A Computer Assisted Instruction Package for Computer Graphics

Robert Normand Dutilly

Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/masters_theses

Part of the Computer Sciences Commons

Recommended Citation
http://scholarworks.wmich.edu/masters_theses/1698

This Masters Thesis-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Master's Theses by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
A COMPUTER ASSISTED INSTRUCTION PACKAGE
FOR COMPUTER GRAPHICS

by

Robert Normand Dutilly

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
December 1982
A COMPUTER ASSISTED INSTRUCTION PACKAGE
FOR COMPUTER GRAPHICS

Robert Normand Dutilly, M.S.
Western Michigan University, 1982

The purpose of this thesis is to:

(1) Present the concepts and significance of Computer Assisted Instruction (CAI).

(2) Demonstrate the usefulness of CAI by the implementation of a system to instruct in the fundamentals of Computer Graphics.

(3) Describe methods used for implementing a large CAI package on a microcomputer.

(4) Development of software tools to assist in the creation of CAI programs.

The paper will also cover the historical perspectives on the growth of CAI, its impact on the educational field and the proper construction of CAI materials. The system is written in the UCSD Pascal computer programming language for the Apple ][ plus microcomputer with two disk drives.
ACKNOWLEDGEMENTS

I am greatly indebted to my Thesis Advisor, Dr. John Herman, for his advice, encouragement and interest during the design and implementation of this Computer Assisted Instruction package for Computer Graphics. And to Dr. Dalia Motzkin-Groper for reviewing my text and making valuable comments on its structure.

Special thanks and recognition go to my wife, Joyce, who gave assistance and understanding throughout graduate school.

Robert Normand Dutilly
INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of “sectioning” the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
DUTILLY, ROBERT NORMAND

A COMPUTER ASSISTED INSTRUCTION PACKAGE FOR COMPUTER GRAPHICS

WESTERN MICHIGAN UNIVERSITY M.S. 1982

University Microfilms International 300 N. Zeeb Road, Ann Arbor, MI 48106

Copyright 1982 by

DUTILLY, ROBERT NORMAND

All Rights Reserved
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark √.

1. Glossy photographs or pages ______
2. Colored illustrations, paper or print ______
3. Photographs with dark background ______
4. Illustrations are poor copy ______
5. Pages with black marks, not original copy ______
6. Print shows through as there is text on both sides of page ______
7. Indistinct, broken or small print on several pages √
8. Print exceeds margin requirements ______
9. Tightly bound copy with print lost in spine ______
10. Computer printout pages with indistinct print ______
11. Page(s) ____________ lacking when material received, and not available from school or author.
12. Page(s) ____________ seem to be missing in numbering only as text follows.
13. Two pages numbered _____________. Text follows.
14. Curling and wrinkled pages ______
15. Other____________________________________________________

University
Microfilms
International
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ............................................................... ii
LIST OF TABLES ............................................................................... v
LIST OF FIGURES ........................................................................ vi

CHAPTER

I. Computer Assisted Instruction ........................................ 1
   Overview ................................................................. 1
   Description and Purpose ........................................... 1
   History ............................................................................ 2
   Advantages and Disadvantages .................................... 3

II. Procedures for writing CAI courseware ............................. 5
   Overview ................................................................. 5
   Design ............................................................................ 5
   Considerations and Objectives .................................... 9
   Development ............................................................. 14
   Evaluation ..................................................................... 16
   Record Keeping .......................................................... 8

III. Design of the Graphics CAI system ................................. 19
    Overview ................................................................. 19
    Constraints .............................................................. 19
    Logical Construction .............................................. 20
    Physical Construction ............................................. 23
    Data File Construction ............................................. 25
    Implementation ........................................................ 28

IV. Development Tools ......................................................... 30

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
LIST OF TABLES

1. Record Description of Master file................26
2. Record Description of Location Data file.........26
3. Record Description of Results Data file.........27
LIST OF FIGURES

1. Development Flowchart..............................14
2. Logical Network of System..........................21
CHAPTER I

COMPUTER ASSISTED INSTRUCTION

Overview

This chapter will describe what CAI is, its purpose and growth.

Description and Purpose

Computer Assisted Instruction consists of the process of a person interacting with a computer to develop or enhance some knowledge or skill. CAI is used by people ranging from kindergarten children to senior citizens. The majority of CAI materials have been written at the pre-high school level. The most significant and wide ranging project is the PLATO system written at the University of Illinois in conjunction with the Control Data Corporation. The PLATO system covers a vast array of topics throughout the Social and Physical Sciences. For further information on the PLATO system see Evaluation of the PLATO IV Computer Based Education System in the Community College (Murphy, 1977).
A number of independent companies, educational institutions, school systems and individuals have and are developing sophisticated CAI materials ranging from Biology to Reading Comprehension. A large amount of CAI materials have been created that is substandard both educationally and technically. This paper will outline the proper methodology for creating CAI courseware.

History

The growth of CAI has proceeded in spurts and at the end of each spurt, has almost died out.

The stages of growth are:

(1) Teaching machines were used in the late 1950s and 1960s, but never became popular or widely used because of the lack of standards and teaching materials.

(2) Many computer aided instruction research projects were initiated in the 1960s through government funding. Although a lot was learned about CAI, it was still considered an impractical way to teach due to the size and cost of computers.

(3) In the mid and late 1960s, minicomputers inspired a spurt of interest in CAI. The techniques became more practical with the lower cost of computers. While CAI techniques were more widely implemented, they still did not become a significant teaching technique. Computer aided instruction was not living up to its reputation and again fell into a state of limbo.
The newest surge of interest in CAI was sparked in 1975 by the introduction of low cost microcomputers. The same rhetoric about CAI being the ultimate teaching technique is again being heard.

Advantages and Disadvantages

The advantages and disadvantages of CAI are numerous. These can be enumerated.

Advantages:

(1) Provides an interactive means of learning so the user can learn at their own pace.

(2) Usually provides an interactive graphics capability to enhance the learning process.

(3) It is usually an effective teaching technique because students are fascinated by computers.

(4) Material can be presented in an orderly fashion and can provide assistance to the student when needed.

(5) The courseware provides a recognition factor to the students.

Disadvantages:

(1) Good courseware is expensive in both time of production and complexity of software.
(2) There has been more poorly written CAI materials than educationally valuable courseware.

(3) Standardized techniques for writing good CAI materials are not widely distributed. This is a leading cause for the poor quality of current software.

(4) CAI courseware is usually not transportable between different computers or even different models of the same computer.

These disadvantages are slowly being solved by using standard languages, popular microcomputers and published techniques on the proper methodology for writing CAI materials.
CHAPTER II

PROCEDURES FOR WRITING CAI COURSEWARE

Overview

This chapter will cover the techniques in designing, developing, testing and evaluating CAI courseware.

Design

When designing a package several important factors should be taken into account.

(1) It always takes much longer to create a good package than originally estimated.

(2) Trivial problems take a lot of time, effort and program code to solve.

(3) The package is never completely done. New material can always be added, questions reformatted and general improvements made to the package.

(4) The user documentation usually leaves out one or more "self evident" but important concepts. This is especially true for the novice.

In general, the steps to be concerned with are:
(1) Establish some general objectives.

(2) Create a rough outline of the material you would like to cover.

(3) Make a decision on the computer system to be used.

(4) Decide on which computer language or which authoring system you would like to use.

The difference between a computer language and an authoring system is that with the computer language you write the computer programs whereas with the authoring system, the programs are put together by the authoring system. A few of the authoring languages are Gnosis, Learn, Tutor and Coursewriter for the larger computer systems. Pilot is the system most often used on microcomputers. The advantages of an authoring system are:

(1) Easier to use than writing programs. The material just needs to be set up and then typed into the authoring system.

(2) A CAI package can be ready quicker than writing programs because there is no need for program design and debugging.
Some of the disadvantages of an authoring system are:

(1) Not as flexible as writing programs. The design of your package can be limited because of the way the system does operations.

(2) Most do not have a graphics and sound capability. One exception to this is the Pilot system which can perform those operations.

The advantages of writing your own programs are:

(1) The programs can be as flexible as you want them to be. The different features of the system can be used.

