An evaluation of four computer authoring systems

Alice M. Simpkins
Atlanta University
AN EVALUATION OF FOUR COMPUTER AUTHORING SYSTEMS

A THESIS
SUBMITTED TO THE FACULTY OF ATLANTA UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE MASTER OF SCIENCE

BY
ALICE M. SIMPKINS

DEPARTMENT OF MATHEMATICAL AND COMPUTER SCIENCES

ATLANTA, GEORGIA
MAY, 1985
DEDICATION

To

Terrence and Roosevelt
ACKNOWLEDGEMENTS

The author wishes to acknowledge with greatful appreciation the financial assistance that was provided by both the Resource Center for Science and Engineering in Atlanta and Paine College in Augusta.

I wish to thank Dr. Benjamin Martin, my advisor, for his patience, understanding and encouragement during my period of study at The Atlanta University and especially during the development of this project. Thanks, also, to the members of the reading committee for your wise counsel and support.

To the members of my family, especially Mr. and Mrs. Emmett Johnson for the use of their home as my Atlanta residence and my son, Terrence, for talking to me during the many hours that we spent commuting to Atlanta on weekends, I say thank you very much.

Without the encouragement, patience and editing assistance of my colleagues at Paine College, I might have quit long before now. Thanks to each of you for whatever part you played in the completion of this project.

It is with love and honor that I dedicate this work to Terrence, my son, and Roosevelt, my husband.
# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** .................................................. iii

I. INTRODUCTION ......................................................... 1
   Background Information
      Historical perspective
      hardware development
      software development
      Paine College’s computer use
      Reason for topic choice
   Statement of the problem
   Proposed solution to the problem

II. EVALUATIONS OF THE AUTHORING SYSTEMS ............................ 7
   Definition of authoring system
   Evaluation criteria
   Evaluations of four systems
      Teacher Utilities Volume I
      Microinstructor
      Apple PILOT
      Apple SuperPILOT

III. SUMMARY OF FINDINGS .................................................. 30

IV. CONCLUSIONS AND RECOMMENDATIONS .................................. 34
   Choice of system
   Feasibility of using faculty to create software
   Proposal for faculty involvement
   Future considerations

APPENDICES ............................................................. 39

SELECTED BIBLIOGRAPHY ................................................ 53
INTRODUCTION

The increased use of computers in all areas of life—recreation, homes, medicine, business, industry, and schools—has allowed man to increase his knowledge rapidly in all areas where data manipulation is the main focus. Problems whose solutions took hours to obtain manually are now solved in seconds. One of the first devices that contained an embedded computer (a computer which is a part of another device) was the pocket calculator. It was a useful device that immediately gained acceptance in the office and at home.

Graham suggests that "microwave ovens were the first computer-controlled household appliances to appear on the market."¹ The list of other appliances in this category is endless. Recreation enthusiasts led the way for the purchase of personal computers or microcomputers in the home. Their initial acquisitions, game units, were small and inexpensive. We are all aware of the effect that the phenomenal sales of these systems have had on the marketplace. For instance, numerous new jobs have been created. The programs (software) in all of the above devices are built into the equipment and cannot be changed by the consumer.

Medical, business, and industrial applications (word processing, file management, and mathematical computations) all require vast amounts of data (mainly numerical) manipulation. Super or mainframe computers are needed for processing the large volumes of data. In addition, computer-assisted diagnosis as well as control of machines like robots have become very important uses of computers in medicine and industry. The United States government, directly or indirectly, is one of the biggest users of computers. For example, the Social Security Administration, Internal Revenue Service, space program, and the communications industries all depend on computers for efficient operation. The accompanying software for all of these applications is highly developed and expansive since these organizations were the first users of computers and software was initially created because of their needs.

As new technology made computers smaller and more affordable, many major universities gained access to mainframe systems through various means. For instance, "Dartmouth’s liaison with computing became permanent and continuing in 1956 through the New England Regional Computer Center at MIT, sporting a new IBM 704 and heavily subsidized by IBM. ... Before the IBM 704 was installed at MIT, [they] used a similar machine through the courtesy of General Electric at Lynn, Massachusetts."2 University faculties have assisted with the

---

writing of compilers and other software since the beginning of the
development of high-level languages. Major programs in the study of
computer science were also initiated at these large institutions while
smaller colleges and public school systems were left out of this
educational development.

Large scale integrated circuits and microprocessors allowed for
further reductions in the size and increases in the capabilities of
computer equipment. Because of these technological developments,
minicomputers and microcomputers were born. As companies purchased
newer, more modern equipment, mainframe or minicomputers were given to
colleges that could not afford to purchase their own. These computers
were used primarily for administrative data management. Thanks to
persons like Steven Jobs of the Apple Computer Corporation, elementary
and secondary schools are now in the forefront of computer sales,
specifically microcomputer sales. He marketed a small system for public
use that was designed by Stephen Wozniak. Finally, public school
boards of education and small college administrations could afford to
purchase systems for students to use. No longer was computer access
confined to major corporations or school districts with large tax bases
or endowments. Articles in magazines and journals report that
elementary and secondary schools as well as some colleges and
universities have acquired microcomputers as gifts or by purchasing them
at substantial discounts. Four types of software—drill, tutorial,

---

3G. M. Hopper and S. L. Mandell, *Understanding Computers*
simulation, and learning games—have been developed for use in school programs.

Since 1973, the Paine College administrative structure has been committed to assisting faculty and students in using the computer to solve problems and facilitate learning. Acquiring a teletype terminal with funds from the National Science Foundation allowed us to communicate with the mainframe computers at the University of Georgia. Modest use was made of this capability by individual faculty members and their classes. In 1975 and 1976, five Radio Shack TRS-80 Model I microcomputer systems were purchased with college funds and used primarily by students in business administration. Between 1979 and 1982, three Apple II Plus microcomputer systems were added in the Division of Natural Sciences and Mathematics with a combination of federal and private funds.

From 1982 to the present, the College’s microcomputer capability has grown to include a total of 55 systems—mostly Apple IIe and IBM PC models. This acquisition has been complemented by a similar growth in the use of microcomputers by both students and faculty as documented by statistics on file in several offices.

Although their initial applications at Paine were primarily in business, the sciences, and mathematics, computers have been increasingly employed in the humanities and social sciences. Faculty members in the music, psychology, sociology, English and education programs have all found it appropriate to involve their students in computer activities. Locating and acquiring college level software for use at the college level is not easy to do. Most software writers have concentrated on the creation of computer-assisted instruction (CAI)
modules for elementary and junior high school students. Some of this material is appropriate for pre-freshman level developmental courses, but there is a lack of software for upper level college courses. Thus, the need for additional software is obvious. If students are to make use of computers throughout their tenure in college, appropriate software for upper level instruction has to be developed. Because of the upper level software shortage, several faculty members at Paine have requested assistance in the area of software creation. Others have begun to study in this field on their own. Since college teachers can diagnose the needs of college students better than anyone else, the writer feels that it is our responsibility as faculty to lead in the development of this level of discipline specific software.

