DOCUMENT RESUME

ED 099 003 IR 001 405

AUTHOR Turner, Ronald C.
TITLE PIRATS*: A Modified COURSEWRITER Plus Graphics for the DEC POP 11/20 (*Personalized Instructional, Remedial, and Tutorial System).
PUB DATE Aug 74
NOTE 18p.; Paper presented at the Annual Meeting of the Association for the Development of Computer-Based Instructional Systems (Bellingham, Washington, August 1974)

EDRS PRICE MF-$0.75 HC-$1.50 PLUS POSTAGE
DESCRIPTORS *Computer Assisted Instruction; *Computer Graphics; Higher Education; Instructional Media; Program Descriptions; Programing; *Programing Languages
IDENTIFIERS Coursewriter; Personalized Instructional Remedial Tutorial Syste; *PIRATS

ABSTRACT

A new system of computer-assisted instruction (CAI), developed at Whitworth College, is proposed called PIRATS. It is said to have the following advantages: (1) it uses an author language; (2) it provides facility for branching and responses to unpredicted answers; (3) lesson text is prepared off-line; (4) pictures can be used to accompany the text; (5) testing is available for student self-evaluation; (6) the system is adaptable to any language; (7) the program does not exceed 8K; (8) the system is downwardly compatible to teletype. The language is described, along with nine programs and programer options. Appendixes include sample graphic work sheets and target code. (SK)

PIRATS*: A MODIFIED COURSEWRITER PLUS GRAPHICS FOR THE DEC PDP 11/20

(*Personalized Instructional, Remedial, and Tutorial System)

Prepared for the August 1974 meeting of the Association for the Development of Computer-Based Instructiona' Systems (Bellingham, WA) by Ronald C. Turner

Department of Mathematics & Computer Science
Whitworth College--Spokane, WA 99251
Phone: 509-489-3550

Two months ago the fifth annual Conference on Computing in the Undergraduate Curriculum met about three hundred miles from here to assess the state of the art of educational computing on college and university campuses. No doubt it was my own bias, but I suspect that those of you who attended that conference may have been left with the firm impression as I was of the steeply rising ascendance of mini-systems in college instruction. The reasons for this are well-known: the attractiveness of time-shared BASIC, the ease of physical installation, and most of all, the dramatic decline in unit prices which we are told will continue for a decade. But although these are primary factors, one breakthrough in systems design has affected just as profoundly every CAI or CMI professional working with mini-systems: the emulation of large system storage through virtual core strategies. It was bound to happen as access times approached and then surpassed the real-world time necessities of users.

So there are a lot of minis around, with a lot more coming down the chute. You certainly already know that from reading issue of Computerworld. I should like to project that statement somewhat and to issue a highly heretical prophecy: I believe that with the proliferation of mini-systems there will occur a concommitant, irrepressible proliferation of CAI author languages for those systems. I've had the evils of reinventing the wheel preached at me at every computing conference I've attended. And I really did read "Point 3" of the pink sheet announcing this conference, whose purpose is in part to "reduce redundant effort among developers." But what NSF, conference planners, and

college administrators often fail to realize is that academic types are notorious for three traits, the composite of which militates against networking, sharing, and other such schemes to prevent redundancy: they are tinkerers by nature, they are extremely self-reliant, and they are usually stubborn. But before leaving this, I'd like to add a corollary to the prophecy: the homegrown CAI author language may prove to be the one essential catalyst necessary to initiate and foster a vigorous CAI program on campus. I should have also added that we tend to be disgustingly egocentric, judging from the appeal that an author language exerts over a faculty-author, who usually finds his greatest personal rewards in writing original material anyway. And for the language itself to have been developed at his own institution, and to have incorporated some of his suggestions into its design makes the bait just that much more tasty. All of this is to say that a reinvented wheel may be better than no wheel at all, and it may even be superior to the one that is mass produced.