(2) Easier to create record keeping capabilities to your own specifications. The data file can be more flexible by keeping a progress report with comments for each student.

The disadvantages for writing programs are:

(1) It is a very lengthy process to develop a good systems design and writing the programs.

(2) It can be very expensive in terms of computer cost.
The following steps should be followed in the design part of the courseware.

(1) Establish firm educational objectives.

(2) Analyze the audience that will use the package.

(3) Define the environment to meet these objectives.

(4) Select the computer and peripheral devices to be used.

(5) Select the communications strategy to use.

(6) Build a master design.

(7) Analyze the content of the material.

(8) Begin the overall implementation.

(9) Evaluate the design, content and implementation.

(10) Restructure any of the previous steps.

The last step (10) actually is an iterative process that takes place while improving and changing the design details.
Considerations and Objectives

The analysis considerations and design objectives are the areas to be considered in parallel with the initial design phase. The analyses to be used in parallel with the design phase are:

(1) A functional analysis on the feasibility of the CAI package for the computer and language/authoring system to be used.

(2) An analysis of the tasks to be performed in developing the package.

(3) What is the prerequisite knowledge that is needed by the user?

(4) What type of CAI instructional mode should be used? Drill and Practice, Simulation, Tutorial, Problem Solving or Games?

(5) Should the CAI courseware be developed as strictly straight line or have branching capabilities?

The design objectives are more specific in that they describe many of the concepts and problems to be solved. Hartley in a paper presented at the MACUL Convention described rules for developing software.
Seven Golden Rules For Good Software

(1) Good software is easy and comfortable to use.

(a) Are instructions adequate and clearly stated? Are they appropriate to the reading level of the intended users?

(b) Is the presentation free of grammatical and spelling errors?

(c) Does student use require little or no teacher assistance?

(d) Is appropriate documentation provided?

(e) Are response input requirements consistent? Does the input sequence stay the same throughout the package?

(f) Is the program crash proof? Use of a string variable for input instead of a real or integer variable.

(2) Good software has content suitable for its purpose.

(a) Does the program fulfill the purpose for which its use is intended? Visual aid, tutorial, review, testing.

(b) Are the facts, rules, concepts presented in the program valid?

(c) Is the material in the lessons consistent with current practice—spelling, terms, numerals?

(d) Is the level of difficulty appropriate for the intended user?

(e) Is the reading level appropriate?

(f) Is the presentation friendly and nonthreatening to students?
(g) Is the content adaptable to a range of abilities and learners?

(3) Good software is educationally sound in design.

(a) Is the pedagogical structure based on sound educational theory?

(b) Does the program provide the learner with the objectives?

(c) Is the program content pertinent to the stated objectives?

(d) Does the program provide learning guidance?

(e) Does the program assess performance and not just a percentage correct?

(f) Is the use of graphics/color/sound appropriate? Nondistracting and nonrepetitive?

(g) Is the feedback appropriate? Is there positive reinforcement? An absence of put-downs?

(h) Is the response requirement "friendly"? Does the student have a second chance at missed items?

(i) How does the program handle inappropriate responses - by a "try again" message or by crashing?

(j) Does the student have any control over the number of items? The level of difficulty?

(k) Is there provision for different learning modalities? Seeing and hearing is important for beginners, bi-lingual students and special education.

(4) Good software provides record keeping.
(a) Record keeping is essential and must be more than just the number or percentage correct. Teachers need to know exactly what errors were made on which tasks in order to plan effectively for their students.

(b) Record storage - can the records be accessed at the teacher's convenience?

(c) Are student records clear and concise? Are they available in hard copy?

(5) Good software is well formatted.

(a) Is the material readable by the intended user? Are there blank lines between each line of text? Can upper and lower case both be used?

(b) Is the presentation by pages, not scrolled?

(c) Is the presentation in small, readily understood segments?

(d) Is the screen uncluttered?

(e) Are the response requirements clearly delineated to the user and consistent throughout the program?

(f) Is the pace of the program under user control?

(6) Good software provides adequate documentation.

(a) Does the documentation give a clear statement of what is on the disk?

(b) Does the documentation provide clearly stated objectives?

(c) Is the documentation complete and lucid enough for a first time user of a microcomputer to understand and follow?
(d) Are any special features of the program clearly presented?

(e) Is a clear explanation of the management features and their use given?

(7) Good software is flexible.

(a) Are the entry points varied so not all students must start at the lowest level?

(b) Can more practice be given to those who need it and less given to those who don't?

(c) Is an escape feature provided - no possible "infinite" loops, a way out?

(d) Are modification features provided - can the teacher modify the vocabulary or correct a "bug"?

(e) Can additional lessons be created easily to fit the program format?

(f) Are data updates easily made?

(g) Can the student look at the response, change their mind and erase what is not needed?

(h) Is the use of sound optional?

(i) Can the teacher modify the response format used by the program?

These concepts give the idea of a properly constructed software package. The next section will develop a chart to describe the steps involved.
Development

The development of this type of package can be viewed by the use of a flowchart.

Figure 1 Development flowchart
Several iterations?

Design
Mainline
Data files

Start
Writing

Testing

Rewrite

Documentation

Evaluate

Several iterations?

Figure 1 Development flowchart
Figure 1 Development flowchart

Evaluation

The evaluation model provides the opportunity for the designer and others to determine if the package your team had developed satisfies the constraints and initial optimism.

Several of these have been touched on in previous sections but are worthwhile to restate in this context.

1) Have the instructional goals been fulfilled? Is the content adequate for students?
(2) Have teaching principals been violated? Such as, are lessons and instructions clear and correct and does it provide motivation and knowledge to the student? If not, change those areas in the package.

(3) Is the package technically adequate - "bullet proof", easy to use? "Bullet proofing" is the term used to describe the ability of the program to withstand any user input.

(4) Has an instructional guide been prepared not only with technical material but also the prerequisite skills that are needed?

(5) Are supplementary materials necessary and if so, do they follow educational practices? Such as, is it relevant to the student needs and is it appropriate for the learning objective?

(6) Is the user documentation provided with a sufficient number of clear and concise examples? Do these examples cover all the possibilities?

(7) Is the language appropriate and the grammar correct for the package and all documentation?

(8) Finally, check over the amount of material on the screen? Is it too much? Too little? Distracting or threatening?
Record Keeping

This is an important area for not only the user/student and the instructor but also for the designer of the system. The student and instructor need access to a comprehensive master data file which keeps track of the student's scores and progress. The designer needs a number of low level data files that are connected to the actual teaching routines. These data files are used for keeping statistics on the routines in the package. Such as, the number of errors that were made by the students on each test question, the number of times review modules were used or the actual text of unexpected responses made. These files give the designer the ability to know where to modify the lessons or flow of the package. Consequently, these two types of data files are needed to properly utilize and maintain this system of programs - the educational package.
CHAPTER III

DESIGN OF THE GRAPHICS CAI SYSTEM

Overview

This chapter will describe how the Graphics CAI package was constructed, the constraints imposed on it by the Apple ][ plus microcomputer system, the data files used and the final implementation of the system.

Constraints

The constraints imposed by the Apple ][ plus computer and UCSD Pascal can be summarized as follows:

(1) The size of any one Pascal source program is limited to 17,376 bytes by the system editor.

(2) The amount of storage on a 5 1/4 inch disk is 280 blocks or 140,000 bytes.

(3) The executable size of a program with external library procedures is 39,900 bytes.

(4) The maximum size of a source procedure is 1200 bytes.

(5) The maximum size of any file is 38 blocks or 18,400 bytes.
(6) A maximum of 15 segments or external procedures.

None of these constraints pose insuperable problems although they do require some finesse on the part of the implementer and/or programmer. One solution to this is to subdivide the system into a tree structure. The details of this will be discussed in the next sections.

To counter these constraints are several distinct advantages.

(1) The string manipulation capabilities of Apple Pascal.

(2) The random access capability for data files.

(3) The ability of Apple Pascal programs to chain to one another.

(4) The capability of making modifications to the system library and being able to create a library to be used by the package.