Generally, the average faculty person expresses an aversion toward learning to "program" the computer using high-level languages like BASIC and Pascal. This aversion is manifested through an attitude of dislike relative to learning the elements of a programming language and the length of time required for performing the task adequately. Thus, a need for a more direct and simpler medium exists.

Since the task of authoring discipline specific CAI materials in a high-level language is not easy, the writer of this paper explored the feasibility of using faculty to develop CAI materials with the aid of one of the available software authoring systems. The project involved determining answers to the following questions:

1. If faculty members at a small college want or need to author their own CAI materials, which authoring system should they use?

2. Is it feasible for small college faculty to author software?

3. How should this effort be organized?
Answering these questions required selecting representative samples of authoring system software for review. Realizing that the faculty members might be using any system that was recommended required the writer to carefully consider the various types of programs that are available. After studying several catalogs and attending a conference on creating software using an authoring system, four of the more than one hundred available teacher utility and authoring system programs were chosen for review. An evaluation scheme incorporating questions that a faculty member who is unfamiliar with the available authoring systems might ask was devised.

Summary evaluations based on the writer's experiences while learning to use each system comprise the nucleus of this paper. It was thought that this method was superior to researching and reporting the results from commercial reviewers since it would allow for focusing on the needs of a full-time teacher instead of those of a full-time software developer. A summary of the findings and recommendations for using each system is included. In addition, a proposal for the systematic training of faculty who are interested in creating CAI materials to supplement their lessons is provided.
EVALUATIONS OF THE AUTHORING SYSTEMS

In this paper the term "authoring system" is defined as a set of programs that teach the computer to understand commands given by a writer as he creates lessons for use in computer-assisted instruction. However, this name and definition are not universal; some authors prefer definitions that more adequately define the intent of their specific software. Terms like "language" or "utility" are included in the names that have been given to such specialized software.

Some of the programs require as much from the user as a programming language like Pascal or BASIC. They use symbolic representations of the commands, many of which must be memorized before any writing can begin. Others use commands that are sometimes very similar to standard English and the program gives the user prompts or reminders of what the next possible step or command might be. Most use a mixture of these.

Therefore, prior to embarking upon the task of creating CAI materials, a new user should assess his needs in terms of the expected outcome and then seek competent assistance when selecting an authoring system. The evaluations which follow are intended to serve as a guide to information about four authoring systems. The questions can and should be extended to other systems until the person has sufficient information to make an intelligent choice from among the many and varied authoring systems that are available today. Those included in this study are:


d. Apple SuperPILOT. Distributed by Apple Computer, Inc. 1982.

Each was chosen because of the features that make it different from the others in the group. The evaluation summaries are presented in order from least capability to most capability. All are commercially available except Microinstructor, which is presently being revised for submission to a publisher. In order to evaluate adequately each system, sample lessons which demonstrate their unique features were created. Printouts of these are presented in the appendices of the paper.

The evaluation scheme consists of a list of questions that were developed from this writer’s experiences as a reviewer of other software. Some of the questions were added specifically for use in evaluating authoring systems. This list is not meant to be exhaustive; however, it should serve as a guide to anyone, but is specifically geared toward the new user.

A. Descriptive Information

1. What type (utility, language, system) of program am I using?
2. What are the system’s capabilities (text preparation, graphics, sound, interactive, non-interactive)?
3. How many and what type of disks are included in the system?
4. Are there any accompanying manuals or helpful information on the disks?
5. Is everything included in the system package or must one purchase additional programs or equipment in order to maximize its use?
6. Is the system command or menu-driven or does one use a combination of the two?
7. How much is the initial cost?
B. System requirements

1. How much memory, minimally, does one need?
2. What is the minimum equipment that one needs to use the system?
3. Does one need any special equipment?
4. Can one use special equipment to obtain special effects (graphics, sound, etc.)?

C. Ease of use

1. How long does it take to get started?
2. How long does it take to learn the commands of the system?
3. How difficult is it to use the editors or to read and understand the manuals?
4. Approximately how much time can one anticipate spending in studying the manuals or other materials prior to writing the initial lessons?
5. Is the student disk self-sufficient or must it be used with a system disk?
6. How well does the system perform when a student is using the lessons?

D. Error handling

1. How are errors corrected during the development stages?
2. How are student errors handled?

E. Versatility

1. Is the structure of the lessons pre-determined by the authoring system?
2. Can test banks of short-answer questions be created?
3. Can test banks of multiple-choice questions be created?
4. Can interactive lessons for review or initial learning be created?
   a. Can the presentation of explanatory text with graphics or sounds be integrated with questions?
   b. Must one always begin lessons from the first frame or question?
   c. Is branching possible?
5. What are the advantages and disadvantages of using the system?

F. Recommendations for use

1. What is the best use of the system?
2. Why is this the best use of the system?
The remainder of this section of the paper contains the detailed evaluations of the four selected systems. Discussion of each begins with a description which is followed by the six part evaluation whose sections are headed system requirements, ease of use, error handling, versatility, recommendations, and sample program. All sample programs were created on an Apple //e microcomputer eventhough versions of the programs are available for other computers where indicated.

Teacher Utilities Volume 1
(Time Saver - Teacher Created Puzzles and Tests)

Teacher Utilities Volume 1, published by the Minnesota Educational Computing Consortium (MECC), is a set of programs (written in BASIC) that allow teachers to create signs in block letters, posters in different sized letters, crossword puzzles, word find puzzles, tests and review exercises for students. Descriptions of and directions for using each program are included on the disk; however, a support manual which contains directions for editing and entering information is also provided. This disk of programs is menu driven, i.e. numbers are typed to make selections from the menu. Three programs, REVIEW, REVIEW LOAD and TEST GENERATOR from this disk are being evaluated.

The ability to create different versions of tests from the same question file is the greatest asset of this program. This is especially helpful when the instructor has large classes. Rearranging the room becomes unnecessary and once the students become aware that their test copies will be different, there is a tendency to prepare more carefully for a test. The REVIEW LOAD program is used to enter the file of
questions; each question has an objective number as well as a question number which allows for ease of editing and question selection. For instance, a file may be created such that the objective number corresponds to the chapter numbers in the textbook; to create a final examination one can select the desired number of questions from each chapter which the computer scrambles so that no chapter has all of its questions together. The computer does this using the TEST GENERATOR program.

In courses where memorization of new information is important, the REVIEW program can be used with the same file of questions to quiz students using either short-answer, completion or multiple-choice format. One can study directly on the computer or get a hard copy of the study questions. Student records can be maintained on disk.

Two other programs on the Teacher Utilities Volume I disk are very useful. The teacher can create crossword or word find puzzles using his own vocabulary list. This is the only "graphics" capability; pictures to accompany the questions cannot be produced using the MECC programs. The cost of $40.00 is inexpensive compared to other authoring systems. Purchase is through MECC.

