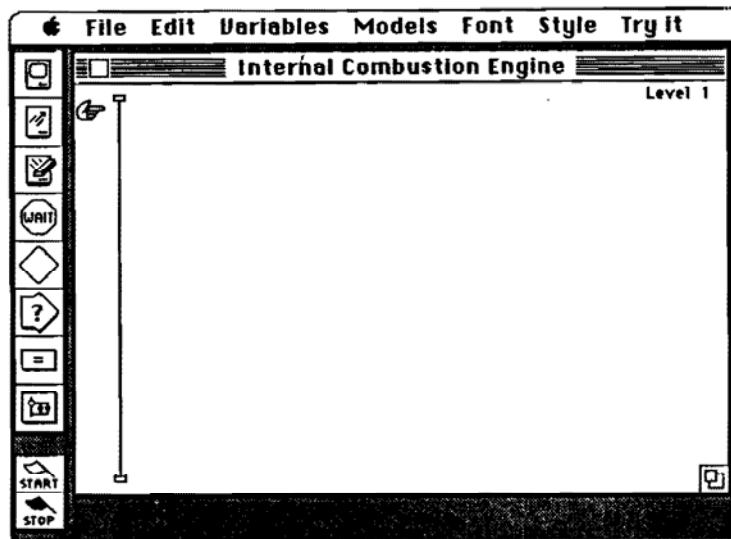

Figure 1. The top level of Authorware's Course of Action as seen by a course author. (Copyright 1986, 1987, Authorware, Inc., reprinted with permission.)

icons include:
- *display* for specifying text and graphics to be displayed to students,
- *animate* for moving objects on the screen,
- *erase* for erasing parts of the screen,
- *wait* for indicating that the system should pause until the user responds or a specific time interval has elapsed,
- *decision* for branching on a variety of contingencies,
- *question* for presenting students with both structured (multiple choice or menu selection) and free response items and providing appropriate feedback,
- *calc* for making calculations and "dropping down" to the system's underlying code (the language C), and
- *map* for creating macros (or subroutines) that are actually sequences of course component icons treated as a whole.

Other course components are available in various stages of completion, but the eight listed above are the main ones around which most courseware is built.

As authors move icons to the course flow line, the structure they create is automatically represented in the flowchart-like format shown in Figure 2. (Allen and his colleagues refer to this representation as the

course "map.") By touching one of the icons on the course flow line, the author can "edit" that course component. I'm not sure that "edit" is the right term to use here, because one does not "edit" on the Mac in the normal sense of the word. The Mac's integrated environment really is more of a screen "assembler" than editor, as graphics and text are pulled from other applications and arranged on the screen.

Figure 3 shows a typical course display invoked by touching one of the icons on the course flow line. Note that this display includes the standard Mac operating system icons and keywords at the top of the screen and other keywords added by *Course of Action*. Touching the keyword "Try it" causes a pop-up menu to appear. One of the options on this menu is "Jump to maps," which returns the author to the course flow line display.

I asked Carl Philabaum if he felt that authors *programmed* using *Course of Action* and he replied "No, they *author* or *design*." I find this use of terms less than satisfactory, but it is perhaps a compliment to the non-verbal Mac environment and the degree to which *Course of Action* is integrated into it that I find it difficult to describe verbally just

what one does while working with the system.

To me, the work of an author using *Course of Action* more closely resembles that of a designer and layout artist than that of a writer using a word processor. One can, of course, "edit" screens in the normal sense, but I do not perceive this as one of the system's strengths. It is clear that working on the Mac with visual, icon-oriented software is quite different from programming with editors on other systems. I strongly encourage readers who have not experienced this type of software environment to pay a visit to their local computer stores and spend an hour playing with the Mac to get a feel for its possibilities.

Course of Action has all of the standard CBT facilities for specifying anticipated correct and incorrect answers and providing feedback. It also has a rich set of easily-accessible system variables that automatically keep track of statistics such as the number of tries at answering a specific question, the total number of correct responses, and so on. I won't dwell on these facilities here because they're common to most authoring systems. It would be a disservice, however, not to mention that author-defined variables are available and can be used in calculations of all types, allowing for sophisticated software such as simulations.