Logical Construction

The Graphics CAI package is designed in a tree structure which utilizes the Library facility to the greatest extent possible. The main program of the package is the root of the tree with a number of separate
programs (modules) that are chained to from the main program. These level 1 programs have the ability to chain back to the main program, laterally to a program (module) on its same level or down to a lower level program (lesson). The level 2 programs (lessons) can chain back up to the module or chain laterally to another lesson. The library on level 3 is pre-linked to its parent lesson. Each lesson also has access to the data files for update of the users progress. The tree can be visualized using this diagram.

![Logical network of system](image)

**Figure 2** Logical network of system
Each level has the ability to form this network with the exception of the last level, the Libraries. As noted, this lowest level is actually linked to its parent, the Lesson. A module is an intermediate routine that routes the user to the correct lesson. A lesson is a program that directs the user to the correct frame, deciphers user responses and updates the data files. A unit is a number of frames within the library that is accessible via the lesson. There are two units per library each with five to ten frames. A frame is one small portion of material that the user learns. Units will be further discussed in the next section on Physical Construction. The data files shown in Figure 2 are located in the root. The data files for users/students/instructors and the low level data files for the designer/programmer are located on level two. The data files are accessible from level 2 for updating purposes.

This logical construction gives a feeling for the flow of the system and how the different levels interact with each other.
Physical Construction

As previously noted, the design of this system is modular in form using a tree structure. The main program is set up as a turnkey system. This means that the main program is automatically executed when the power is turned on. The purpose for this is to "make life easier" for the user.

The main program interacts with the user and the Master Data file. Users are required to type in their first and last name to identify themselves to the program. This is then used to recover the users progress from the Master file or to create a new record for a new user. A menu is then displayed which gives the user the following options:

(1) Description of the system.
(2) Start with the lessons.
(3) Review previous material covered.
(4) Optional material review.
(5) Display the users progress.
(6) Leave the system.

Options (1) and (5) keep the user inside the main program. Whereas options (2), (3) and (4) enable the user to leave the main program and enter the network.
structure. Option (6) returns the user to the query screen where the name of the next user can be typed in. After the network has been entered, the only contact the user will make with the package is through the frames within the Libraries. The other programs will be transparent, not visible, to the user. The reason for this is the fewer programs the students have to interact with the less frustration they will encounter. This means that if the student only has to interact with the material to be learned, the more the student will learn and with fewer frustrations.

Since the user is interacting with each frame by typing in a variety of responses, this will enhance the learning situation. The definition of a frame is a specific piece of knowledge that teaches the student about the material. The important idea to remember is that only one concept is presented per frame. Graphic frames, those with pictures, are interspersed throughout each lesson. This gives some tangible feeling to a concept that was presented on a text frame. Some type of response is read in each frame but these responses are usually deciphered in the lesson. By having a response routine in the lesson versus having that same routine in the library involves some tradeoffs. Having the response routine in the Library increases the flexibility of the
system especially in test frames. The disadvantages though outweigh that advantage. The overhead cost of duplicating that routine in each Library is high and having a limited amount of room in each Library makes adding overhead procedures very costly. The test frames though do decipher the responses within that frame. The reason for this is the ability of the test frame to call a review frame if needed. Overhead routines are classified as procedures that do not contribute to the student's education but are necessary to have the package run efficiently. Some of these routines are: response routines, message routines, conversion routines, etc. So these routines, even though necessary are very costly in terms of program size and execution time.

Data File Construction

Alluded to in the previous sections were the data files used in the package. The two types of files were the Master files for the user/student/instructor and the Data files for the designer/programmer. The Master files are made up of three files, one being the Master file that contains the name of each student and a record number that points to the two student data files. One data file contains the Module number, Lesson number and Frame number the user is in. The other has the student's
progress on each lesson. The record in the Master file is described by the following fields and positions:

Table 1. Record description for master file

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Number</td>
<td>1 to 3</td>
</tr>
<tr>
<td>Name</td>
<td>4 to 23</td>
</tr>
</tbody>
</table>

The records in the data files are described by:

Table 2. Record description of Location Data file

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Number</td>
<td>1 to 2</td>
</tr>
<tr>
<td>Lesson Number</td>
<td>3 to 5</td>
</tr>
<tr>
<td>Frame Number</td>
<td>6 to 9</td>
</tr>
</tbody>
</table>
Table 3. Record description of Results Data file

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiz 1</td>
<td>1 to 3</td>
</tr>
<tr>
<td>Quiz 2</td>
<td>4 to 6</td>
</tr>
<tr>
<td>Quiz 3</td>
<td>7 to 9</td>
</tr>
<tr>
<td>Quiz 4</td>
<td>10 to 12</td>
</tr>
<tr>
<td>Quiz 5</td>
<td>13 to 15</td>
</tr>
<tr>
<td>Quiz 6</td>
<td>16 to 18</td>
</tr>
<tr>
<td>Quiz 7</td>
<td>19 to 21</td>
</tr>
<tr>
<td>Quiz 8</td>
<td>22 to 26</td>
</tr>
</tbody>
</table>

The lesson file contains data on each lesson, which questions were incorrect, any unrecognizable responses and any other problems that occur. The data files used in a CAI package are very important because without their proper use a student cannot have an adequate idea of their progress.
Implementation

The implementation of a CAI system depends solely on its proper design. In the Graphics CAI package, the design of the system went through several iterations until a successful one was developed. A large number of libraries were created, one for each lesson. Each library contained two units with each unit having 5 to 15 frames or procedures. The libraries themselves were not fully utilized because the Lesson program that is linked to the Library can be of a certain maximum size. Usually two units in the Library was the maximum possible for the size constraints on the program.

The process involved in setting up a new portion of the system is:

1. Creating a new unit to go into a Library of procedures. Compile this new unit to create the object code.

2. Execute the APPLE3:LIBRARY program.

3. Create a NEW.LIBRARY, copy the old Library into NEW.LIBRARY.

4. Copy the new Unit into the NEW.LIBRARY using the "N" command and leave the Library program.
(5) Delete or change the name of the OLD.LIBRARY and change the name of the NEW.LIBRARY to the name of the previous Library.

(6) Modify the Lesson program to access the new Unit in the Library and add in the new frames to be used.

(7) Compile the Lesson program which also uses the Units in the Library that was just created.

(8) Use the L(ink) command to link the Lesson program and the Library together which creates a very large Lesson code file. The linking procedure involves typing the name of the lesson, the name of the library and then the name of the output file, usually the lesson.

(9) Execute the entire system to determine if all programs and procedures are working properly.

These are the basic steps involved in implementing this Graphics system.
CHAPTER IV

DEVELOPMENT TOOLS

Overview

The purpose of development tools are to decrease the amount of time usually needed to create a CAI package. Some of the tools to simplify and speed development are:

(1) Packaged subroutines - these routines are used throughout the system.

(2) Editing programs - these programs enable the designer to create text screens or graphics screens. These are programs that let the designer use the screen to create a frame which the program then uses to create a Pascal procedure. This will be further discussed in the next section.

(3) Proper documentation so that frames and data files can be constructed quickly with a minimum of errors and effort.

(4) Creating lessons so that they can be copied to a new lesson with a minimum of conversion.
Text Creating Program

The purpose of this program is to enable the designer to create text frames quickly and error free. This program is separate from the Graphics CAI package and is executed from the Command level of the computer system.

The designer can write a frame of text material. Then by executing this program, the text can be directly input onto a blank screen using a series of commands. After starting the program, the designer is asked what the name of the frame will be. This can consist up to an eight character name since the Apple system only recognizes the first eight characters of a file name even though it does accept up to 15 characters for a name.

The commands that position the cursor on the screen are:

(1) U - Move the cursor up a line.

(2) D - Move the cursor down a line.

(3) B - Move the cursor back one space.

(4) F - Move the cursor forward one space.

(5) L - Left justify the line of text.

(6) R - Right justify the line of text.
(7) C - Center justify the line of text.

(8) A - Add one space to the line of text.

(9) S - Subtract one space from the line of text.

(10) MU - Move the line of text one line up.

(11) MD - Move the line of text one line down.