It is not my purpose in this report to try to sell you on the virtues of PIRATS; all of you who are active in CAI systems and courseware development are already working with an author language adequate for your needs. Nor do I frankly expect ever to see PIRATS implemented on a large number of college campuses. My purpose in this paper is rather to document a few of the priorities and anti-priorities that were confronted in the development of the language. My hunch is that the needs which motivated PIRATS are highly similar to those of the majority of college

campuses. Our experience over the past two years might be viewed as a grass-roots approach to CAI system development, inasmuch as the system was written and modified to communicate with current needs of real teachers and students. Here then is a list of those priorities as felt and expressed by the users.

1. An author language is essential. Curriculum revision and course development are taken seriously at Whitworth, and faculty as a whole desire to be in command of the material and of the way it is delivered. In such an environment the teacher is much more eager to develop his own courseware than to implant commercially available programs. Naturally, if it were a matter of elevating the computer to the status of surrogate professor and of writing entire stand-alone CAI courses, few faculty would take the bait. But when the computer can be exploited heavily as an adjunct to the classroom in a tutorial mode, then the faculty can immediately view a CAI system as an extension of themselves. With the sort of mind set, an author language is prerequisite to the system.

2. The language must provide every facility for branching, for responses by the computer to predicted and unpredicted answers, for analysis of partially predicted answers, and for all other operations which are felt to be necessary by the authors themselves. Yet the conventions of the language must be simple enough to be learned within two hours and the user's guide sufficiently clear and concise to be mastered by an amateur.

3. The text for CAI lessons must be prepared off-line. At the time the Whitworth computer was acquired, few universities

or colleges had been able to make a go of utilizing a time-sharing mini-system for both academic and administrative applications. Not only does heavy usage tend to degrade response times for both types of work, but the number of terminals available is severely restricted. (We currently have three CRT's and five teletypes available to the students and faculty at large.) It was unthinkable to encourage faculty authors to tie up the few available terminals for the laborious, head-scratching task of frame-oriented CAI module construction. The drab, traditional punched card has proved to be the smoothest way to preserve the equally traditional routing of curricular materials from the teacher's study in draft form, to a typist (probably someone other than the professor), and finaly to the medium specialist (traditionally the print shop; in our case, the keypunch operator and CAI manager). By means of a carefully formulated listing facility within PIRATS, the author can, in the quiet of his study, simply enter proofreader's marks in the listing of his program; he can even locate the erroneous cards, unless he hands that task to his secretary or T.A. Although this system sounds prehistoric in theory, it has proven in actual practice to be very simple to resubmit a deck of cards to the computer, once corrections have been made. Furthermore, this procedure guarantees that the author is working always with a whole listing of his program in a hardcopy form. One further byproduct is that the cards, if they reside with the author at a distance from the computer itself, serve as a cheap and practical back-up medium.

4. The driver program for the text material must include the capability for pictures to accompany the text, either schematic drawings on the CRT screen or randomly-accessed slides to be projected on a separate screen. Whether the author is particularly fascinated by the topic of computer graphics or not, he is almost certain to feel the need for pictures in his lessons. Thus the PIRATS picture options have proven to be a very liberating experience, pedagogically speaking. The old adage of a picture being worth a thousand words may in fact be a rather precise statement of the tradeoff in computer storage and processing time due to the availability of pictures.

5. Computer-generated testing, based on CAI lessons, must be available for self-evaluations by the students.

6. The system in general, and the driver program in particular, must be able to handle text in any language (which can be represented by the type face available). The "default responses" by the CAI system (which we shall describe shortly) must be accessible and programmable by the author.

7. None of the CAI system programs must exceed 8K, the maximum core area available to a single user.

8. The system must be "downwardly compatible" to a teletype. Even though PIRATS is designed to exploit the CRT, it should also be available for use with printed-text-only programs, which can also be run on the teletype.

It should be evident that the pervasive tone of PIRATS is one of simplicity and practicality. At each stage of development a microcosmic exercise in technology assessment occurred. The

question was not "Can this feature be added?" but rather "Is this feature really needed by our authors?" This mentality of simplistic conservatism is, of course, anathema to state-of-the-art buffs, but perhaps the art has finally come of age sufficiently to ask itself some hard questions, even within a conference such as this one. I should like to level my own mini-caliber guns at some sacred cows of CAI technology in order to set the stage for describing the salient features of PIRATS.

