TrainingNews
Volume VIII, Number 1 September 1986

THE CBT CRAFTSMAN

Copyright (c) 1986
Weingarten Publications, Inc.

Jump Up, Drop Down

Reprinted with permission.

Course authors, like scuba divers, can work at different levels—and course authors can even shuttle back and forth between "depths."

Jesse M. Heines

I am not an enemy of authoring languages, authoring systems, or any tool that increases CBT author productivity. It's just that I don't want my courseware to be limited by what someone else thinks a CBT program is supposed to do.

My July column ("Can I Call You?") discussed one feature that can be incorporated into authoring languages and systems to remove their limitations: making them callable. This month's column discusses another feature—allowing authors to work with the authoring system at multiple levels. Consider, for example, an authoring system that offers two levels, or "shells" (borrowing the term used by *Unix* for a specific interactive user environment).

The upper-level shell might be menu-driven, leading CBT authors through the creation of standard instructional sequences. If authors wish to develop non-standard sequences, however, they can "drop down" to the language in which the upper-level shell is written, gaining access to more primitive system commands and functions. While working at the lower level, authors can "jump back up" to the upper level at any time, with the system automatically incorporating the non-standard code inserted at the lower level.

I'll use a common multiple-choice interaction as an example of both the usefulness of being able to move between authoring levels and an example of how a multi-level authoring system might appear to course authors.

Let's suppose we're developing a course on basic open-water scuba diving, and we want to ask the question shown in Figure 1. This screen is rather basic, containing a very simple graphic (the international scuba flag), a title, a question that contains only text, and four multiple-choice, text-only alternatives. (I always label my multiple-choice alternatives with numbers rather than letters because the numbers are easier for non-typists to find on standard QWERTY keyboards.)

This type of screen is easily created with most authoring systems by specifying the text of the question followed by its four alternatives. In some authoring systems, all of the text will be contained in one text block. In others, the texts of the alternatives might be specified separately so the system can "randomize" the alternatives, thus generating multiple forms of the question.

Once the question and alternative texts are entered, most authoring systems expect the author to indicate which of the alternatives are "correct." In addition (or perhaps concurrently) the author is expected to enter responses to be displayed to the student for each of the possible alternatives or branches to occur when each alternative is selected.

For example, for the question shown in Figure 1, alternative 2 is the only correct answer. If the student selects alternative 1, the system might respond as shown in Figure 2. Once again, this type of interaction can usually be specified most easily at the highest level of the authoring system because it's straightforward and involves only text.

Now comes the fun part. Suppose that the numbers 58 and 42 presented in the question text are actually generated by the program using some algorithm rather than "hard-coded" in the question text. We must now generate both the question and our explanation for wrong alternatives "on the fly" rather than being able to specify them directly to the authoring system. The advantage of this approach is that one question format can generate a very large number of actual questions, allowing students to practice the skills repeatedly until they master the concepts.

But we must first write the algorithm for selecting the numbers. Some authoring systems will allow you to do this at the top level by indicating, for example, that the first number is an integer between, perhaps, 30 and 70, and the second is an integer between perhaps 40 and 60. Unfortunately, the problem isn't that simple, because the logical range of the second number will be dependent upon the value of the first number.

No big deal, you say—just use the first number in the expression for the second. That won't quite work in this case, because the relationship is not algebraic. What you really want is to look up the first number in a data structure that contains the standard "Dive Tables" and derive a logical range for the second number from the data structure. ("Dive Tables" are charts that relate dive depths, times, and surface intervals to blood nitrogen levels so that divers can determine how long they can stay at specific depths without risking decompression sickness, or the "bends.") This data structure will also provide logical response alternatives which, again, cannot be generated algebraically.