System Requirements

Using the programs requires at least a 48K system with one disk drive, a monitor, and a printer. They have been used on an Apple //plus and an Apple //e system. Neither the programs nor the questions use a lot of memory. Questions are stored on the disk as they are created and read from disk as needed. Though only one disk drive is required, using two is definitely recommended since swapping diskettes in and out may cause them to become damaged.
Ease of Use

Teachers will find that this utility is ideal for the beginner. Of the four authoring systems that were evaluated, it is the easiest one to use. Instructions are given at each step when using any of the programs and no programming skill is required. Only a little time is needed to read the directions and begin to use any of the utilities. It takes time to enter the questions but it is worth it if they are to be used over and over; good typing skills help, but are not required.

Some minimal testing of lessons created using this system has been done. Performance was superb. Students used the lessons with after one introductory lesson on the components of the computer system. Two problems have been noted to date. Several files containing the questions were inadvertently erased either while a student was using them or during storage. The company indicated that this was caused by using a faulty program disk which was immediately replaced. The other problem is that students need two disks to review a set of questions; the file disk must be used with the program disk. It would be better if students had to handle only one disk.

Error Handling

The creator of the lessons has to leave the "input" level of the program to make corrections in the questions. Unfortunately, any line that needs to be corrected must be completely re-typed.

Student errors are taken to be incorrect responses once the return key is pressed. Students are given one (1) chance to input a correct answer when studying on the computer; then it is displayed. The student must type in the correct answer for practice before the computer will allow him to move on. Failure to do so results in the student being
"hung" up in the program. All missed questions are repeated before the study session ends. Even if they answer the question correctly on the second try, their score is not changed since scores are based on the number of questions the student answered correctly on the first try. They are encouraged to review the lesson again if their score is below the passing score that has been designated by the instructor.

Versatility

The range of uses is small. Using the appropriate programs, one can create short-answer or multiple-choice questions. Essay questions are not allowed if students will be using the file for review since their answers cannot be checked. Lessons that integrate text instruction, branching, and questions for review cannot be developed using the programs.

The main advantage of creating lessons using this system is the ease with which the teacher can create the file of questions and answers. From the student's perspective the main advantage is being able to decide on the number of questions he would like to study ahead of time. There are no disadvantages if the system is used as intended. The lack of graphics and sound capability is not a disadvantage if one is aware of this before beginning any work.

Recommendations for Use

Based on the experiences of one instructor and several students, it is recommended that these utilities be used to create review lessons for students and to test them. Questions that use either a short-answer or multiple-choice format will be the basis for the review.
Sample Program

A sample file of questions with two versions of a test are included in Appendix A. The questions were used to create the two quizzes that appear after them. The objective numbers in the example represent how difficult a question is in relation to each of the other questions. (A one (1) is assigned to the questions that are the least difficult with either a two (2) or three (3) being assigned to those that are more difficult.) The numbers immediately to the left of certain lines in the questions and answers designate the line numbers and are used during the editing process. When the questions are used with REVIEW, either of the given answers is accepted as correct.

Microinstructor: A Microcomputer System for Computer-assisted Instruction

Microinstructor: A Microcomputer System for Computer-assisted Instruction, usually referred to as Microinstructor, is one of the newest Pascal-based authoring systems. The author, Richard Pogue, Ph.D., is in the process of revising this system and it should be on the market soon. However, the writer of this paper had the honor of serving as one of the field testers and decided that it merited being included in this paper because its structure is very different from those of its predecessors.

The system is presently configured with four (4) diskettes -- STUDENT, AUTHOR, INTRO and UTILITY -- and a manual. The manual includes tutorial sections and can easily be used as a reference. The disks are available in 40 and 80-column formats. As a field tester, the
writer feels that the cost ($25.00) for the program is negligible compared to the lessons learned from using it. This may not be the case when the system is marketed.

Microinstructor allows one to create interactive lessons that teach as well as test. One should consider using this system when instruction as opposed to review of material already covered in class or testing is the main intent. Other systems are available that allow for creating files of multiple-choice questions if that is all that is needed. This is the main difference between Microinstructor and easier to use systems like Teacher Utilities Volume I which was evaluated earlier. Fortunately for the lay person, no computer programming experience is required to use the components of the Microinstructor system. This second feature is what makes it different from those that are authoring languages. (Authoring languages will be discussed in further detail in a later section.) Menus are used to select the various options (create a lesson, edit a lesson, etc.) while commands are used to edit the lessons.

With Microinstructor, the teacher is in full control of the type of activity desired for the student. A teacher can include remediating comments in his responses to the student’s answers when creating lessons using Microinstructor, i.e. appropriate responses to anticipated or unanticipated answers to questions can be included so that a student who needs it is given assistance. In addition, helpful hints and a chance to "bail out" are usually incorporated into each lesson.

Minimal graphics that are created using letters or other symbols can be included in the lessons (see FRAME 1 in Appendix B.); more advanced graphics capabilities and music or other sounds are not yet
available. A review of the table of contents indicates that the intent is to add these at a later date.

**System Requirements**

One should be prepared to use a computer system with a minimum of 48K RAM, two disk drives, a monitor, and a printer when developing lessons. Students will need the same equipment minus the printer. Use of this system is not limited to the Apple computer. Versions are also available for some International Business Machine (IBM) personal computer and Tandy Radio Shack (TRS) microcomputer models.

**Ease of Use**

Microinstructor is Pascal-based and uses some commands that are familiar to Pascal programmers. However, beginners can use them without any hindrance as they develop their initial lessons. The system is organized such that only a limited number of the commands are available for use at a given time and occasional beeps warn of incorrect selections. Menus whose terms are similar to English words are used as much as possible.

Prior to booting the computer system to use Microinstructor one should create a carefully planned lesson scheme. This should include, minimally, how much study information is to be included, how many questions will be used for review of the information and how much help should be given to the student as he attempts to move from one question to the next. Dr. Pogue suggests that you "think of yourself as a 'tutor' dealing with a student face-to-face."4 This is the most

---

time-consuming portion of the project.

One can spend a few hours of intense work at the computer and create a lesson that can be used immediately. The emphasis should be placed on the word "intense." One cannot be passive and learn the concepts in a minimal amount of time. The number of hours that are spent in learning to use the system increases in direct proportion to the amount of time that one studies in a less than ideal learning situation. Though this writer had been introduced to the system in a workshop, approximately eight hours was needed to complete the sample lesson in Appendix B from start to finish. This time included learning to use the system since the workshop atmosphere was not conducive to this purpose.

The amount of time that one uses to create the lessons is dependent on the individual. The differences between five basic concepts -- COURSES, LESSONS, FRAMES, STRUCTURE GROUPS and ACTION COMPONENTS -- must be mastered if one is to enjoy the experience of creating lessons using this system. Without this mastery, one spends much time flipping through the manual while trying to decide what to do or use next. It is recommended that one take advantage of the inclusion of the tutorials and practice ahead of any lesson writing.

The system performed well during the learning and lesson development phase. From the student perspective, performance is also good. Being able to include helpful hints in the lessons is an invaluable aid to learning and retention. There are two drawbacks. Both a student disk and a lesson disk are required to run the lessons. In laboratory settings the preference is for single disks for the users of the lessons. The lack of graphics capabilities is seen as a
weakness. When this is added, a powerful authoring system will be in the hands of the user (faculty and student).