(12) EE - Erase the line of text, the first E shifts the line of text to the right half of the screen and the second E actually erases the text.

(13) I - Start input of text.

(14) Carriage Return - used after the line of text was input.

(15) H - Help screen with the commands and explanations.

(16) Q - To quit the program.

After finishing, the program will then ask for a confirmation of your action. When it receives this confirmation, a 'Y', then the program creates an error free Pascal procedure which is then output to the diskette. This procedure can then be copied directly
into the program or library that will use it. This can save a great deal of time because the procedure that was created will display the text onto the screen as the designer wanted it. The debugging time for the procedure is saved as well as the time involved in making the text "pretty enough" for the student. A program that has graphics capabilities is somewhat more difficult and was not designed due to time constraints.

Packaged Routines

These are routines that are developed and used throughout the system. For example: input and output routines for data files, message routines, help routines, conversion routines from string input to integer or real numbers and response routines that decipher the input from the terminal. All of these routines can be designed and programmed once and used throughout the package as needed.
CHAPTER V

SYSTEM EXECUTION

Overview

The execution and pace of the Graphics CAI package is under the control of the user/student. That person is able to decide the amount of material to cover and the speed at which to learn it.

As stated in previous chapters, the student directly interacts with the initiating main program and the frames in the Libraries. Each lesson contains ten to twenty frames which can be used by the student. Each frame is made up of one screen of text or graphics. The student executes the next frame by a valid response to the frame that is presently on the screen. The valid responses to a frame that is not a test frame are:

(1) Carriage Return to continue to the next frame.

(2) "B)ack" to back up a frame.

(3) "D)epth" to have further discussion on the concept being presented.

(4) "H)elp" to obtain some assistance on the execution of the package.
(5) "Q)uit" to stop the execution, update the data files, return to the main initiating program and present the Menu screen to the student.

The commands, Back, Depth, Help and Quit can be entered as a response using the first letter, second, third, etc or all the letters that make up the command. This provides a flexibility to the package and lets the user/student type in the minimum amount of response for recognition by the package (response routine). All commands, requests and other input are always ended with a carriage return. This is in keeping with the rule number (1e) in Chapter II, the Seven Golden Rules For Good Software. Further descriptions for the student will be given in Appendix A, the Users Manual.
CHAPTER VI

Conclusions

CAI can be a valuable tool in the learning process if it has been developed correctly. As has been stated, the design and development process is long and arduous. It is an iterative process where during and after development, changes to the package are constantly occurring. An important method used during these phases is to have "test subjects" who have little knowledge of the material to actually execute the package. These "test subjects" usually come up with a number of suggestions for the improvement of the system. Then after implementation has been completed, again find several new "test subjects" that have no knowledge of the material and let them use your package. Keep good notes on their comments, frustrations and questions. Let them review the Users Manual and you may quickly find out that your piece de resistance needs improvement, more examples and further explanations.

The cost of developing CAI is very high and is somewhat lowered by using an authoring language. This decreases flexibility somewhat but speeds development. I have tried to provide an explanation and description on
the processes involved in the proper development of CAI courseware. The advantages and disadvantages of different methods have been described. Each of these involve a number of conflicting requirements. So to summarize CAI by using Doug Green's 11 Commandments For Educational Software.

(1) Thou shalt fit the software to the task.

(2) The least among you shall inherit the software.

(3) The software shall be able to take a joke.

(4) Thou shalt not be baffled by the microcomputer's documentation.

(5) Thou shalt exploit all of thy microcomputer's capabilities.

(6) Clutter not, lest ye be confused.

(7) Thou shalt purchase only the better mousetrap.

(8) Suffer the little children to come unto thy microcomputer.

(9) Thou cannot tell thy students without a scorecard.

(10) Thou shalt be entitled to receive proper support.

(11) He who has the gold, rules.
REFERENCES


**TABLE OF CONTENTS**

**CHAPTER**

I. Required System Configuration .......... 41
II. Diskette Backup ......................... 42
III. System Startup and Use ................. 43
IV. Material Covered ....................... 44
    Outline ................................... 44
    Input and Output ....................... 45
    Progress ............................... 46
    Descriptions ........................... 46
V. Reference Materials ..................... 48
    Textbooks ................................ 48
    Magazines ............................. 48
    Journals ................................ 48
VI. Terminology and Definitions ........... 49
CHAPTER I

REQUIRED SYSTEM CONFIGURATION

The hardware that is required to run the Computer Assisted Instruction (CAI) Graphics package is a:

- Apple II plus system with 48K of memory
- Two disk drives
- Language card for execution of UCSD/Apple Pascal programs

A color monitor or tv should be used because all of the graphics are done in color.
CHAPTER II

DISKETTE BACKUP

It is highly recommended that you make a copy of every diskette in the package. The reason for this is that a diskette can become damaged by accident or misuse which then makes the diskette absolutely useless. The procedure for making a copy of the diskette is to use the F(iler) command utility and then the T(ransfer) command within that utility.

Sequence of Commands:

F.............for the F(iler)

T.............for the T(ransfer)

Insert formatted blank disk into drive 1

Insert diskette to be copied into drive 2

5:, 4: transfer contents of diskette in drive 2 to diskette in drive 1

Insert your Applel: diskette back into drive 1

Q............quit the F(iler) utility
CHAPTER III

SYSTEM STARTUP AND USE

This package is set up as a turnkey system. This means that when the Apple is turned on, the Graphics package is automatically started. This also occurs when the RESET key is used. It is NOT recommended to use the RESET key while the package is executing. The reason for this is the fact that the master data file with your progress is NOT updated when the RESET key is pushed. The master data file is updated with new information when going from one lesson to another and when leaving the package by using the Q(uit) command. There is a total of 8 diskettes with this package labeled 0 to 7. Diskette 1 always goes into drive 1 before the Apple is turned on. Diskettes 0 and 2 through 8 always go into drive 2. After typing in your name and menu selection 2, 3 or 4, you will be told which diskette to put into drive 2.
CHAPTER IV

MATERIAL COVERED

Outline

The material that is covered in this CAI package is the following:

I. Fundamentals of Graphics
   (A) Description, Concepts and Uses
   (B) Definitions and Quizes
   (C) Explorations with graphic commands

II. Algorithms used in Graphics
    (A) Straight Lines
    (B) Curves and Circles

III. 2-D Graphics and Transformations
     (A) Methods and Descriptions
     (B) Mathematical Development
     (C) Quizes and Definitions

IV. Windowing and Clipping Algorithms

V. Polygon Filling Algorithms

VI. 3-D Graphics and Transformations

VII. Optional Review Material
     (A) UCSD Pascal
     (B) Trigonometry
Input and Output

The input and output of this package is consistent and straightforward. The input must always be typed in and a carriage return must be used to enter the input. The commands that can be entered are:

- Carriage Return........... to continue
- B(ack)................... to back up a frame
- D(epth)............... to get an in depth discussion of the material
- H(elp)............... to get these commands
- Q(uit)............... to get out of the package

The output of this package is the material to be studied and your progress through the material. The progress will be further described in the next section.
Progress

The progress of a student is measured by the accuracy and learning that takes place. The section of the package that you are currently in (module, lesson, frame) is also displayed along with the quiz statistics you have accumulated.

Descriptions

The description of each section

I. Fundamentals - The basic concepts of computer graphics is presented along with definitions, examples and quizzes.

II. Algorithms - Some of the basic algorithms for drawing straight lines, curves and circles are described.

III. 2-D Graphics - The concepts for 2-D graphics and their manipulations.

IV. Windowing and Clipping - Showing only portions of the displayed scene.

V. Polygon Filling - Algorithms that are used to fill the inside of an enclosed figure.
VI. 3-D Graphics - Concepts and ways of creating 3-D graphics are discussed.

VII. Optional Review - UCSD Pascal, sine, cosine and matrix algebra are reviewed.
CHAPTER V

REFERENCE MATERIALS

Textbooks:
Computer Graphics Primer
Mitchell Waite
Howard Sams Co.

Fundamentals of Interactive Graphics
James Foley, Andries Van Dam
Addison-Wesley

Principals Of Interactive Computer Graphics
William Newman, Robert Sproull
McGraw Hill Co.

