Concepts underlying the development of modules

DOUGLAS B. EAMON

University of Houston—Clear Lake, Houston, Texas

The need for common pedagogical and design features in courseware is discussed. A particularly useful feature would allow use of courseware in either of two "modes": as an "electronic chalkboard" for the instructor who wishes to take advantage of the computer in a traditional lecture setting, or as an "electronic study guide" in which students interact individually with the programs. Development of programs using these two modes can be facilitated using Author-Assist, a set of routines and a program skeleton being developed by CONDUIT (1983). Author-Assist provides procedures for use of several types of automatic menu generation, automatic text formatting in "text ports," mixed text and graphics, protected input routines, lowercase characters in text, automatic scaling of graphic displays using "graphports," on-screen calculator functions, and a variety of other features.

Users of microcomputer software in psychology are aware of a wide range in quality, usability, and structure of available programs. Some programs appear to be well protected against user errors, whereas others "crash" unexpectedly due to entry of numbers outside the ranges required by the program. Some clear the screen before continuing to the next entry; others allow text to scroll as it reaches the bottom of the screen. Some employ standardized routines for "paging" through the materials ("Press RETURN to continue"); others proceed at whatever pace the programmer has deemed appropriate.

The instructional objectives of the program should, of course, be the primary consideration in courseware design. Nonetheless, a student interacting with a variety of different programs by different authors will need to determine the various operational features of each of the programs in order to attain the objectives. To the extent that learning this additional information is not directly related to the objectives of the programs, the student is faced with a more difficult task than the designer of the programs may believe. This may be particularly confusing when several programs are used in the same course. One student described her reactions to the programs I used as similar to reading a book in which the print was of various shapes and colors, the pages of different sizes, and each chapter was written by a different author unknown to the others.

The modules under development attempt to rectify some of these problems by employing several common features. One purpose is to create a set of programs that will allow the learner to move from any one to the others with little interference and without requiring

that he/she learn an entirely new set of conventions in each case. In addition, it is expected that the availability of the routines that implement these common features will form the beginnings of a set of stylistic conventions others may find equally useful.

PEDAGOGICAL DESIGN

Three features shared by the modules form their common pedagogical basis. The concepts of horizontal integration, vertical integration, and the "electronic chalkboard" are described by Castellan (1984) and Spivey and Jackson-Smith (1984). The third feature, the use of the modules as an "electronic study guide," needs further elaboration.

The "electronic study guide" complements and extends traditional textbook supplements in a number of ways. Important terms are keyed to several popular introductory textbooks so that the modules can be as useful and as general as possible. Textual material is presented in a controlled sequence that reflects the success of the student attempting to learn it. Presentation of materials for each important new concept is preceded with a pretest, with review of old materials provided if appropriate (since the programs are menu driven, however, the student may skip portions of the materials he/she has already covered). Particularly difficult material may be presented in a variety of ways simultaneously; for example, a figure may be drawn or filled while appropriate text is presented. Demonstrations or simulations of important concepts and results are introduced at appropriate points in the learning sequence, thus avoiding a wait for the classroom or laboratory period. Some modules may also collect data from members of the class, which later can be used as the basis for demonstrations or analyzed by the instructor as illustrations of the concepts being learned. Testing is done at various points in the presentation, and review

Preparation of this paper was supported by a grant from the University Available Fund at UH—CL and by a FIPSE grant to CONDUIT. Reprints of this paper are available from Douglas B. Eamon, Box 108, University of Houston—Clear Lake, 2700 Bay Area Boulevard, Houston, TX 77058.

of selected materials is based on the student's mastery of them. Particular attention is given to anticipating possible errors the learner might make and providing appropriate feedback in those cases. In summary, the electronic study guide is a blend of traditional and non-traditional instructional strategies in which such strategies are introduced at critical and appropriate points in the instructional sequence.

The remainder of this paper focuses on the implementation of the electronic study guide using a program skeleton and a set of software routines particularly suited to that purpose.

PROGRAM STRUCTURE AND DESIGN

The systematic approach to the development of the modules included: (1) identification of an appropriate content area (see Butler & Kring, 1984), (2) preparation of a detailed outline of the content and concepts to be presented, along with identification of the particular pedagogical techniques to be employed (e.g., drill and practice, demonstration, simulation), (3) development of a complete set of screen-display "frames" and their presentation sequence, and (4) BASIC coding (see Spelt, 1984), for a detailed description of the development of a particular module). Of primary interest here is the organization and coding of the modules.

Clearly defined conventions have been followed in the programming of the modules. Many of these are implemented through a set of subroutines and a program skeleton called AuthorAssist currently under development by CONDUIT (1983). AuthorAssist is a revision and extension of "User Frontend" (Hirst, 1979) developed earlier by CONDUIT for mainframe computers. In addition to providing the appropriate subroutines, AuthorAssist also suggests a set of stylistic conventions easily supportable using the system.