I always cringe and nod dutifully when I hear critics bellowing about a particular system or program functioning "merely as a page turner." I should like to raise the following rhetorical question: What is really so wrong about a computerized page turner <u>if</u> the computer can effectively assist in mastering what is on the page and if it can determine precisely when it is appropriate to turn the page forward <u>or</u> backward? I am frankly more bold now to admit that some of my programs <u>are</u> page turners, but I insist that the mode of reading is qualitatively superior to the solitary activity of a confused reader.

My second projectile is closely related to the first: What is wrong with CAI material that is sequenced in a relatively linear manner if that indeed is the nature of the subject matter and if that is the expectation on the part of the student user anyway?

Thirdly, why must college-level CAI, especially if it is to be used primarily in a tutorial function, seek a receding nirvana of semantic analysis, artificial intelligence, and 2001-ish response evaluation algorithms when a healthy slice

of the CAI market is begging for an effective and undisguised machine to perform mechanical detail work in the educative process.

Skeptics of frame-oriented CAI author languages habitually snort at the constraint of predicting responses from the student. I should like to offer my rebuttal by affirming that an experienced professor can indeed predict responses and can write clever, engaging dialogue, even within the constraints of frames with predicted responses.

Finally, we must frequently respond to the criticism from the educational researchers that the alleged effectiveness of CAI is probably a function of the Hawthorne effect. But while this is admittedly a nightmare for the researchers, I feel no compunction in simply recognizing the intrinsic fun of working at a terminal as part of the fall-out process and as a valuable by-product to aid in getting the job done.

PIRATS is similar to COURSEWRITER in that within each frame the statements each bear an operation code. There are eleven such statement types:

RD	"Read	A statement to be read; no response required
QU	"Question"	A stimulus to which the student must respond
AC	"Answer [which is] correct"	An anticipated correct response
CC	"Comment [to this] correct [response]"	Author's comment to the above AC
AI	"Answer [which is] incorrect"	An anticipated incorrect response

CI	"Comment [to above] incorrect [response(s)]"	Author's comment to the above AI('s)
CU	"Comment [to an] unanticipated [response]"	Author's comment to one unanticipated response; may be up to three CU's per frame
LA	"Label"	A locator within a lesson; used when branching
BR	"Branch"	Directs lesson execution to labeled location within the same lesson
BS	"Branch [to] subprogram"	Directs lesson execution to another lesson
BP	"Branch[to] parent [program]	Returns lesson execution back to original lesson, to labeled location within that lesson specified by the author

Before the punched lesson is read into the computer, the author must specify a set of six "default responses." These are the comments to be output by the computer in the absence of specific comments within a frame. For example, the comment "RIGHT ON!" may be provided as a default comment, but the programmer may include a specific comment to a particular response whenever he desires. Of course, these default comments may be in any language desired; this is a real boon to language teachers and it makes the PIRATS driver program an extremely versatile apparatus. Other comments are provided for the standard cases: when the student is wrong, when he is given the correct and alternate correct answer(s) for free, when he hits the carriage return without attempting to answer, and when he has reached the end of this lesson and is about to be handed back to the general CAI monitor program.

A set of nine programs (in BASIC) comprise the PIRATS system. They are as follows:

1. LOADER reads a deck of cards and stores the text field of each card (64 characters in length) in a sequential virtual storage array. It creates a statement table in which all of the statements are identified by type and by location within the array. A label table is also created to permit branching when the CAI driver executes this particular lesson. The "default comments," described above, occupy a fourth array.

2. ANALYZ scans the statement table of the lesson to discover any illegal sequences of statements.

3. LISTER provides the author with a neatly formatted printout of his lesson, with the program title, date, time of day, and page number at the top of each page. Visual frame boundaries and label lines are inserted to enhance the readability of the listing. Sequential numbers are assigned to all lines and statements of the lesson. The author uses the numbers to locate erroneous cards in his deck or to modify his program directly within virtual core using EDIT to change only.