Magazines:
Byte
Creative Computing
Graphics World
Personal Computing

Journals:
Communications Of The ACM
IBM Journal
SIGGRAPH Journal
CHAPTER VI

TERMINOLOGY AND DEFINITIONS

Algorithm......A defined procedure or recipe for solving a task or problem.

Backup........The procedure for copying files from one diskette to another.

CAI............Computer Assisted Instruction. Using the computer to study some area of interest.

Diskette.......A small device shaped like a record which is used to store information.

Flowchart.....The process of design that involves diagramming a procedure before coding it.

Frame........A single unit of material for the user/student to learn.

Graphics......A display of information in picture form.

Input.........The information that the user/student types into the computer.

Lesson.......A grouping of materials that are composed of several frames.

Turnkey.......The method of starting a program up when
the power is turned on.

UCSD Pascal...The University of California at San Diego implementation of Pascal on which Apple Pascal is based.

2-D Graphics..Two dimensional graphics where the scene is flat.

3-D Graphics..Three dimensional graphics where the scene simulates three dimensions by projecting the 3-D scene onto a 2-D surface.
PROGRAM TEXTCREATE;

(*AUTHOR ROBERT DUTILLY
DESCRIPTION THE PURPOSE OF THIS PROGRAM IS TO CREATE
PASCAL PROCEDURES FROM TEXT INPUT FROM
THE SCREEN.*)

USES APPLESTUFF;

VAR
  TXT : FILE OF CHAR;
  INPUT : ARRAY[0..24] OF STRING[80];
  TOTALS : ARRAY[1..24,1..2] OF INTEGER;
  CH : CHAR;
  I,J,LIN,E,OVER : INTEGER;
  CLEAR,NAME,FILENAME : STRING;

PROCEDURE INITIALIZE;

(* INITIALIZES THE VARIABLES AND ARRAYS*)

BEGIN
  FOR I := 0 TO 24 DO
    BEGIN
      INPUT[I] := ' ' ; (*ACTUAL TEXT*)
    END;
  FOR I := 1 TO 24 DO
    BEGIN
      TOTALS[I,1] := 0; (*LENGTH OF TEXT*)
      TOTALS[I,2] := 0; (*BLANK SPACES*)
    END;
  CLEAR := ' ' ;
  FOR I := 1 TO 79 DO
    BEGIN
      INSERT(' ',CLEAR,1);
    END;
  NAME := ' ' ;
  I := 0; J := 0;
  LINE := 0; OVER := 0;
END;

PROCEDURE DESCRIBE;

(* DESCRIBES THE PROGRAM TO THE USER AND REQUESTS A FILE
NAME FROM THE USER TO BE USED FOR THE OUTPUT OF THE
PROCEDURE IN A TEXT FILE*)

BEGIN
  WRITE(CHR(12)); WRITELN; WRITELN;
WRITELN('THIS IS THE FRAME CREATION PROGRAM.');
WRITELN;
WRITELN('YOU WILL BE ABLE TO INPUT TEXT ONTO');
WRITELN;
WRITELN('A BLANK SCREEN USING A SERIES OF');
WRITELN;
WRITELN('COMMANDS. THESE COMMANDS ARE GIVEN');
WRITELN;
WRITELN('ON THE NEXT SCREEN.');
WRITELN; WRITELN;
WRITE( 'PLEASE INPUT FILE NAME? ' ) ;
READLN(NAME);
END;

PROCEDURE INSTRUCTIONS;

(* PROVIDES THE INSTRUCTIONS TO THE USER FOR THE PROPER
USE OF THIS PROGRAM*)

BEGIN
WRITELN('
THE COMMANDS ARE:');
WRITELN;
WRITELN( '<U MOVE CURSOR UP'>);
WRITELN( '<D MOVE CURSOR DOWN'>);
WRITELN( '<B MOVE CURSOR BACK'>);
WRITELN( '<F MOVE CURSOR FORWARD'>);
WRITELN( '<L LEFT JUSTIFY THE LINE'>);
WRITELN( '<R RIGHT JUSTIFY THE LINE'>);
WRITELN( '<C CENTER JUSTIFY THE LINE'>);
WRITELN( '<A ADD A SPACE TO THE LINE'>);
WRITELN( '<S SUBTRACT A SPACE FROM LINE'>);
WRITELN( '<MU MOVE LINE UP'>);
WRITELN( '<MD MOVE LINE DOWN'>);
WRITELN( '<EE ERASE LINE'>);
WRITELN( '<I START INPUT OF TEXT'>);
WRITELN( '<RETURN USED AFTER INPUT OF A LINE'>);
WRITELN( '<H HELP'>);
WRITELN( '<Q QUIT'>);
WRITELN;
WRITE('
PRESS RETURN ');
READLN;
WRITE(CHR(12));
IF (CH = 'H') THEN
BEGIN
WRITELN;
FOR I := 0 TO 23 DO
BEGIN
WRITELN(INPUT[I]);
END;
END;
END;
PROCEDURE READIN;

(* READS EACH LINE OF TEXT THAT WAS INPUT TO THE SCREEN*)

BEGIN
  INPUT[LINE] := ";
  READLN(INPUT[LINE]);

  FOR I := 1 TO OVER DO
    BEGIN
      INSERT(" ", INPUT[LINE], 1); (*INSERTS BLANK SPACES*)
    END;

  TOTALS[LINE,1] := LENGTH(INPUT[LINE]);
  TOTALS[LINE,2] := OVER;

  LINE := LINE + 1;
  GOTOXY(0,LINE);
  OVER := 0;
END;

PROCEDURE LEFT;

(* LEFT JUSTIFIES THE LINE OF TEXT AND UPDATES THE DATA IN THE DATA ARRAYS*)

BEGIN
  REPEAT
    TOTALS[LINE,2] := TOTALS[LINE,2] - 1;
    DELETE(INPUT[LINE],1,1);
  UNTIL (TOTALS[LINE,2] <= 0);
  GOTOXY(0,LINE);
  WRITELN(CLEAR);
  GOTOXY(0,LINE);
  WRITELN(INPUT[LINE]);
END;

PROCEDURE RIGHT;

(* RIGHT JUSTIFIES THE LINE OF TEXT AND UPDATES THE DATA IN THE DATA ARRAYS*)

BEGIN
  J := 39 - LENGTH(INPUT[LINE]);

  FOR I := 1 TO J DO
BEGIN
  INSERT( '  ', INPUT[LINDEX], 1 );
END;

GOTOXY(0, LINDEX);
WRITELN( CLEAR );
GOTOXY(0, LINDEX);
WRITELN( INPUT[LINDEX] );
END;

PROCEDURE CENTER;

(* CENTER JUSTIFIES THE LINE OF TEXT AND UPDATES THE DATA IN THE DATA ARRAYS*)

BEGIN
  J := 19 - (TOTALS[LINDEX, 2] + (TOTALS[LINDEX, 1] DIV 2));
  REPEAT
    IF (J < 0) THEN
      BEGIN
        DELETE( INPUT[LINDEX], 1, 1 );
        J := J + 1;
      END;
    IF (J > 0) THEN
      BEGIN
        INSERT( '  ', INPUT[LINDEX], 1 );
        J := J - 1;
      END;
    UNTIL (J = 0);
  GOTOXY(0, LINDEX);
  WRITE( CLEAR );
  GOTOXY(0, LINDEX);
  WRITELN( INPUT[LINDEX] );
END;

PROCEDURE ADDSPACE;

(* ADD A BLANK SPACE TO THE LINE OF TEXT AND MOVE THE LINE OVER TO THE RIGHT*)

BEGIN
  IF (LENGTH(INPUT[LINDEX]) < 39) THEN
    BEGIN
      INSERT( '  ', INPUT[LINDEX], 1 );
      TOTALS[LINDEX, 1] := TOTALS[LINDEX, 1] + 1;
    END;
TOTALS[LINE,2] := TOTALS[LINE,2] + 1;
GOTOXY(0,LINE);
WRITE(CLEAR);
GOTOXY(0,LINE);
WRITELN(INPUT[LINE]);
END;