Error Handling

Correcting errors in the lessons means using an "editor" with its various components. The Pascal editor has been modified so that it is very easy to use when making corrections is necessary. For those persons who have never worked in an editor environment, the necessity of practice before creating a lesson cannot be over-emphasized. Student errors are handled according to the procedure that is included in the lesson. Sometimes answers are counted as being wrong; sometimes they form a close enough match to be acceptable. The teacher decides which format should be followed.

Versatility

The range of uses is good. Very interactive teaching modules, multiple-choice and short-answer reviews and tests can be written. This is due to the ease of branching to and from different points in the lesson. Persons who are new to the field of creating computer software can use this system with a minimum of initial difficulty.

Recommendations for Use

Microinstructor is recommended as a good program for developing teaching modules or lessons if no graphics or sound capabilities are needed. Other programs that are easier to use should be employed to create multiple-choice or short-answer reviews or tests.

A teacher can use this system without knowing how to program a computer. The author has created a lesson format. However, its versatility allows for some flexibility on the part of the lesson writer. Inclusion of this "structure" is its greatest asset and, thus,
its major advantage over other systems. Remediation for the student is another advantage. The only disadvantage is that the tutorials do not sufficiently prepare one as he learns the relationships between the various components of the system.

Sample Program

The frames from a sample lesson are included in Appendix B. The structure of the lesson appears below. It is impossible to write the lesson until this analysis of the components to be used in each of the frames is completed.
TABLE 1

SUMMARY OF FRAME STRUCTURES FOR MICROINSTRUCTOR SAMPLE LESSON

<table>
<thead>
<tr>
<th>FRAME NUMBER</th>
<th>TYPE OF STRUCTURE GROUP</th>
<th>ACTION COMPONENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3, 4</td>
<td>Information</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Branch</td>
</tr>
<tr>
<td>5, 6</td>
<td>Question</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td>ANA (Anticipated Answer)</td>
<td>Answer (correct response)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td>ANA (Anticipated Answer)</td>
<td>Branch (to next question)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Answer (incorrect response)</td>
</tr>
<tr>
<td></td>
<td>UNA (Unanticipated Answer)</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automatic return to question</td>
</tr>
<tr>
<td></td>
<td>Failure-Giveup</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td>Help</td>
<td>Branch (back or forward)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automatic return to question</td>
</tr>
<tr>
<td>7</td>
<td>Information</td>
<td>Display</td>
</tr>
</tbody>
</table>

Notice the use of words in Table 1 instead of symbols except in two cases. This makes for ease of lesson creation. One does not have to memorize the meanings of numerous symbolic representations. Information "displays" can be formatted to be attractive to the reader. The "display" that is seen by the student after he inputs an answer has to be geared to his response to the question. Thus, the teacher must have the ability to anticipate possible answers to questions ahead of time. This is main reason for using discipline instructors to create
the lessons. Someone who is unfamiliar with the subject matter cannot adequately do this.

**PILOT Evaluation**

Apple PILOT is a Pascal-based authoring language that is based on the latest standard PILOT development called COMMON PILOT. PILOT is an acronym that stands for Programmed Inquiry, Learning or Teaching. The original PILOT was developed in 1968 by Dr. John A. Starkweather and associates at the University of California, San Francisco.\(^5\) Versions of PILOT are available for other microcomputers as well as many mainframe systems. The Apple version contains several commands specifically geared toward taking advantage of the graphic and sound capabilities of the Apple computer. Using PILOT, one can develop a variety of different types of interactive lessons for computer-assisted instruction (CAI). Student records can be stored on the same disk that contains the lessons.

The Apple PILOT Language Reference Manual, and the Apple PILOT Editor's Manual accompany the Author (master and backup) and Lesson disks. This package retails for $100.00. A separate computer tutorial, Apple Co-PILOT, is available from Apple Computer, Inc. for $50.00 and is recommended to the beginner. It assists one in learning the commands and allows one to practice using the instructions at the computer prior

---

to embarking upon the task of creating a lesson.

While high-level programming languages consists of "reserved words", Apple PILOT consists of twenty-two instructions. Each instruction consists of two parts, the instruction itself and the object of the instruction which are separated by a colon. These instructions are used in combinations that are chosen by the author to create the lessons. The four PILOT editors -- Lesson, Graphics, Sound, and Character Set -- are accessed by selecting the appropriate one from the main menu. It should be noted that graphics displays can be created in both the Lesson and Graphics editors. Each editor's menu displays a list of the files that have been created using the specific editor in addition to a prompt line. Prompt lines assist the users by displaying the appropriate set of available command options at each level. Thus, one does not have to memorize all of the commands. However, effective use of the language implies that one will memorize the instruction set at his or her earliest convenience.

System Requirements

Apple PILOT requires a minimum of 48K RAM, a suitable high-resolution video display device and at least one disk drive. To work in author and lesson modes simultaneously requires the addition of another disk drive. Notice the absence of a printer. Since the intent is to learn by using the computer, printed copies of work are not normally needed or used. While it is possible to print copies of the programs, it is not possible to print the result of text or graphics commands to a printer.
Ease of Use

Using an authoring language containing four editors along with two manuals requires many hours of pre-planning. Initial start up is made a little easier with the use of the Co-PILOT tutorial. However, use of the tutorial does not replace the need to memorize the single or double character labels that always begin an instruction.

Learning to use a "language" is never easy. PILOT attempts to shield the user from the aspects of programming in a regular language, but it fails in this respect. Its instruction set is almost synonymous to the reserved words of a high-level language. Thus, though the amount of typing is decreased, the use of one or two letter instructions forces the user into a mode of learning and lesson creation that is very similar to writing programs in any high-level language. As this writer developed lessons using the instructions, use of flowcharts for planning the logic became a necessity; the writer must create the structure as well as the content of the lesson. Herein lies the first problem; teachers generally do not have sufficient time to do the required planning. Further, the version for the Apple computer was written prior to being able to use the shift key for obtaining uppercase letters. A set of special control commands must be used whenever one wants uppercase letters.

Students need only one disk to run a lesson. Since the programmer is responsible for how the computer responds to student responses, ease of use from the student perspective is under the total control of the developer. Some programs are made very easy for students to use; others lack the appropriate programming steps which would make their use easy for the student. The performance of the program is very good. Some
lessons created using the program perform very well; others need revising to decrease the number of problems that a student would encounter. This conclusion is based on this writer's minimal review of several lessons that are reported to have been written in SuperPILOT.

**Error Handling**

During the development stage, errors are corrected in the editor mode which is the same as that used for input. Programmers will not mind this method, but most teachers and others who are new to the world of programming might prefer a simpler method. Student errors are handled in whatever manner the teacher has included in the lesson. If error-handling is planned carefully enough, the student will never be "hung up" in the program.

**Versatility**

The range of uses is almost unlimited. Tests, as well as interactive lesson reviews and teaching materials can be created using PILOT. Until the developer considers the possibility of using other peripherals with the computer system, PILOT can be the language that lets him develop integrated (text, graphics, sound) interesting lessons.