4. EDIT permits the author to modify his program without sending the whole deck again through the card reader. In practice it is more efficient to make major changes (insertions and deletions of whole lines) in the cards and to re-read the deck, since the deck is then a reliable back-up of the program. But EDIT is very useful for correcting minor typographical and spelling errors.

5. PIX is a translator which accepts code from the author

pertaining to the lines of the picture to be displayed on the CRT. That code is translated by PIX into executable BASIC. The picture is then stored as a compiled BASIC program which may be called by any CAI lesson. For each picture, the programmer specifies how much of the CRT screen is to be retained for text. The picture area at the top of the screen is thus cleared during the lesson when the picture is about to appear (unless the author wishes only to modify an existing picture by using another picture program). That picture will remain on the screen until the programmer chooses in the logic of his program to return to the normal "scrolling" mode of the CRT or until he clears the screen for another picture. For each line within the picture, the author specifies a character which is to be duplicated. He provides the row and column coordinates of the origin of the line, the length of the line, and its compass direction from the origin Or he may provide for whole strings (comprising any characters) to proceed from given coordinates in a given direction.

6. HELP is the universal driver which manipulates all of the PIRATS lessons and the student responses. It is the largest program of the PIRATS system. Much of the coding of this program is concerned with overriding the default "scrolling" algorithm of the CRT when a picture is being presented. When a picture is on, text is displayed and student responses are received line-by-line within the text area until the bottom line of the screen is reached. When that occurs, the cursor

pauses for a time which is calculated on the basis of the length of the last message displayed; it then returns to the top of the text area and erases the remainder of the screen. Other programmer options such as keyword search, picture modification from within the program text, and activation of the slide projector, have pushed this program to the limit of its core allocation, and all clever program-condensing strategies have been utilized. (These very useful programming options will be described shortly.)

7. CAI is the monitor which offers the choice of lessons to the student and which then directs the driver to the files of the program selected or which enables the student to exit from the CAI system.

8. TESGEN is a program which is run only once by the author for the purpose of building a virtual storage array which contains pointers to the questions and correct answers he selects from a particular PIRATS lesson. (Of course, that lesson may have been written expressly as a quiz and so all the questions may thus be opted for when TESGEN is run.) The "default responses" for each quiz may be entered by the author to TESGEN or he may simply adopt the "stock" phrases suggested by TESGEN. And he specifies for each lesson how many attempts per question are to be allowed before the question is counted wrong and the correct answer(s) given.

9. QUIZME is the quiz driver run by the student which accesses the files built by TESGEN. For each quiz, the student is informed how many questions the quiz contains, and he may

choose the number of questions he wishes to answer during that session. For each execution of QUIZME the order of presentation of the questions is randomized, and all missed questions are put back into the "unused pile" to be presented again. The student is thus forced to answer all questions correctly. The screen is cleared before each question is presented. At the end of the session, the number of missed attempts, totally missed questions, and average number of tries per question are all reported to the student.

A set of programmer options are available to enable several interesting things to occur at run-time. These options are identified by enclosing a special code letter within cross-hatches (#) and inserting that code at the head of the line of text.

1. #C# clears the screen completely and positions the cursor at the middle line. This is aesthetically desirable when there has been a sizeable number of lines of text rolling by on a full screen. It also prevents the student from seeing material over which he is now to be questioned, if the programmer so desires.

2. #P# informs the driver that a string is to be inserted into a picture. This is a very fast method of altering pictures, since it does not require the program to chain to an entirely different picture program and then to chain back to the driver (as is done in the case of executing PIX-generated programs). Position coordinates are provided with #P# to inform the driver of the origin of the string.

3. #-# informs the driver that the lesson is finished using the picture currently being displayed at the top of the screen and that the normal "scrolling" activity of the CRT may now resume. The picture will thus be scrolled out of view after the cursor reaches the bottom of the screen.