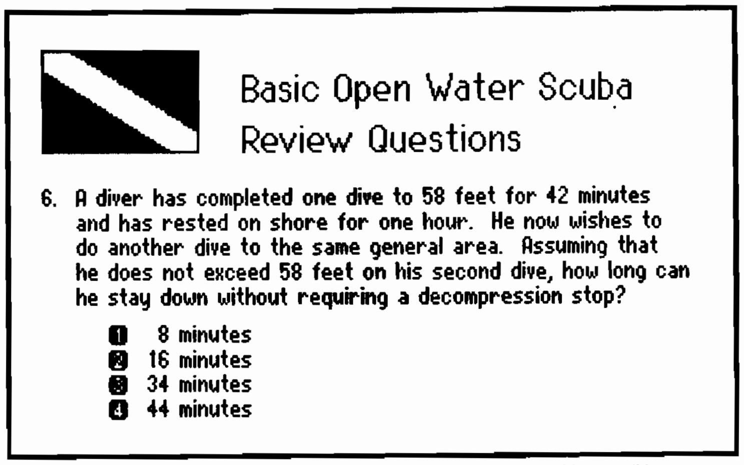
This type of data manipulation is usually beyond the scope of most authoring system top levels. We therefore want to drop down to a lower level where we can access data files and do complex calculations. Once we have our data, we want to store it in variables that can be used in the text display statements at the top level. At the top level, the lower level code will look like a subroutine call, with its complexities hidden. The subroutine might be written by a programmer who is not the primary course author. All the course author needs to know is how to call the subroutine and how to refer to the numbers returned from the lower level code so that he or she can include these variables in the question text. In this way, the two levels allow course authors to take advantage of all the text formatting, error checking, and pre-programmed multiple choice interaction provided by the authoring system, without limiting authors' ability to generate a large number of complex questions whose data is generated by a subroutine.

Let's take things one step further. Suppose that instead of responding with the simple text message shown in Figure 2 when the student selects alternative 1, we want the system to go into a detailed explanation of why the response is wrong and review for the student how to derive the answer from the Dive Tables. To do this, we want the system to display appropriate sections of the Dive Tables in a window and indicate how each is used to derive the answer. (There are three Dive Tables that must be consulted.)

The first such display might look like that shown in Figure 3. The appropriate section of the first Dive Table is displayed with the relevant reference data circled. The numbers within the box represent times in minutes. The numbers in the reverse video indicate the maximum total time a diver can stay at that depth without risking decompression sickness. This format is copied directly from the actual tables.

Generating this screen "on the fly" takes even more computing than generating reasonable alternatives. The system must "know" how to use the Dive Tables so that it can decide the appropriate section to display and format the screen to fit within

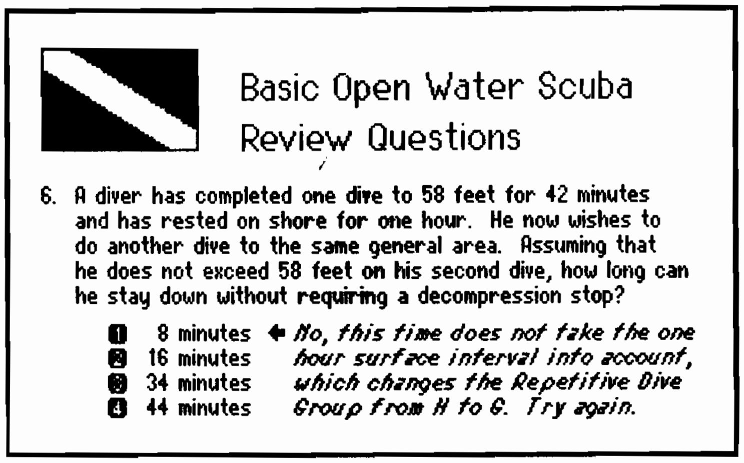
Jesse M. Heines, Ed.D., is an assistant professor of computer science at the University of Lowell in Lowell, Massachusetts. He is the author of Screen Design Strategies for Computer-Assisted Instruction *as well as numerous articles on courseware development. Dr. Heines provides training and consultation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


Basic Open Water Scuba
Review Questions

6. A diver has completed one dive to 58 feet for 42 minutes and has rested on shore for one hour. He now wishes to do another dive to the same general area. Assuming that he does not exceed 58 feet on his second dive, how long can he stay down without requiring a decompression stop?