PILOT has many advantages over the previously discussed authoring systems. Its lessons can incorporate the use of graphics, sound effects, and special character sets, in addition to the standard text, problems (or questions), branching, arithmetic computations, and file handling. Using all of the capabilities implies that a programmer will have created a lesson that any student will be anxious to study.

**Recommendations for Use**

Lesson writers with a need to incorporate sound and graphics in their materials could use the PILOT language. Of the programs that have
been reviewed so far, this is the only one with more than superficial graphics capability. When Microinstructor is completed, it will also include some graphics capability. Even with all of these capabilities, however, this language is not recommended for use since an enhanced version, SuperPILOT, is now available.

Sample Lesson

Lessons written in PILOT and SuperPILOT look very similar. Thus, a sample lesson written in the SuperPILOT lesson mode, but utilizing primarily the graphics function is included in Appendix C. Instructions that are not available in the PILOT language are starred (*).

Apple SuperPILOT

Apple SuperPILOT is an authoring language that gives educators a set of powerful program-authoring capabilities. Based on the PILOT language, SuperPILOT provides many benefits in course ware programming that are geared to the complex needs of education professionals. SuperPILOT is compatible with Apple PILOT-developed lessons. It has all the benefits of Apple PILOT, including foreign language characters, "friendly" menu and help screens, and storage of student records. The Apple II SuperPILOT Language Reference Manual and the Apple II SuperPILOT Editor's Manual accompany the Author (Master and Backup), Lesson, and Super Co-PILOT (#1 and #2) disks.

In addition to the capabilities of PILOT, SuperPILOT

a. allows one to use double-size character sets, color text on color background, automatic student record-keeping

b. turtlegraphics

c. control of external devices, such as printers, video disks, and
videotape from within the programs

d. immediate execution mode which allows for experimentation with the instructions prior to including them in a lesson

The lack of a specific structure as seen in both Teacher Utilities Volume I and Microinstructor allows for a wider range of applications and procedures to be developed. This is its greatest asset. Teachers want their students to enjoy as well as learn from any lessons that have been created. Oftentimes these persons are in competition with the wide range of activities for the students' attention. Being able to create lessons that are attractive, interesting, and entertaining requires a great deal of time and planning; but, it is worth it. Unfortunately, as is the case with Apple PILOT, Apple SuperPILOT makes the process only a little easier than writing programs in a programming language.

Apple SuperPILOT retails for $200.00 and can be purchased from Apple Computer, Inc. 20515 Mariani Avenue Cupertino, California 95014. It is presently the most expensive of the systems that were evaluated. This should not be surprising due to its capabilities.

System Requirements

Because of the new capabilities, Apple SuperPILOT requires the use of an Apple II system with a minimum of 64K RAM, one disk drive for lesson mode only or two disk drives for author and lesson modes. In addition, a monitor is required. For some applications one would need a printer, handcontrollers, videodisc, or other peripherals.

Ease of Use

Using the Apple SuperPILOT authoring system which contains four editors along with two (2) manuals is not any easier than using Apple PILOT. Use of A/V equipment requires modifying the AP: (accept point)
and V: (audio visual device control) instructions so that the appropriate interfaces can be made. This requires obtaining the SuperPILOT Technical Support Package from a dealer or using one's knowledge of Apple Pascal to make the necessary changes.

Since Apple PILOT is very similar to Apple SuperPILOT, learning to use it is relatively easy for the experienced PILOT user. Attaining the skill necessary to use some of the new features is not difficult. However, the new user will experience some difficulty. New users must be open to the possibility that they can write "programs" and strictly adhere to the syntax of a language. In addition, more time to modify the system and learn to use the new capabilities will be required.

It is recommended that one study either the Apple Co-PILOT or Super Co-PILOT tutorial before trying to create lessons using SuperPILOT. These tutorials contain modules that let one practice creating parts of lessons; unfortunately, how to create graphic designs or sound is not included. The user must refer to the language manual for a tutorial on these aspects. Each computer tutorial (14 in all) takes approximately 20 minutes to study. Very few people can or will learn how to use the instructions without extensive practice.

Error Handling

During the development stage, errors are corrected in the editor mode which is the same as that used for input. Programmers will not mind this method, but most teachers and others who are new to the world of programming might prefer a simpler method. Student errors are handled in whatever manner the teacher has included in the lesson. If error-handling is planned carefully enough, the student will never be "hung up" in the program.
Versatility

This program is the most versatile of the four that were studied. Mixing the components described in the SuperPILOT introduction enables a programmer to create very elaborate lessons if this is desired. It is amazing how even a little color or a change in the size of the characters can turn a dull and boring lesson into one that is attention getting, interesting and motivating. Since one can create files using either the lesson, graphics, sound or character editors, personalizing lessons is possible.

Getting an accurate estimate of the number of software packages were created using PILOT or SuperPILOT is probably a good idea for another research project. However, based on the software packages that this writer has been exposed to and that were created using the PILOT or SuperPILOT authoring language, it is safe to say that numerous persons have found using the system to their liking. The potential for creating extremely versatile lessons whose performance is very good gives SuperPILOT an advantage over the other systems. There are two disadvantages. To use all of the capabilities requires purchasing some expensive peripherals. Also, finding the time to learn how to interface all of the equipment as well as how to use the program is not something that is easy or the full-time teacher to do.

Recommendations for Use

It is recommended that one consider using this language when there is a need or desire to incorporate either graphics, sound, or the use of other peripherals like printers or video discs into an integrated lesson. One of the other authoring systems should be used if the creator is interested in preparing lessons that include text or simple
branching only.

Sample Program

This writer has not explored all of the capabilities of the SuperPILOT authoring language. However, since the use of graphics is not possible with Teacher Utilities Volume I or Microinstructor, a sample graphics program is included in Appendix C. The program incorporates printing text to the screen and printer simultaneously. This latter capability is not available in the PILOT program. Printing graphic designs to a printer, another feature of SuperPILOT that is lacking from PILOT, is possible if one has access to an Apple Silentype printer.
SUMMARY OF FINDINGS

A comparative analysis of the capabilities, characteristics and components of the four systems suggests that each system has strengths and weaknesses that should be carefully reviewed prior to purchasing and using one to create lessons designed to enhance learning. Table 2 in Appendix D contains a graphic summary of the various capabilities.

Prior to using either system it is necessary to carefully think through the proposed lesson format. This is the first step to determining which program to use. Each could serve some special need at any given time. Some research on instructional design would be helpful during this planning stage. This research could provide a person with some helpful hints regarding length of frames, how to incorporate appropriate graphics or sound and how not to create lessons that add nothing more to a study session than what is gained by reading a book.

Instructors with a need to create short-answer or multiple-choice lessons would be wise to consider using the Teacher Utilities Volume I (or a similar program) which requires very little pre-planning. A set structure is inherent in the program. But, this does not cause problems when one is aware of this ahead of time. A teacher only needs the list of questions and answers and access to a computer. Once the program REVIEW LOAD is in the computer, the typing begins. When allowing the students to use the lessons for review, anticipating all acceptable answers is essential since it is possible to include more than one
answer in the lesson. After typing the questions and answers, printing individual tests or allowing the students to use the questions for review is a matter of choosing to run either TEST GENERATOR or REVIEW. The students who will be using the lessons need to be informed that correct spelling is important.

