A SIMPLE AUTHORING SYSTEM FOR COMPUTER-ASSISTED INSTRUCTION

B. Gregory Louis
President
Dynamicro Consulting Limited
494 Millwood Road
Toronto, Ontario, Canada M4S 1K5

The FORTH outer interpreter provides a convenient infrastructure on which to build support routines for writing computer-assisted instructional material (CAI). We have implemented a system that runs courseware by LOADing FORTH screens on which the course author has prepared the material to be presented. The system supports both unconditional and conditional branching (necessary to allow the courseware to adapt itself to the student's progress based on test results). There are few tools to master; course writers who know FORTH can expect to be ready to use the system productively within half an hour of initial contact.

INTRODUCTION

An ongoing problem with "distance education" - provision of educational courses to people unable to reach an institution of higher learning - is that many students have difficulty pursuing the course systematically because of the lack of personal direction and feedback. The inclusion of computer-assisted instruction (CAI) material in such courses can go far to reduce or eliminate this difficulty. The computer substitutes in part for an instructor by providing interactive instruction and testing; in a well-written CAI program, material to be presented may be selected based on student performance. The student is additionally placed in contact with a human instructor at regular intervals to ensure that progress continues and any student problems are appropriately dealt with.

The production of good CAI material is difficult and costly. There is an analogy between this activity and the production of "expert systems", inasmuch as it is rare for experts in the field of study to be, as well, competent in course design and familiar with the peculiar requirements of the CAI medium. Much effort - in the author's view, misguided - has gone into attempts to produce CAI-writing tools that are easy for the non-programmer to manipulate. This paper will present another approach: a simple toolset intended for use by a FORTH-oriented "knowledge engineer" engaged in production of CAI courseware.

METHODOLOGY AND DESIGN

The use of FORTH as the environment for CAI authorship greatly facilitates the design, because most of the required facilities for CAI writing are already inherent in the FORTH system. The FORTH block-based disk I/O provides a natural CAI "text frame" of suitable size, and the frame text can be emplaced with the aid of a full-screen FORTH editor such as is now commonly supplied by most FORTH vendors. The fact that the screen is interpreted, rather than simply displayed, by the FORTH system allows the provision of flow-control and user-interaction tools that the course author can access "in line" during the course-writing process. There is thus no separation between the text and the CAI "program" that drives the text display, which gives a very natural "feel" to the working environment; further, the author has access to the full capabilities of the FORTH outer interpreter at any point in the course.

Specific requirements of the CAI environment seen by the student include the following:

1. Simple and secure student interaction. In the design presented here, only keys that give responses valid in context are accepted by the system; the rest of the keyboard appears "dead".
2. Flow control. Conditional execution, binary and multiway branching are all needed to permit the author to write courseware that tailors itself to the student's demonstrated ability and progress. An execution-tracing mechanism "smart" enough to back up over such branches is needed to satisfy student requests for repetition. The student must be able to interrupt the session at any point and resume at the point of interruption with backtracing capability intact.
3. Information about the student and student progress may need to be saved for use throughout the course. Mass-storage access is required for this and the backtracing facility.
4. An optional tutorial on the use of the student's CAI environment itself is part of the base package. It is undesirable for the student to begin study of the course material while still unfamiliar with the use of the medium.

It is a tribute to the inherent power of FORTH that these capabilities are all implemented in six screens, with a seventh "load screen" to pull the system together and implement any portability changes. The design details may be studied by reference to the glossary and code that follow.

DISCUSSION

The system has been used to implement a CAI-based distance-education course on the subject of the C programming language. The simplicity of design proved valuable: The course author had many other responsibilities and the production of the C course was spread over eleven months with lengthy interruptions; yet the resumption of course-writing was never delayed by the need to relearn a complex set of tools. The need for one refinement was identified: the branching words should be made relative, rather than absolute, to aid in screen relocation.

GLOSSARY

Display words

TITLE (---) ?LOAD - must be LOADing
Clears the display screen and displays, centred on the second display-screen line, text from the input stream following the word TITLE till the end of the input line. The display cursor is set to the left column of the third display-screen line.

DSP` (---) ?LOAD
Beginning on the display line beneath the current cursor position, text from the input stream is copied until a line ending with the character ` has been displayed (the ` is suppressed), or till the end of the input screen.