The programming language is BASIC. Although the current modules are being written in Applesoft, we use a "generic" version of BASIC that is easily transportable to other microcomputers. Commands peculiar to specific microcomputers or versions of BASIC are not used (e.g., the four "Ps": PEEKs, POKEs, POPs, and pixel references).

The program skeleton (see Figure 1) assumes that the instructional content of the modules consists of a series of "frames" or screen images. Such frames may be either text or graphics, or mixtures of the two. The entire program is thus a means of presenting this set of frames in an appropriate sequence.

Figure 1 shows a division of the program into a portion supplied by AuthorAssist (left side) and a portion written by the module programmer. Standard line-numbering conventions will be established for each of these components. (Because AuthorAssist is still under development, the particular order in which these components appear in Figure 1, as well as the line-numbering conventions, may be altered in the final version. The

Figure 1. Program structure provided by AuthorAssist and by author/programmer.

underlying concepts of the program organization and the components shown in the figure, however, are still expected to be present.)

The subroutines for initialization of AuthorAssist set up the computer to take advantage of AuthorAssist's features (i.e., establish protected memory areas for machine language routines, initialize the character generator, define certain variables used by AuthorAssist). Subroutines supplied by AuthorAssist are partly in machine code and partly in BASIC. Calls to the machine code are generally implemented using forms of the "&" command (for Applesoft).

Any additional special initialization needed in the module is also isolated from the remainder of the program in order to encourage transportability.

The "frame manager" determines the pedagogical organization by sequencing the frames. In general, this portion of the program will consist of a series of calls to the frame subroutines. Included in the frame manager will be an examination of learner input returned from each frame. Requests for help, to quit, to go back to the previous frame, to go back to the beginning of the lesson, to obtain additional information or feedback, etc., are used to control the frame sequencing. In addition to its structured programming character, a useful feature obtained by isolating this portion of the program is that it allows the development of the module to proceed using "stubs" ("dummy" routines that hold a position in the program, but that perform little real "work"). Thus, the module developer is able to examine the organization and sequencing of the frames prior to writing each of them in detail.

The frames themselves actually present the materials,

obtain and evaluate the learner's responses, and return appropriate values to the frame manager, where they are used to select the next appropriate frame. Most of the AuthorAssist routines are called from these frames, and it is here that the real strength and versatility provided by AuthorAssist becomes most apparent.

Although the following list is not complete, some of the more interesting and powerful features of AuthorAssist are described below.

Protected Input Routines

A frustrating and unfortunately common feature of many instructional programs is that they allow the student user to enter numbers or strings that will later (or sometimes immediately) cause the program to "crash." AuthorAssist provides protection against this possibility through the use of routines that examine input for type (integer or real), range of value, and length of the entry. If the user attempts to enter values outside the parameters specified by the module designer, the routine prompts the learner with the range or type of acceptable values and asks that it be reentered.

Character Generator

Like many other microcomputers, an "off-the-shelf" Apple II or Apple II+ is unable to present lowercase letters on the screen (the Apple IIe represents a considerable improvement in this respect, but still has limitations). Although several hardware modifications are available to correct this deficiency, many users have not installed them. Such characters are often desirable or necessary in instructional modules.

AuthorAssist includes a character generator capable of displaying upper- and lowercase letters, Greek alphabetic characters, a variety of special symbols, and additional graphics characters. The AuthorAssist character generator, unlike many others, is compatible with the complete ASCII-defined character set.

Programming lowercase in text requires the programmer to use a machine that has been modified to accept the normal shift-key function (in the Apple II or Apple II+); presentation of either special symbols or graphics requires specification of the ASCII value of the character.

All printable characters may be displayed in any of five "renditions." These include normal video, underscored, double underscored, reverse image (Apple's "inverse"), and boxed (encased in a box one line high).

Text Ports

Information printed to the screen is displayed in "text ports." Two text ports are defined by AuthorAssist: the upper 22 lines, and the lower 2 lines. The upper 22 lines are the primary text area. The lower 2 lines are normally used as a "status" area to present non-instructional messages to the user: for example, "Press Return to continue," error messages when the user has

entered values outside the requested range of values, or brief menus.

In addition to these, the programmer may define any number of other screen areas as text ports simply by providing an identifying number for the port and the upper left and lower right corners. Such additional text ports may overlap previously defined ports. Display of text on the screen will always occur within the most recently defined text port; AuthorAssist will automatically break sentences at word boundaries so that it is easy to display program-generated text without having to worry about its on-screen appearance.

The programmer may also specify left and/or right justification or centering of text within the text-port display.