Teacher Utilities Volume I is the easiest and least time-consuming of the four systems to use when creating lessons. However, there are several reasons to consider using an alternative system. First is the need to have a program disk and a file disk available when students are using the files of questions for study. Secondly, three feedback statements—"VERY GOOD", "THAT'S CORRECT", "NO, THE CORRECT ANSWER IS: ___", and "NO, LOOK AT THE CORRECT ANSWER AGAIN."—are already built into the program; no other responses are possible. Thirdly, when questions are answered incorrectly, the correct answer is given and must be typed in by the student before he is allowed to move on. Pressing return or typing anything else forces the student to stay in one place. Lastly, when a student is working on the computer, there needs to be an escape route when he wants to quit before ending the lesson. The only one in this case is to turn the computer off.

Microinstructor is a step above Teacher Utilities Volume I. The first difference that one notices between the Teacher Utilities Volume I system and Microinstructor is the ability to create lessons that provide periodic, helpful feedback to the students. When students study, they need to work with CAI software that gives some indication of where their thinking might be incorrect when wrong answers are given. Thus, the need is for anticipating all possible correct answers is still
important. In addition, some teachers like lessons that allow the student to study information for its own sake or as preparation for answering questions. Microinstructor allows for either type of lesson development while including the possibility of escaping in the middle of a lesson. Its major drawback is the need to use two disks when running a lesson.

This program is very structured; however, this is its strongest asset. The titles for the structure groups and action components automatically appear and serve as reminders to the writer. These prompts are an invaluable aid to beginners and non-programmers. It is not necessary to write text for each of the groups or components. However, the student benefits from having at least one anticipated wrong answer with an appropriate response included in the lesson as well as the help component. The teacher can use these sections to assist the student in improving his logical thinking skills.

By now you have noticed that no graphic designs (aside from the title page in the Microinstructor lesson) have been included in the lessons. This is a major short-coming when teaching students who have learned by playing video games that computers can be programmed to show elaborate graphic displays. Thus, the need to review at least one other authoring system was evident. The last two authoring systems include graphic capabilities as well as the ability to incorporate sound in a lesson. Their summaries are given together because as indicated earlier, one is an extension of the other.

Using either PILOT or SuperPILOT and given sufficient time, one can create very complex lessons that any student would find enjoyable and interesting. The ability of the user to integrate presentation of
information, multiple-choice questions, short-answer questions, musical accompaniment, and graphics makes these programs the most powerful of the four that were studied. Being able to interface with videodisc players and other peripherals is the main advantage of using SuperPILOT instead of PILOT.

The key word to making use of all of the capabilities is "time." It takes time to learn the commands, to plan the lessons, and to test them. These programs are "languages" in the truest sense. One must be able to think logically and debug with ease. These programs do not give any assistance with either of these aspects of programming. Generally, the administrators at most colleges and universities do not give instructors enough free time to create lessons that require a lot of time. Therefore, they are not recommended for use by beginning writers of CAI lessons.

If one has decided to "bite the bullet" and create CAI lessons to supplement their teaching, it is worth noting that three of the systems—Microinstructor, PILOT and Super PILOT—have tutorial lesson disks available. These provide a good place to start for any individual who decides that Teacher Utilities Volume I does not meet their needs. Otherwise, Teacher Utilities Volume I provides a good beginning as one enters the world of creating computer-based lessons.
CONCLUSIONS AND RECOMMENDATIONS

After creating three mini lessons using three different authoring systems this writer learned that one can gain knowledge relative to the use of an authoring system with a minimum of difficulty and investment of time. Though the time to master each system varies greatly, this should not deter any prospective developer of CAI materials. Moreover, it is feasible to use college faculty as developers of computer software using a computer authoring system. The choice of which system to use is dependent upon the desired lesson outcomes.

The following proposal provides a framework for involving seven (7) faculty members in a series of tasks that would culminate in the creation of three CAI lessons using an authoring system. The plan allows for adding this project to the regular college program and requires only that the team members be released from their other responsibilities for 1/4 FTE. The specific details have not been determined and should not be determined in advance. They should be defined by the proposed CAI Task Force. Small colleges could use the plan as it is presented. Large colleges and universities might select the seven faculty members from the same department and prepare lessons for three courses within the same department. The major steps involved in adding the development of software by individual faculty members at a college are:
1. Delegate one person who is experienced in the use of computers (mainframe or microcomputers) as coordinator of CAI development. This person would be responsible for coordinating all activities of the task force including scheduling workshops, ordering supplies, assisting with the development of the lessons and serving as the liaison person between the CAI Task Force members and the administration.

2. Select six (6) faculty members from different departments based on the following criteria:

   a. Each should be committed to the institution and its mission and goals.

   b. Each should plan to continue to work at the institution for at least two (2) years after the start of the project.

   c. Each should be committed to the idea that computer-assisted instruction has a positive effect on the students who use it.

   d. Pairing of individuals who work in related areas should be possible. This is essential to the success of the project.

4. Appoint a CAI Task Force composed of the coordinator of CAI development and the six faculty members whose primary responsibilities will be to choose three topics, and write, revise and test three comprehensive CAI lessons based on the chosen topics within two years. Appropriate studies to determine the quality and effectiveness of each lesson should be a major part of the development phase.

3. Provide 1/4 FTE released time for each faculty member from their teaching and other responsibilities (advising, committee work, etc.) for two years.

Assuming that a normal load is 12 semester hours of teaching and 10 office hours per week, each faculty member would have at least five hours (3 class hours + 2 office hours) per week to dedicate to this special project and the total available faculty time for the project would be 2520 hours (5 hours/person/week X 7 people X 36 weeks/year X 2 years). This is sufficient time to create 8 hours of good software if one accepts the "rule of thumb" which dictates that it takes 300 hours to produce 1 hour of good software. The other hours would be spent doing research and learning how to use one or more authoring systems.
5. The schedule of activities would be as follows:

**Fall Semester (first year)**

a. Name a coordinator.

b. Name the department representatives and the pairings.

c. Survey available software related to the specific disciplines as the first step to choosing the topics in the spring semester of the first year.

d. Learn to use at least one CAI authoring system.

e. Review the concepts involved in instructional design.

**Spring Semester (first year)**

a. In consultation with the appropriate departments, select 3 topic areas (1 for each pair of department representatives.)

b. Select an authoring system for use in creating the lessons.

c. Design the lessons.

d. Write the first draft of the lesson and the evaluation instrument; test them for validity using faculty in the respective departments.

e. Revise the first draft of the lesson for use in a pilot study with students in the fall semester of the second year.

**Fall Semester (second year)**

a. Select students for the pilot study.

b. Place copies of the lessons on file in the academic computing center.

c. Conduct the pilot study. (Let the students use the programs and evaluate the quality and perceived effectiveness of each.)

d. Collect data via the evaluation instrument.

e. Revise the lessons based on the data collected.