We now turn to my three favorite questions of authoring system vendors. First, "can I call you?" As discussed previously, this question is somewhat moot on Macintosh systems, but not completely. Philabaum told me that the system *is* callable, and that some vendors are looking at it as a vehicle for on-line documentation for their products as opposed to a training vehicle per se.
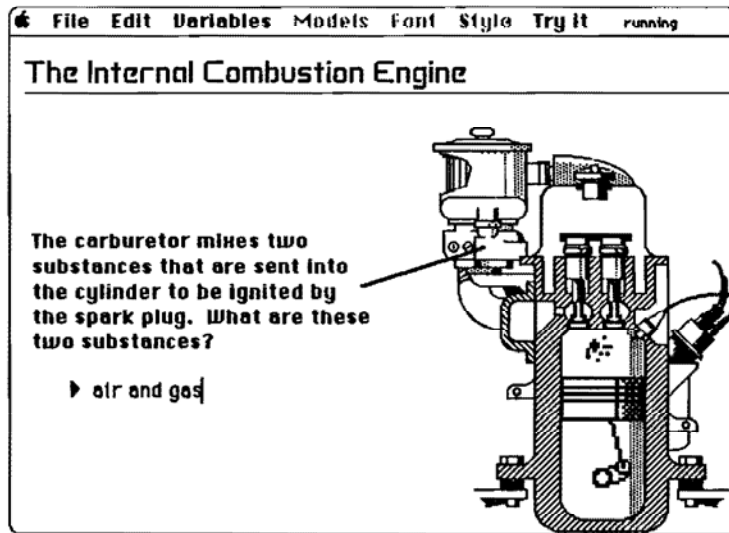
Second, "can I 'drop down' to the language in which the authoring system is written?" The *calc* procedure, Philabaum informed me, is really precisely that. In addition to allowing calculations, *calc* actually makes the underlying C language available to authors who wish to do sophisticated programming.

My third question deals with the degree to which free-format student responses can be used to control lesson execution. That is, "can the course do unanticipated things in addition to handling either correct or incorrect responses anticipated during course development?" Suppose, for example, that we are teaching about a mathematical function and present examples of graphs generated with a variety of data values. We then want to let students enter their own data values and show how these values affect the formula by plotting the appropriate points.

With this scenario in mind, I therefore asked Philabaum the following question: "Suppose my course concerns circles and I show students circles of various radii. Can I then ask them to enter a radius and plot the appropriate circle?" "Hmmm," he replied, "not at present." He then hastened to add, "We're working on it," and based on what I've seen so far, I'm sure they are. (As we have seen, *Course of Action* does allow you to "drop down" and program this level of interaction in C if you have the programming knowledge.)

In addition to the competence of Authorware's product development team and the excellent Mac integration of their product, I was impressed with the fact that they're not pushing *Course of Action* onto the market before it's ready. As one would expect, some of the facilities shown in Authorware's demonstration used capabilities that are currently under development in versions that are not yet ready for beta test. One has to compliment Authorware, however, for showing the current version of their product to the experienced course authors who make up the ADCIS membership.

Philabaum stated that the product, including a translator that allows courses developed on the Mac to run on an IBM PC, is scheduled for delivery next spring. The price of the authoring system package is a big variable at this time, but it looks like Authorware is going to try to keep the price of publishing a course written with their system very low—under one dollar per disk—making it viable for the academic as well as business markets. Readers interested in further information should write to Michael Allen, president and CEO, Authorware, Inc., 8621 Pine Hill Road, Bloomington, Minnesota 55438, or call (612) 941-5752. ☐
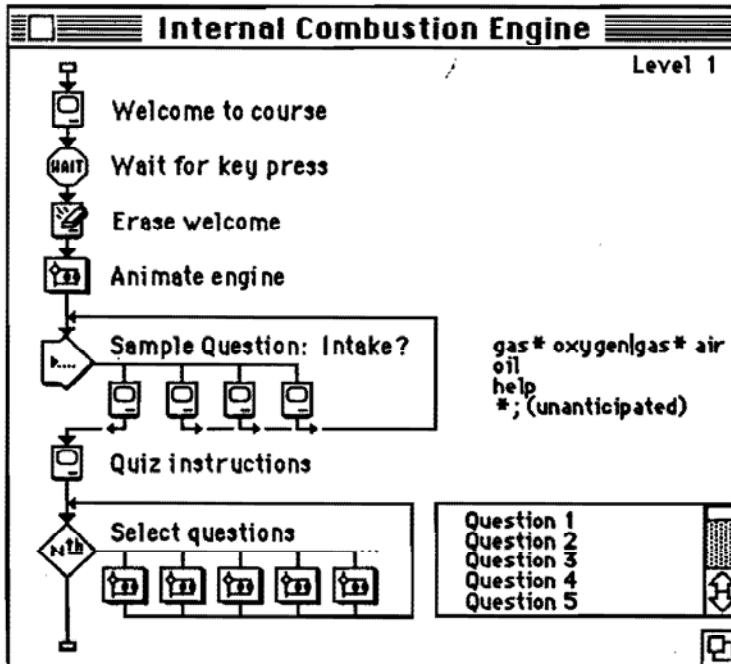

Figure 3. *A typical course flow display as seen from the authoring environment. (Copyright 1986, 1987, Authorware, Inc., reprinted with permission.)*

Figure 2. *A course "map" representing flow of control in a CBT lesson. (Copyright 1986, 1987, Authorware, Inc., reprinted with permission.)*