PROCEDURE SUBSPACE;

(* SUBTRACT A BLANK SPACE TO THE LINE OF TEXT AND MOVE THE LINE OVER TO THE LEFT*)

BEGIN
IF (TOTALS[LINE,2] > 0) THEN
BEGIN
DELEITE(INPUT[LINE],1,1);
TOTALS[LINE,1] := TOTALS[LINE,1] - 1;
TOTALS[LINE,2] := TOTALS[LINE,2] - 1;
GOTOXY(0,LINE);
WRITE(CLEAR);
GOTOXY(0,LINE);
WRITELN(INPUT[LINE]);
END;
END;

PROCEDURE MOVELINE;

(* MOVE THE LINE OF TEXT EITHER UP OR DOWN ON THE SCREEN BY ERASING THE OLD LINE AND REWRITING IT ON THE NEW LINE.*)

BEGIN
IF (TOTALS[LINE,1] > 0) THEN
BEGIN
READ(KEYBOARD,CH);

(* MOVE THE LINE UP ONE*)

IF (CH = "U") AND (LINE > 0) THEN
BEGIN
INPUT[LINE-1] := INPUT[LINE];
TOTALS[LINE-1,1] := TOTALS[LINE,1];
TOTALS[LINE-1,2] := TOTALS[LINE,2];
INPUT[LINE] := " ";
TOTALS[LINE,1] := 0;
TOTALS[LINE,2] := 0;
GOTOXY(0,LINE);
WRITELN(CLEAR);
LINE := LINE - 1;
END;
GOTOXY(0,LINE);
WRITE(INPUT[LINE]);
GOTOXY(0,LINE);
END;

(* MOVE THE LINE DOWN ONE*)

IF (CH = 'D') AND (LINE < 23) THEN
BEGIN
   INPUT[LINE+1] := INPUT[LINE];
   TOTALS[LINE+1,1] := TOTALS[LINE,1];
   TOTALS[LINE+1,2] := TOTALS[LINE,2];
   INPUT[LINE] := '"';
   TOTALS[LINE,1] := 0;
   TOTALS[LINE,2] := 0;
   GOTOXY(0,LINE);
   WRITELN(CLEAR);
   LINE := LINE + 1;
   GOTOXY(0,LINE);
   WRITE(INPUT[LINE]);
   GOTOXY(0,LINE);
END;
END;

PROCEDURE ERASE;

(* THIS ENABLES THE USER TO ERASE A LINE OF TEXT BY ENTERING TWO
E'S. THE FIRST E SHIFTS THE TEXT LINE TO THE RIGHT HALF OF
THE SCREEN AND THE SECOND E ACTUALLY ERASES THE LINE. IF AN
ERROR IS MADE, THEN BY TYPING ANOTHER COMMAND OTHER THAN AN E
ENABLES THE USER TO SAVE THAT LINE OF TEXT.*)

BEGIN
IF (TOTALS[LINE,1] > 0) THEN
BEGIN
   J := 40 - TOTALS[LINE,2];
   (* SHIFTS TO THE RIGHT*)
   FOR I := 1 TO J DO
   BEGIN
      INSERT(' ',INPUT[LINE],1);
   END;
   WRITE(INPUT[LINE]);
   GOTOXY(0,LINE);
   READ(KEYBOARD,CH);
(* IF SECOND E THE ERASE, OTHERWISE RESTORE TEXT LINE*)

IF (CH = 'E')
THEN
BEGIN
    INPUT[LINEL] := '"';
    TOTALS[LINEL,1] := 0;
    TOTALS[LINEL,2] := 0;
    WRITELN(CLEAR);
END
ELSE
BEGIN
    FOR I := 1 TO J DO
    BEGIN
        DELETE(INPUT[LINEL],1,1);
    END;
    WRITE(INPUT[LINEL]);
    GOTOXY(OVER,LINEL);
END;
END;

PROCEDURE COMMAND;

(* THIS IS THE COMMAND ROUTINE WHICH READS THE COMMANDS, DECIDES THE ACTION TO TAKE AND ROUTES THE ACTION TO THE PROPER PROCEDURE.*)

BEGIN
    WRITE(CHR(12));
    LINE := 0;
    OVER := 0;
    REPEAT
        GOTOXY(OVER,LINEL);
    REPEAT
        READ(KEYBOARD,CH);
    (* THIS PRINTS OUT THE COMMAND THAT WAS JUST ENTERED*)
    GOTOXY(32,0);
    CASE CH OF
        'U' : WRITE(' UP');
        'D' : WRITE(' DOWN');
        'B' : WRITE(' BACK');
        'F' : WRITE(' FORWARD');
"L" : WRITE(' LEFT');
"R" : WRITE(' RIGHT');
"C" : WRITE(' CENTER');
"A" : WRITE(' ADD');
"S" : WRITE(' SUBTRACT');
"M" : WRITE(' MOVE');
"E" : WRITE(' ERASE');
"I" : WRITE(' INPUT');

END;

GOTOXY(OVER,LINE);

IF (CH = 'U') AND (LINE > 0) THEN LINE := LINE - 1;
IF (CH = 'D') AND (LINE < 23) THEN LINE := LINE + 1;
IF (CH = 'B') AND (OVER > 0) THEN OVER := OVER - 1;
IF (CH = 'F') AND (OVER < 39) THEN OVER := OVER + 1;
IF (CH = 'H') THEN INSTRUCTIONS;
IF (CH = 'L') THEN LEFT;
IF (CH = 'R') THEN RIGHT;
IF (CH = 'C') THEN CENTER;
IF (CH = 'A') THEN ADDSPACE;
IF (CH = 'S') THEN SUBSPACE;
IF (CH = 'M') THEN MOVELINE;
IF (CH = 'E') THEN ERASE;

GOTOXY(OVER,LINE);

IF (CH = 'I') THEN READIN;

UNTIL (CH = 'Q');

(* SECTION TO LEAVE THE COMMAND ROUTINE, IF YES, THEN
 LEAVE ELSE REWRITE THE SCREEN AND RETURN TO COMMAND *)

WRITE(CHR(12));
WRITELN; WRITELN;
WRITE('CONFIRM TO FINISH THE FRAME, Y OR N? .');
READLN(CH);

IF (CH = 'N')
THEN
  BEGIN
    WRITELN;
    FOR I := 0 TO 23 DO
      BEGIN
        WRITELN(INPUT[I]);
      END;
  END
ELSE
  CH := 'Q';

UNTIL (CH = 'Q');
PROCEDURE OUTPUT;

(* THIS IS THE FINAL PROCEDURE THAT ACTUALLY CREATES A PASCAL
PROCEDURE FROM THE TEXT LINES THAT WERE INPUT AND THEN
CREATES AN OUTPUT TEXT FILE.*)

BEGIN
WRITE(CHR(12)); WRITELN;
WRITELN('OUTPUTTING TEXT TO FILE');
FILENAME := CONCAT(NAME,'.TEXT');
REWRITE(TXT,FILENAME);
INPUT[0] := 'WRITE(CH(12));';

I := 24;
J := 0;
REPEAT
  IF (TOTALS[I,1] = 0) THEN J := J + 1;
  I := I - 1;
UNTIL (TOTALS[I,1] > 0);
FOR I := 1 TO 24-J DO
BEGIN
  IF (TOTALS[I,1] > 0)
  THEN
  BEGIN
    BEGIN
      IF (I < 24-J)
      THEN INPUT[I] := CONCAT('WRITELN(','\n',INPUT[I])
      ELSE INPUT[I] := CONCAT('WRITE(','\n',INPUT[I]);
    INPUT[I] := CONCAT(INPUT[I],',\''\'');
  END;
  ELSE
  BEGIN
    INPUT[I] := 'WRITELN;';
  END;
END;

(* OUTPUT INFORMATION TO THE TEXT FILE*)

NAME := CONCAT('PROCEDURE ',NAME);
NAME := CONCAT(NAME,','';\''');
WRITELN(TXT,NAME);
WRITELN(TXT);
WRITELN(TXT,'BEGIN');

(* OUTPUT THE LINES OF TEXT TO THE FILE*)