1. 8 minutes
2. 16 minutes
3. 34 minutes
4. 44 minutes

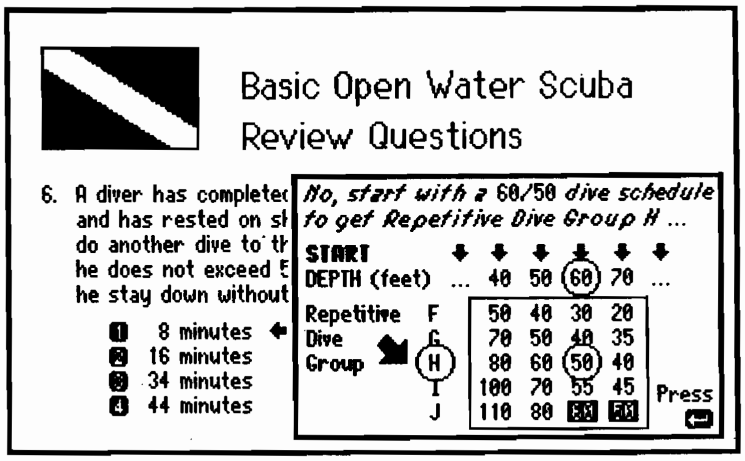
Figure 1 is a fairly standard CBT screen. It has a graphic, a title, a question, and four possible answers.


Basic Open Water Scuba
Review Questions

6. A diver has completed one dive to 58 feet for 42 minutes and has rested on shore for one hour. He now wishes to do another dive to the same general area. Assuming that he does not exceed 58 feet on his second dive, how long can he stay down without requiring a decompression stop?

1. 8 minutes ← *No, this time does not take the one hour surface interval into account, which changes the Repetitive Dive Group from H to G. Try again.*
2. 16 minutes
3. 34 minutes
4. 44 minutes

Figure 2. The student has chosen the first answer, which is wrong. The standard response from the authoring system is to simply notify the student that his or her answer is wrong. This particular system adds a brief explanation.


Basic Open Water Scuba
Review Questions

6. A diver has complete
and has rested on sh
do another dive to th
he does not exceed 5
he stay down without

1. 8 minutes ←
2. 16 minutes
3. 34 minutes
4. 44 minutes

No, start with a 60/50 dive schedule to get Repetitive Dive Group H ...

START		↓	↓	↓	↓	↓
DEPTH (feet)	...	40	50	60	70	...
Repetitive	F	50	40	30	20	
Dive	G	70	50	40	35	
Group	H	80	60	50	40	
	I	100	70	55	45	
	J	110	80	60	50	

Press ↵

Figure 3. In this sophisticated screen, the system responds to the student's wrong answer by not only specifying why the answer is wrong, but actually calling up the appropriate Dive Table and then showing the student how to derive the direct answer.

the window. Most authoring systems expect the screen layout to be fixed at "compile time," allowing precious little reformatting as the lesson is running. By allowing the author to drop down to a lower level, the system can give authors the ability to tailor their courseware to individual student responses. Once the display parameters are computed, however, authors will want to jump back up to the top level to accept the simple RETURN key press to display the next screen.

We now have two techniques for taking advantage of the features that authoring systems offer, yet removing the restrictions they often impose—making them callable, as discussed in my July column, and providing multiple authoring levels, as discussed in this column. In November's "CBT Craftsman," I'll discuss a third technique, designing custom authoring "front ends" for specific instructional applications. As always, I encourage readers to write to me (in care of *Training News*) describing other techniques for getting the most out of our authoring tools. □