4. #K# is inserted before a predicted correct response to indicate "keyword." This means that the answer will be counted correct if it finds at least the string specified by #K#. (That substring in the student's answer must be preceded by a blank and be followed by a blank, comma, or period.)

5. #S# is the operator which "wakes up" the interface to the slide projector. The two digits following #S# refer to the slides (01-99) in the carrousel. #S# may be used within any statement of text which is to be output to the student.

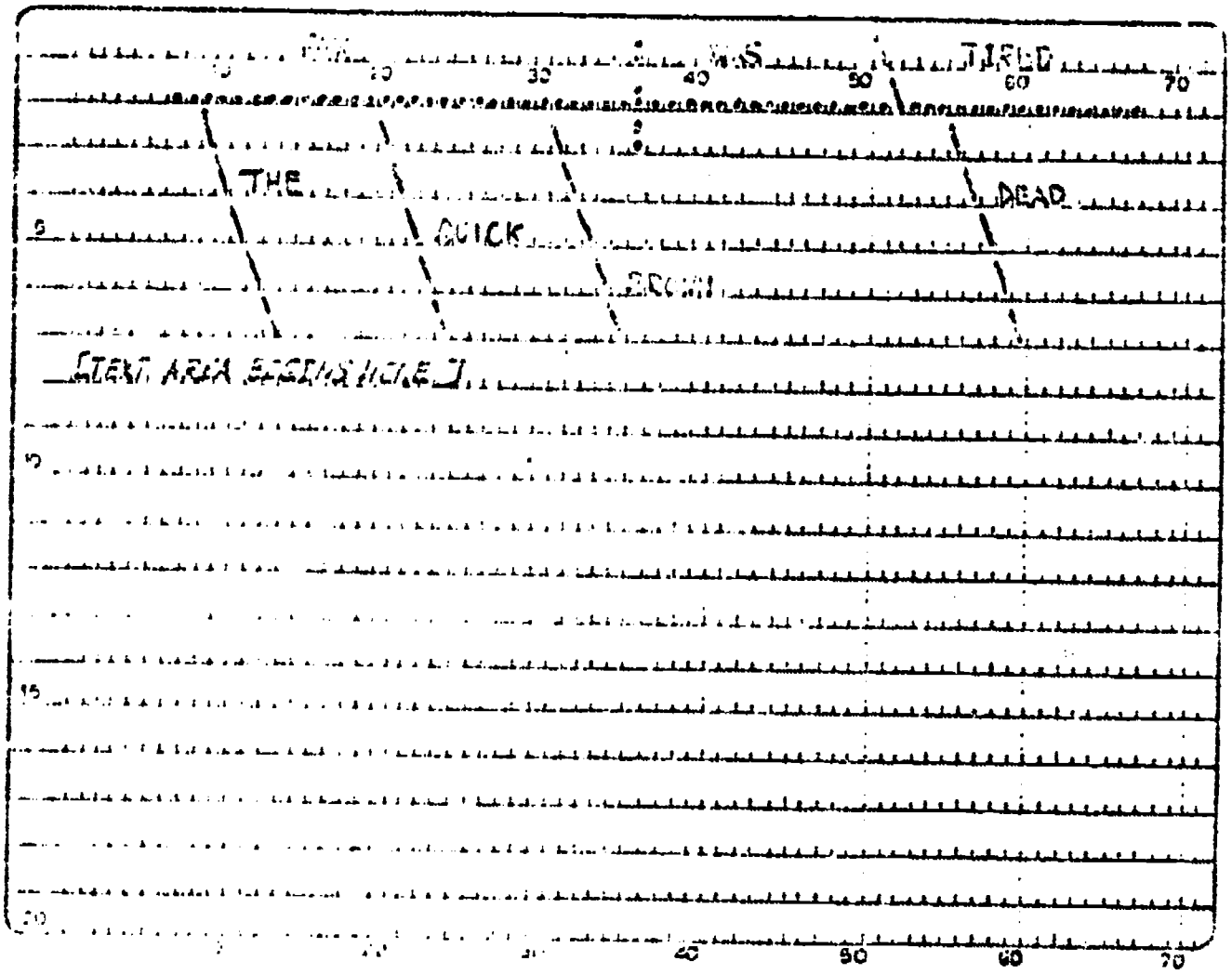
When a student is running a PIRATS lesson he may exit gracefully by typing STOP, and he will be returned to the CAI monitor. He may get an answer for free by typing HELP in response to the question. However, the author may "trap" the student by designating HELP as an incorrect answer within the frame and so force him to try again or branch him to some other location or routine. And should a particular question get scrolled out of view on the CRT, the student may recall it simply by typing QU.