FOR I := 0 TO 24-J DO
BEGIN
```
WRITELN(TXT," \[\text{INPUT[I]}\]);
END;

WRITELN(TXT," \text{READLN(R);"});
WRITELN(TXT," \text{END;"});
CLOSE(TXT,LOOK);
WRITE(CHR(12));
END;

(* MAIN LINE *)
BEGIN
  INITIALIZE;
  DESCRIBE;
  INSTRUCTIONS;
  COMMAND;
  OUTPUT;
END.
```
Physical Setup

35 tracks per diskette

16 sectors per track

256 bytes per sector

8 bits per byte

2 sectors per block or 512 bytes

8 blocks per track

280 blocks or 140,000 bytes per diskette

Textfiles can be as large as 38 blocks

Files must be contiguous

77 files possible on a diskette

Editor Constraints

18,400 bytes maximum per text file

-1,024 bytes for the header

17,376 bytes available for actual text

Compiler Constraints

1200 bytes maximum for a procedure
Diskette block layout

0 to 1 bootstrap
2 to 5 directory
6 to 280 files
APPENDIX D

PROGRAM EXAMPLES
(*$S++*$)
PROGRAM THESIS;

(*PROGRAM : THESIS.PL
AUTHOR : BOB DUTILLY
DESCRIPTION : THIS IS THE MAINLINE PROGRAM FOR THE CAI
GRAPHICS PROGRAM FOR THE MASTERS THESIS.
*)

USES CHAINSTUFF;

VAR
MASTER : FILE OF CHAR; (*MASTER FILE*)
DATA : FILE OF RECORD
  M,L,F : INTEGER; (*LOCATION DATA FILE*)
END;
RESULTS : FILE OF RECORD
  H : ARRAY [1..8] OF INTEGER; (*QUIZ DATA FILE*)
END;
NAME,INPUT,FIRSTNAME,LASTNAME : STRING[20];
CHOICE,I,J,N,MODU,LESS,FRM,MODULE,LESSON,FRAME,
CODE,RECNUM : INTEGER;
RESPONSE,MESSAGE,INP : STRING[80];
STR1,STR2,STR3,STR4,SNUM : STRING[2];
BLANK,REV : STRING[1];
NEWS : BOOLEAN;

(* PROCEDURES *)

PROCEDURE TRIM(VAR INP:STRING);
(* THIS PROCEDURE REMOVES LEADING AND TRAILING BLANKS FROM
THE RESPONSE INPUT BY THE USER*)
VAR T : INTEGER;
BEGIN
(*REMOVE LEADING BLANKS*)
  REPEAT
    T := 0;
    T := POS(' ',INP);
    IF (T = 1) THEN DELETE(INP,1,1);
  UNTIL (T > 1) OR (T = 0) OR (LENGTH(INP) = 0);
(*GET RID OF TRAILING BLANKS*)
  FOR T := LENGTH(INP) DOWNTO 1 DO
    IF POS(' ',INP,T) = T THEN
      DELETE(INP,T,1);
    END IF;
END IF;
END;

(*$S++*$)
BEGIN
  BLANK := COPY(INP,T,1);
  IF (BLANK = ' ') THEN DELETE(INP,T,1)
  ELSE T := 0;
END;
BLANK := ' ';

;/*CHECK IF USER WANTS OUT*/
IF (RESPONSE = 'Q.') OR (RESPONSE = 'QU.'). OR
  (RESPONSE = 'QUI'). OR (RESPONSE = 'QUIT') THEN
BEGIN
  WRITECHR(12));
  MESSAGE := ' ';
  SETCV(LMESSAGE);
  SETCHAIN('74:THESIS');
  EXIT(THESIS);
END;

PROCEDURE QUERY;

/* THIS PROCEDURE GETS THE USERS NAME*/
BEGIN
  WRITECHR(12));
  WRITELN; WRITELN; WRITELN;
  WRITELN(' THIS IS THE COMPUTER ASSISTED ');;
  WRITELN;
  WRITELN(' INSTRUCTION PACKAGE FOR COMPUTER');;
  WRITELN;
  WRITELN(' GRAPHICS. ');;
  WRITELN;
  WRITELN;
  REPEAT
    GOTOXY(0,13);
    WRITELN('PLEASE TYPE IN YOUR FIRST AND LAST NAME ');;
    WRITELN;
    WRITE(' AND PRESS RETURN ');;
    READLN(INP);
  UNTIL (LENGTH(INP) > 0);

/*CHECK IF USERS QUITS*/
IF (INPUT = 'Q.') OR (INPUT = 'QU.') OR
  (INPUT = 'QUI') OR (INPUT = 'QUIT') THEN
BEGIN
  WRITECHR(12));
  MESSAGE := ' ';
  SETCV(LMESSAGE);
  SETCHAIN('74:THESIS');
  EXIT(THESIS);
REPEAT
  I := 0;
  I := POS(´ `,INPUT);
  IF (I = 1) THEN DELETE(INPUT,1,1);
UNTIL (I > 1) OR (LENGTH(INPUT) = 0);

IF (LENGTH(INPUT) = 0) THEN QUERY;

I := POS(´ `,INPUT);
FIRSTNAME := COPY(INPUT,1,I-1);

(*MAKE INPUT TO 20 CHARACTERS*)
FOR I := 20 DOWNTO LENGTH(INPUT)+1 DO
  INPUT := CONCAT(INPUT,BLANK);
END;

FOR I := 20 DOWNTO LENGTH(FIRSTNAME)+1 DO
  FIRSTNAME := CONCAT(FIRSTNAME,BLANK);
END;

END;

PROCEDURE READFILE;
(* THIS PROCEDURE READS THE MASTER AND DATA FILE*)
BEGIN
  CODE := 0; NEWS := FALSE;
  RESET(MASTER,'#4:MASTER.TEXT'); (*OPEN FILE*)
  RESET(DATA,'#4:DATA.DATA');
  IF (IRESULT <> 0) THEN
    EXIT(READFILE);
  END;
  N := 1;
  WHILE NOT EOF(MASTER) AND (CODE = 0) DO
    BEGIN
      READ(MASTER,RECNUM);
      READLN(MASTER,NAME);
      SEEK(DATA,RECNUM);
      GET(DATA);
      (*IF NAME MATCHES THEN USE IT*)
      IF (INPUT = NAME) THEN
BEGIN
  CODE := 1;
  MODULE := DATA^M;
  LESSON := DATA^L;
  FRAME := DATA^F;
  STR(MODULE,STR1);
  STR(LESSON,STR2);
  STR(FRAME,STR3);
  STR(RECNUM,STR4);
END
ELSE
  N := N + 1;
END;

(*THIS IS A NEW USER, SETUP RECORDS*)

IF (CODE = 0) THEN
BEGIN
  RESET(MASTER);
  RESET(DATA);
  RESET(RESULTS, 'RESULT.DATA');
  NEWS := TRUE;
  FOR I := 1 TO N - 1 DO
     BEGIN
       READLN(MASTER);
     END;
  NAME := INPUT;
  DATA^M := 1;
  DATA^L := 1;
  DATA^F := 1;
  MODULE := 1;
  LESSON := 1;
  FRAME := 1;
  RECNUM := N - 1;

  (*OUTPUT THE INFO TO THE FILES*)
  WRITELN(MASTER,RECNUM:3,NOME:20);
  SEEK(DATA,RECNUM);
  PUT(DATA);
  FOR I := 1 TO 8 DO RESULTS^H[I] := -1;
  SEEK(RESULTS,RECNUM);
  PUT(RESULTS);

  (*SETUP STRINGS FOR PASSING DATA TO MODULE*)
  STR(MODULE,STR1);
  STR(LESSON,STR2);
PROCEDURE NEW;

(*THIS PROCEDURE CREATES NEW FILES WHEN THIS PROGRAM IS FIRST USED*)

BEGIN

(*$I-**)

REWRIGHT (MASTER, "#4:MASTER.TEXT"); (*INSERT NEW NAME AND DATA*)