Flow-control words

#IF (f ---) ?LOAD
If the value of f is nonzero, no branch takes place. If f is zero, input words are read and discarded up to and including the first occurrence of the word #FI. May not be nested.

#IFN (f ---) ?LOAD
Negative of #IF; no branch if f is zero, skip through #FI else.

#IFDEF (---) ?LOAD
Reads a word from the input stream. If the word can be found in the FORTH dictionary, no branch takes place; otherwise input words are read and discarded up to and including the first occurrence of the word #FI. May not be nested with other #IFs or #IFDEFs.

#IFNDEF (---) ?LOAD
Negative of #IFDEF; branches if the word is found.

LINK (blk# ---) ?LOAD
Unconditional branch to new input screen. Continues LOADing at the start of disk block number "blk#". Does not nest LOADs so there is no return to the branch point.

Tracing words

TRC (Variable) (An array of 128 block-numbers with an initial index)
Used as a "hollow-bottomed" stack, i.e. the 129th push causes the first (bottom) item on the stack to be discarded.

>TRC (blk# ---) Pushes "blk#" onto the TRC stack.

TRC> (--- blk#) Pops a block number off the TRC stack. If the stack is empty, the number in system variable BLK is used.

FORWARD (blk# ---) ?LOAD
Pushes the contents of system variable BLK onto the TRC stack and LINKs to block number "blk#".

BACK (--- blk#) ?LOAD
Pops a block number off the TRC stack and LINKs to it.

ADVANCE (---) ?LOAD
Pushes the contents of BLK onto the TRC stack and executes -->.

>ADVANCE (---) ?LOAD
Takes the contents of BLK, subtracts 1, pushes the result onto the TRC stack, and executes -->. Used when the current input screen is a continuation of a display that started on the previous one.

Start/Resume words

MENU-BLK (Constant) Block number of the main menu display.

INSTR-BLK (Constant) Starting block number of a short tutorial on how to use the CAI system for study.

SAVBLK (Constant) Block number of the start of the CAI data-storage area. TRC is saved here between sessions.

SAVET (---) Saves TRC at SAVBLK, signs off and exits from FORTH.

RESUME (---) Restores TRC from SAVBLK. Pops TRC and LOADs the block the number of which is popped, then exits from FORTH.

SETSAV (blk# ---) Initializes SAVBLK's TRC space with a single entry, block number "blk#".

MENU (---) Clears out the TRC and LOADs the menu screen at MENU-BLK, then exits from FORTH.

LEARN (---) If the SAVBLK copy of TRC is empty, executes MENU; else displays "Resume where you left off (Y/N)? ", gets a keypress, and executes RESUME if it's an upper- or lowercase Y; else executes MENU.

STARTUP (---) Displays a signon message, followed by "Would you like instructions on the use of this system? Press the Y key for 'yes' or the N key for 'no':" The program then loops till either a Y or an N (either case) is keyed in, displays the key, and if it was Y saves INSTR-BLK in SETSAV's TRC space and RESUMEs; otherwise executes LEARN.

User-interaction words

?ESC (--- n) A shell for KEY that traps the Esc character and exits via SAVET is Esc is pressed.

PAUSE (---) Displays "Press RETURN to continue, UP-ARROW to review" and loops till one or other is pressed. If it's UP-ARROW, BACK is executed.

?LINK (n1 n2 n3 ... n_i i ---) ?LOAD

Allows the user to enter a number between 1 and i where i is in the range [2,10]. The values of n_i are popped from the parameter stack and the one corresponding to the user's choice is fed to LINK.

EOS (---) A shell for PAUSE, defined as CR CR PAUSE CR ;

SAMPLE SCREEN

```
Screen #15
    0 TITLE MENUS
    1    DSP` Sometimes I'll ask you to choose from a number of
    2 options, like answering a multiple-choice question.  When
    3 that happens you'll have to enter a number (one of the keys
    4 on the top row of the keyboard) corresponding to the option
    5 you want to select.  After you press the number key, you
    6 can either confirm your selection by pressing RETURN or you
    7 can change it by pressing the BACKSPACE key.
    8
    9 Here's an example:
   10
   11                        1 = go on
   12                        2 = go back
   13
   14 Try option 2 first, then option 1:`
   15 BLK @ >TRC   16 13 2 ?LINK
```