At present there are nine different subject areas in the Whitworth CAI library, ranging from church history and baroque

music to the nomenclature of hydrocarbons. This summer has seen the development of a serious CAI effort in the health sciences revised curriculum. PIRATS has indeed become a way of life within the curriculum of the college, and we would indeed welcome the opportunity to tell you and show you more of the system.

GRAPHICS WORK SHEET

DIAG22



"SOURCE" CODE FOR SENTENCE DIAGRAM

```
DIAG22          [Name of picture]
17              [Number of lines reserved for text area below picture]
CLEAR           [Specifies that picture area is to be cleared]
,,2,7,F,61      [Specifies a line of 61 dots, going eastward from row 2, colum
2,1,34,4,3      [A vertical line of colors (3), from row 1, column 34, so
\,1,51,or,2     [A line of 2 diagonals, from row 1, column 51, going south
\,3,9,5E,5      [A line of 5 diagonals, from row 3, column 9, going south
\,3,31,or,3     [A line of 3 diagonals, from row 3, column 31, going south
\,3,51,or,3     [A line of 3 diagonals, from row 3, column 51, going south
ENDPIX          [End of picture program]
```

APPENDIX III.

"TARGET" CODE GENERATED FOR "DIAG22"

```
 10 C1$=SYS(CHR$(7)):B2$=" "+"
 20 L0%= 13
 30 I$=MID$(C1$,4%,2%):M$=MID$(C1$,4%,6%)
 40 DIM$1%(72%)
 50 PRINT CHR$(29%)
 60 PRINTCHR$(14%);CHR$(I%+31%);CHR$(32%);CHR$(30%)FOR I%=1%TO2J%-L0%
 70 PRINTCHR$(14%);CHR$( 33 );CHR$( 38 );
 80 FORJ%=1%TO 41
 90 PRINT".";
100 NEXT I%
110 PRINTCHR$(14%);CHR$( 32 );CHR$( 47 );
120 FORJ%=1%TO 3
130 PRINT"!";
140 PRINTCHR$(8%);CHR$(11%);
150 NEXT J%
160 PRINTCHR$(14%);CHR$( 32 );CHR$( 82 );
170 FORJ%=1%TO 2
180 PRINT"\";
190 PRINTCHR$(11%);
200 NEXT I%
210 PRINTCHR$(14%);CHR$( 34 );CHR$( 40 );
220 FORJ%=1%TO 5
230 PRINT"\";
240 PRINTCHR$(11%);
250 NEXT I%
260 PRINTCHR$(14%);CHR$( 34 );CHR$( 41 );
270 FORJ%=1%TO 5
280 PRINT"\";
290 PRINTCHR$(11%);
300 NEXT I%
310 PRINTCHR$(14%);CHR$( 34 );CHR$( 42 );
320 FORJ%=1%TO 5
330 PRINT"\";
340 PRINTCHR$(11%);
350 NEXT I%
360 PRINTCHR$(14%);CHR$( 34 );CHR$( 47 );
370 FORJ%=1%TO 5
380 PRINT"\";
390 PRINTCHR$(11%);
400 NEXT I%
410 PRINTCHR$(14%);CHR$(32%+L0%);CHR$(32%);CHR$(31%);
420 C1$=SYS(CHR$(8%)+B2$+CVT%$(L0%)+I$+B2$+"RETURN");CHAIN"DK2:HELP"(L1%
430 END
```