**Spring Semester (second year)**

a. Select control and experimental groups of students.
b. Conduct the study. Some aspects of the study might be:

1) including the use of the lessons on the syllabus as one would any other course activity.
2) requiring the students in the experimental group to use the software.
3) testing the quality and effectiveness of the lessons by possibly giving the same unit test to the control and experimental groups and by administering the evaluation instrument to the experimental group only.

c. If the evaluations are positive, the last step would be to prepare the final edited versions for filing in the computer center and use in the future.

d. If the evaluations are not positive, the lessons should be revised and the study replicated.

6. At the end of this initial phase, the structure of the project could take on one of many possible formats. Since training of new faculty would be required, an on-going training schedule would need to be developed. New faculty could then be paired with an experienced faculty member. Other activities would remain the same.

An alternative plan would be to assign members to the CAI Task Force for a longer period of time. Their major out-of-class activity would be to develop and test CAI lessons.

Recent progress toward attaining the goal of 80% computer literacy by both the faculty and students at Paine College suggests that such a project would add another dimension to its use of computers. In addition, the project would be successful and participation in it would be very rewarding since use of the lessons would improve instruction and a cooperative spirit would be fostered.

The following questions that were asked during the study suggest a need for further study:

1. Which commercially available authoring systems are similar to each of those used in this study?
2. Are there other types of systems that would be more appropriate for use in creating CAI lessons?
3. Does the type of authoring system determine the effectiveness of the lessons that are developed?

4. Is it better to create lessons using a high-level programming language instead of an authoring system?

5. Should colleges employ full-time discipline-oriented programmers to create lessons instead of using regular faculty members on a part-time basis?
APPENDICES
The following questions are contained in a file called PROB1 that was created using the REVIEW LOAD program on the MECC Teacher Utilities Disk. They were used to create the two quizzes that appear after the questions. The objective numbers in this example represent how difficult a question is in relation to each of the other questions. (A one (1) is assigned to the questions requiring one step to determine the solution with either a two (2) or three (3) being assigned to those that require more than one.) The numbers immediately to the left of certain lines in the questions and answers designate the line numbers and are used during the editing process. When these questions are used with REVIEW, any of the given answers are accepted as correct.

QUESTION #1

1 What is the probability of getting a 2 on the top face in a roll of one 2 die?

ANSWERS TO QUESTION #1

1 1/6

QUESTION #2

1 What is the probability of getting an odd number larger than 1 when a fair 2 die is tossed once?

ANSWERS TO QUESTION #2

1 1/3
2 2/6

QUESTION #3

1 What is the probability of pulling a heart from a deck of cards in one 2 try?

ANSWERS TO QUESTION #3

1 1/4
2 13/52
3 .25
4 25%
QUESTION #4  OBJECTIVE #2

1 What is the probability of pulling a face card from a deck of cards in one try?

ANSWERS TO QUESTION #4

1 3/13
2 12/52

QUESTION #5  OBJECTIVE #3

1 What is the probability of not pulling a black card from a deck in one try?

ANSWERS TO QUESTION #5

1 1/2
2 26/52
3 .5
4 50%

QUESTION #6  OBJECTIVE #2

1 What is the probability of getting a number divisible by 3 on the top face in a roll of one die?

ANSWERS TO QUESTION #6

1 1/3
2 2/6

QUESTION #7  OBJECTIVE #1

1 How many possible outcomes are there for a roll of 2 dice?

ANSWERS TO QUESTION #7

1 36

QUESTION #8  OBJECTIVE #1

1 How many ways can one roll a sum of 4 on two dice?

ANSWERS TO QUESTION #8

1 3
QUESTION #9

1 What is the probability of rolling a sum of 4 when rolling two dice?

ANSWERS TO QUESTION #9

1 1/12
2 3/36

QUESTION #10

1 What is the probability of rolling a sum of 8 on 2 dice?

ANSWERS TO QUESTION #10

1 5/36

QUESTION #11

1 What is the probability of picking a blue sock from a drawer containing 2 6 blue, 4 brown, and 5 black socks if you are not allowed to look in the drawer?

ANSWERS TO QUESTION #11

1 2/5
2 6/15
3 .4
4 40%

QUESTION #12

1 An umbrella rack contains 9 umbrellas belonging to different people. What is the probability of selecting your red one from the rack during an electrical storm with all of the lights out?

ANSWERS TO QUESTION #12

1 1/9
QUESTION #13
1 What is the probability of pulling a 10 or 9 or 7 in one try from a deck of cards?

ANSWERS TO QUESTION #13
1 3/13
2 12/52

QUESTION #14
1 What is the probability of getting a number larger than 6 when rolling one die?

ANSWERS TO QUESTION #14
1 0

QUESTION #15
1 500 tickets were sold during a raffle. If you bought 6 tickets, what is the probability that you will win the prize?

ANSWERS TO QUESTION #15
1 3/250
2 6/500

QUESTION #16
1 Two dice are rolled. What is the probability of a sum appearing that is less than 10?

ANSWERS TO QUESTION #16
1 5/6
2 30/36
QUESTION #17  OBJECTIVE #3

1 What is the probability of pulling a
   7 or a face card in one try from a
   2 deck of cards?

ANSWERS TO QUESTION #17

1 4/13
2 16/52

QUESTION #18  OBJECTIVE #1

1 What is the probability of rolling a
   sum of 13 when rolling two dice?

ANSWERS TO QUESTION #18

1 0

QUESTION #19  OBJECTIVE #3

1 The probability of rain today is 1/5.
   What is the probability that it
   2 will not rain?

ANSWERS TO QUESTION #19

1 4/5
2 .8
3 80%

QUESTION #20  OBJECTIVE #3

1 Which of the following cannot be a
   probability? A) 0.78 B) 2/9 C) 0
   2 D) -0.1 E) 175/176 (Type in the
   letter preceding the correct
   3 response.)

ANSWERS TO QUESTION #20

1 D
2 -0.1
QUIZ 1 #1

Write correct answers (in reduced form when appropriate) to each of the following questions. Each question is worth 2 points.

1) How many ways can one roll a sum of 4 on two dice?
   ANSWER: 

2) How many possible outcomes are there for a roll of 2 dice?
   ANSWER: 

3) What is the probability of getting an odd number larger than 1 when a fair die is tossed once?
   ANSWER: 

4) What is the probability of pulling a face card from a deck of cards in one try?
   ANSWER: 

5) Which of the following cannot be a probability? A) 0.78  B) 2/9  C) 0  D) -0.1  E) 175/176 (Type in the letter preceding the correct response.)
   ANSWER: 

(The answers that are given below are normally printed on a separate page from the questions above.)

ANSWERS TO QUIZ 1 #1

1) 3
2) 36
3) 1/3
   2/6
4) 3/13
   12/52
5) D
   -0.1
QUIZ 1 #2

NAME: _______________________

Write correct answers (in reduced form when appropriate) to each of the following questions. Each question is worth 2 points.

1) How many possible outcomes are there for a roll of 2 dice?

ANSWER: ________________

2) How many ways can one roll a sum of 4 on two dice?

ANSWER: ________________

3) What is the probability of pulling a heart from a deck of cards in one try?

ANSWER: ________________

4) What is the probability of pulling a face card from a deck of cards in one try?