Menus

AuthorAssist provides several programmer-selectable routines for menu generation. A "standard" menu might consist of a list of the options available preceded by numbers; the user enters the number corresponding to the item he/she wishes and presses the Return key to continue. An alternative menu routine uses a "bullet" (filled-in "o") that the user moves up and down beside the menu items by pressing the left (up) and right (down) arrow keys; when the bullet is adjacent to the user's choice, the Return key is pressed. A "box menu" routine works similarly to the bullet menu, except the menu item is indicated by being encased in a box; the user moves the box from item to item with the arrow keys and selects his/her choice with the Return key.

A special "status area menu" option is also incorporated in AuthorAssist. This routine may be useful when the programmer wishes to leave the primary text area unchanged, but allow the student to select continuation options or enter information while examining the current text display. The routine presents up to 15 items (depending on the lengths of the character strings of the items) in the status area. The bullet marker is moved left or right using the arrow keys; selection of an item is made by pressing the Return key.

The important point here is that selection of the particular display by the programmer is done simply by entering the menu items and codes indicating which sort of menu is preferred; AuthorAssist presents the items in the selected format and obtains the user's selection performing appropriate checks for range of entry.

An interesting feature of the menu routines is the ability to include a "hidden" item not displayed on the screen. Such items might provide access to special routines that should be available only to the instructor (e.g., data on student performance or routines for modifying program parameters). Selection of these items is done by entering a "password" corresponding to the hidden item; the menu routine will "beep" when each character of the password is pressed (to discourage the student from entering the hidden password and to dis-

guise it), but if all the letters are entered in sequence, the program will select the hidden item.

Graph Ports

AuthorAssist also provides powerful graphics capabilities. The user may identify a portion of the screen as a "graph port" in a way similar to that for a text port. In addition to the screen character positions of the upper, lower, left, and right edges of the graph port, "real-world" boundaries corresponding to each of these positions are also specified. For example, to display a curve showing the relationship of IQ (x-axis) and frequency (y-axis) for a sample of 1,000 persons, the user might specify the real-world left boundary as an IQ of 70, the real-world right boundary as an IQ of 130, the lower boundary as a frequency of 0, and the upper boundary as a frequency of 400. BASIC statements to draw a normal curve can then be entered using these real-world parameters rather than screen pixel locations; conversion to the pixel locations is done automatically, as is scaling the curve to fit the space provided in the graph port. Both horizontal and vertical labeling of the axes can also be done automatically by AuthorAssist.

Any number of programmer-defined graph ports may be identified, and these may overlap text ports or other graph ports.

Animation. AuthorAssist provides for limited animation in graph ports by allowing the user to define one or several shapes that can be moved from one point to another using simple programming commands. The shapes may be scaled for size. The animation routines use an exclusive-or technique of pixel generation that makes the movement particularly smooth and rapid. Many animation routines are relatively slow and appear to "flash" because the entire shape is erased and redrawn for each position shift; the exclusive-or procedure erases and redraws only those portions of the shape that actually change position. In addition, the exclusive-or procedure leaves the screen unchanged after the shape has passed, which allows the shapes to cross over text or other shapes without erasing them. This is a useful and interesting feature, suggesting the possibility of simulating movement in three dimensions.

Function Keys

The programmer may select certain keys to enter

AuthorAssist or user-defined functions. These are normally control-key entries (e.g., control-Q to quit, control-H for help). Some useful possibilities are ESC or control-S to stop the program (a sort of "panic button"), control-B to go back to the previous frame, control-R to review previously presented material, and control-C to enter a "calculator mode," in which the student may use the computer as a calculator to compute information needed to continue with the main frame sequence.

Timers

In addition to these generally usable features, an interval timer and a response timer will be implemented in the version of AuthorAssist to be used for the modules being developed. Both will be accurate to about 1 msec.

SUMMARY

AuthorAssist is a unique and very usable programming environment designed for instructional software. Its program skeleton and programming conventions provide a framework for the creation of structured programs particularly suited to courseware reflecting the electronic blackboard and the electronic study guide. Software subroutines extend the capabilities of the microcomputer to allow easy implementation of module menus, protected input, mixed upper/lowercase text and graphics displays, and a variety of user-defined function-key capabilities.

REFERENCES

- BUTLER, D., & KRING, A. M. (1984). Survey on present and potential instructional use of computers in psychology. *Behavior Research Methods, Instruments, & Computers*, **16**, 180-182.
- CASTELLAN, N. J. (1984). A model for courseware development in psychology. *Behavior Research Methods, Instruments, & Computers*, **16**, 165-167.
- CONDUIT. (1983). *Courseware development using AuthorAssist: Apple II version*. Unpublished manuscript.
- HIRST, S. (1979). *The USER FRONTEND: A generalized input facility for conversational programming*. Iowa City, IA: CONDUIT.
- SPELT, P. (1984). Overview and examples of instructional modules. *Behavior Research Methods, Instruments, & Computers*, **16**, 176-179.
- SPIVEY, J., & JACKSON-SMITH, P. (1984). The electronic chalkboard. *Behavior Research Methods, Instruments, & Computers*, **16**, 172-175.