ANSWER: ________________

5) What is the probability of pulling a 7 or a face card in one try from a deck of cards?

ANSWER: ________________

(The answers that are given below are normally printed on a separate page from the questions above.)

ANSWERS TO QUIZ 1 #2

1) 36

2) 3

3) 1/4
   13/52
   .25
   25%

4) 3/13
   12/52

5) 4/13
   16/52
APPENDIX B
SAMPLE LESSON DEVELOPED WITH MICROINSTRUCTOR

The following lesson (INTRO) is one of several from a course (PROB) on probability that are being developed for use with a mathematics course. The format that is presented represents how the computer sees the lesson. The student version is almost the same without the prompts that are in brackets or beginning with an asterisk. Generally the frames with branch questions are never seen in their entirety by a student since branch statements are included in the lesson. Note: The lessons are developed in a pre-determined frame format.

FRAME 1

[INFORMATION]
*DISPLAY
DICER CARDS

PROBABILITY

CHANCE EXPECTATION
*BRANCH
2

FRAME 2

[INFORMATION]
*DISPLAY
.LS 2
Card games, horse races, lotteries, and TV shows like "Vegas" have probably done more to increase the study of and development of the principles of probability in recent times. Jerome Cardin, a gambler, was interested in his chances of winning at certain games. He decided to enlist the aid of a mathematician to get answers to some of his questions. He wrote to Blaise Pascal who wrote to Pierre de Fermat. It was during the mid 1600's that Blaise Pascal and Pierre de Fermat began to formally define the theory of probability. They were interested in games of chance.

In this unit you will study the elementary rules of probability and use them to solve problems.

.LS 1
*BRANCH
3
The only pre-requisite knowledge for this unit is the ability to count. Yes! COUNT. It is also helpful if you know how to operate on fractions. Note: The following items (or parts of them) are counted very often during the study of probability.

<table>
<thead>
<tr>
<th>Item</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>faces on a die</td>
<td>6</td>
</tr>
<tr>
<td>cards in a deck</td>
<td>52</td>
</tr>
<tr>
<td>face cards in a deck</td>
<td>12</td>
</tr>
<tr>
<td>sums from two dice</td>
<td>36</td>
</tr>
</tbody>
</table>

Before moving on let's review a little.

Some questions on history are always a good place to start.

Who started the letter writing that later lead to the development of the theory of probability? 

Jerome Cardin, Jerome, Cardin

Very Good !!!

Keep this up and you will have a perfect score at the end of this lesson.
Blaise Pascal, Pierre de Fermat, Pascal, Fermat, de Fermat, Blaise, Pierre

Please give the name of a gambler, not a mathematician.

The name does not appear in the introductory paragraph. Check the name or its spelling and try again.

You need to review from the beginning and please pay careful attention this time.

Three names were used in the introductory paragraph:

1) Jerome Cardin
2) Blaise Pascal
3) Pierre de Fermat

(I hope that his hint is sufficient. Watch the spelling.)

Fermat and Pascal developed the first aspects of the theory of probability during which century?

Almost. Read the question again and be careful.
I can tell that you don't pay much attention to numbers. Your response is too far off base.

[+++] [FAILURE-GIVEUP] [DISPLAY]

Perhaps the difference between the "year" and the "century" is a problem for you. Sorry.

[BRANCH 7]

[+++] [HELP GROUP] [DISPLAY]

Do you really need help? Only one year was mentioned in the introductory history section. Match it with what you know about the relationship between being born in 1949 versus being born in the 20th century.

FRAME 7
[INFORMATION] [DISPLAY]

1) INTRODUCTION TO PROBABILITY

Your brief introduction is over. It's time for the real learning to take place. During this series of lessons, you will be given some information followed by several quiz questions. Please read carefully and try to answer the questions as accurately as possible. Your study of these lessons will help you prepare for the unit test on probability. Other lessons will be added to this disk in the future. The anticipated names are:

1) DEFINE
2) COUNT
3) ADD
4) MUL
5) COND
6) EXPECT

Their beginning frame numbers will be added in parentheses beside the name as each lesson is placed on the disk.
APPENDIX C

SAMPLE LESSON DEVELOPED WITH APPLE SUPERPILOT

A simple graphics program was created the LESSON editor of the Apple SuperPILOT authoring language. The effect of each instruction or command is as follows:

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>r:</td>
<td>Allows the user to enter a remark</td>
</tr>
<tr>
<td>ts;p*</td>
<td>Sends text output to the printer</td>
</tr>
<tr>
<td>t:</td>
<td>Sends text to the screen or printer</td>
</tr>
<tr>
<td>ts;q*</td>
<td>Stops the sending of text to the printer</td>
</tr>
<tr>
<td>g:</td>
<td>Accepts graphic commands</td>
</tr>
<tr>
<td>m</td>
<td>Move (object command)</td>
</tr>
<tr>
<td>d</td>
<td>Draw (object command)</td>
</tr>
<tr>
<td>es#</td>
<td>Erase screen to a selected color (object command)</td>
</tr>
<tr>
<td>c#</td>
<td>Set color for drawing (object command)</td>
</tr>
</tbody>
</table>

*not available in PILOT

Print of lesson ALICE

```
r:This program draws the name Alice:
  ts:p
  t:The name Alice was drawn on the
  :screen.
  ts;q
  g:es4
  g:ic6
  r:THE LETTER i:
    g:m350,100
    g:d350,150
    g:m350,170
    g:d350,180
  r:THE LETTER c:
    g:m450,150
    g:d400,150
    g:d400,100
    g:d450,100
  r:THE LETTER A:
    g:m100,100
    g:d100,150
    g:d150,200
    g:d200,150
    g:d200,100
    g:m100,150
    g:d200,150
    g:d200,150
  r:THE LETTER e:
    g:m550,100
    g:d500,100
    g:d500,150
    g:d550,150
    g:d550,125
    g:d500,125
r:THE LETTER l:
  g:m250,200
  g:m250,100
  g:d300,100
```
### APPENDIX D

**COMPARISON OF CAI AUTHORING SYSTEMS**

<table>
<thead>
<tr>
<th>CRITERIA</th>
<th>MECC</th>
<th>Microinstructor</th>
<th>PILOT</th>
<th>SuperPILOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of system disks</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Accompanying manual(s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Capabilities</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text presentation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>graphics, sound</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>interactive</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Type of operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>command driven</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>menu driven</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memory</td>
<td>48K</td>
<td>48K</td>
<td>48K</td>
<td>64K</td>
</tr>
<tr>
<td># disk to create</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td># disks to run</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>printer required</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>special equipment</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>Ease of use²</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>startup</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>learning</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>helpful manuals</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>editing</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>user corrections</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>interactive</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>System performance³</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Versatility</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set lesson structure</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>question/answer</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>graphics/sound</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>branching</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Overall rating³</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

**Notes:** 1X = yes, 0 = no; ²5 = very easy, 4 = easy, 3 = average, 2 = difficult, 1 = very difficult; ³5 = excellent, 4 = very good, 3 = good, 2 = average, 1 = fair
SELECTED BIBLIOGRAPHY