NEWS := TRUE;
NAME := INPUT;
DATA^M := 1;
DATA^L := 1;
DATA^F := 1;
MODULE := 1;
LESSON := 1;
FRAME := 1;
RECNUM := 0;
WRITE (MASTER, RECNUM:3, NAME:20);
CLOSE (MASTER, LOCK);
REWRIGHT (DATA, "#4:DATA.DATA");
SEEK (DATA, RECNUM);
PUT (DATA);
CLOSE (DATA, LOCK);
REWRIGHT (RESULTS, "#4:RESULTS.DATA");
FOR I := 1 TO 8 DO RESULTS^H[I] := -1;
SEEK (RESULTS, RECNUM);
PUT (RESULTS);
CLOSE (RESULTS, LOCK);
(*$I-*)

(*SETUP STRINGS FOR PASSING DATA*)

STR(MODULE, STR1);
STR(LESSON, STR2);
STR(FRAME, STR3);
STR(RECNUM, STR4);
END;
PROCEDURE DUMPONE;

(*THIS PROCEEDURE DUMPS ONE USERS RECORD TO THE SCREEN*)

BEGIN
  (*$!-*)
  WRITE(CHR(12));
  RESET(MASTER,'#4:MASTER.TEXT');
  RESET(DATA,'#4:DATA.DATA');
  RESET(RESULTS,'#4:RESULTS.DATA');

  IF (IORESULT 0) THEN
  BEGIN
    CLOSE(MASTER,LOCK);
    CLOSE(DATA,LOCK);
    CLOSE(RESULTS,LOCK);
    DUMPONE;
  END;
  (*$!+*)

  (*SEARCH THROUGH THE MASTER AND DATA FILES*)

  WHILE NOT EOF(MASTER) DO
  BEGIN
    READ(MASTER,RECNUM,NAMEME);
    IF (NAME ' INPUT) THEN
    BEGIN

      (*OUTPUT THE INFORMATION TO THE USER*)

      SEEK(DATA,RECNUM);
      GET(DATA);
      SEEK(RESULTS,RECNUM);
      GET(RESULTS);
      GOTOXY(0,4);
      WRITELN(' NAME MODULE LESSON FRAME');
      WRITELN;
      WRITE(NAME);
      WRITE(DATA.M,');
      WRITE(DATA.L,');
      WRITELN(DATA.F);
      WRITELN;
(* OUTPUT THE QUIZ RESULTS *)

WHILE (RESULTS^H[J] <> -1) DO
  BEGIN
    WRITELN;
    CASE J OF
      1 : WRITE('QUIZ 1 :');
      2 : WRITE('QUIZ 2 :');
      3 : WRITE('QUIZ 3 :');
      4 : WRITE('QUIZ 4 :');
      5 : WRITE('QUIZ 5 :');
      6 : WRITE('QUIZ 6 :');
      7 : WRITE('QUIZ 7 :');
      8 : WRITE('QUIZ 8 :');
    END;
    WRITELN(RESULTS^H[J]);
    J := J + 1;
  END;
  WRITELN;
  IF (J = 1) THEN WRITELN('SORRY, NO RESULTS TO REPORT YET');
  CLOSE(MASTER, LOCK);
  CLOSE(DATA, LOCK);
  CLOSE(RESULTS, LOCK);
  GOTOXY(8,22);
  WRITE('PRESS RETURN TO CONTINUE');
  READLN(RESPONSE);
  EXIT(DUMPONE);
END;
END;

PROCEDURE MENU;

(* THIS THE THE PROCEDURE THAT LETS THE USER-DECIDE WHAT
ACTION TO PERFORM WHILE USING THIS PACKAGE.*)

BEGIN
  REPEAT
    WRITE(CHR(12));
    WRITELN;
    WRITELN;
  END;
END;
WRITELN('  MENU  ');
GOTOXY(0,6);
WRITELN('  (1) HOW TO USE THIS SYSTEM. ');
WRITELN;
WRITELN('  (2) START WITH THE LESSONS. ');
WRITELN;
WRITELN('  (3) REVIEW PAST MATERIAL. ');
WRITELN;
WRITELN('  (4) OPTIONAL REVIEW MATERIAL. ');
WRITELN;
WRITELN('  (5) DISPLAY STUDENT DATA. ');
WRITELN;
WRITELN('  (6) GET OUT OF THE SYSTEM. ');
WRITELN;
WRITELN;
WRITELN('  PLEASE ENTER SELECTION (1 TO 6) ');
WRITELN;
WRITE('  AND PRESS RETURN ');
READLN(RESPONSE);
(*CHECK THE VARIOUS POSSIBLE RESPONSES, PERFORM ACTION*)

IF (LENGTH(RESPONSE) > 0) THEN TRIM(RESPONSE);
IF (RESPONSE = '1') THEN DESCRIPTION;
IF (RESPONSE = '3') AND (NOT NEWS) THEN REVIEW;
IF (RESPONSE = '3') AND (NEWS) THEN BEGIN
  WRITE(CHR(12));
  WRITELN; WRITELN; WRITELN;
  WRITELN('SORRY ', FIRSTNAME);
  WRITELN('YOU HAVE NOTHING TO REVIEW YET');
  WRITELN;
  WRITE(' PRESS RETURN ');
  READLN(RESPONSE);
  IF (LENGTH(RESPONSE) > 0) THEN TRIM(RESPONSE);
  RESPONSE := '';
END;
IF (RESPONSE = '5') THEN DUMPONE;
IF (RESPONSE = '5ALL') THEN DUMPALL;
IF (RESPONSE = '6') THEN BEGIN
  WRITE(CHR(12));
  MESSAGE := '';
  SETVAL(MESSAGE);
  SETCHAIN('$.4:THESIS');
  EXIT(THESIS);
END;
UNTIL (RESPONSE = '2') OR (RESPONSE = '4');
END;
PROCEDURE BREAK;
(* This procedure is used when returning from the module, the message is taken apart, then the strings themselves are converted into numeric values to indicate what has occurred at the lower level.*)

BEGIN

FIRSTNAME := COPY(MESSAGE,1,20);
STR1 := COPY(MESSAGE,21,2);
STR2 := COPY(MESSAGE,23,2);
STR3 := COPY(MESSAGE,25,2);
STR4 := COPY(MESSAGE,27,2);
INPUT := COPY(MESSAGE,29,20);
REV := COPY(MESSAGE,50,1);

(* Sections to convert string data to numeric data*)

MODULE := 0; CODE := 0;
REPEAT
    MODULE := MODULE + 1;
    STR(MODULE,SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR1)) THEN
        SNUM := CONCAT(SNUM,BLANK);
    IF (SNUM = STR1) THEN CODE := 1;
UNTIL (CODE = 1);

LESSON := 0; CODE := 0;
REPEAT
    LESSON := LESSON + 1;
    STR(LESSON,SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR2)) THEN
        SNUM := CONCAT(SNUM,BLANK);
    IF (SNUM = STR2) THEN CODE := 1;
UNTIL (CODE = 1);

FRAME := 0; CODE := 0;
REPEAT
    FRAME := FRAME + 1;
    STR(FRAME,SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR3)) THEN
        SNUM := CONCAT(SNUM,BLANK);
    IF (SNUM = STR3) THEN CODE := 1;
UNTIL (CODE = 1);

RECNUM := -1; CODE := 0;
REPEAT
    RECNUM := RECNUM + 1;
    STR(RECNUM,SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR4)) THEN
SNUM := CONCAT(SNUM,BLANK);
IF (SNUM = STR4) THEN CODE := 1;
UNTIL (CODE = 1);

(* REVIEW WAS PERFORMED BY THE USER*)

IF (REV = "R") THEN
BEGIN
   STR1 := COPY(MESSAGE,51,2);
   STR2 := COPY(MESSAGE,53,2);
   STR3 := COPY(MESSAGE,55,2);

   MODU := 0; CODE := 0;
   REPEAT
      MODU := MODU + 1;
      STR(MODU,SNUM);
      IF (LENGTH(SNUM) < LENGTH(STR1)) THEN
         SNUM := CONCAT(SNUM,BLANK);
      IF (SNUM = STR1) THEN CODE := 1;
      UNTIL (CODE = 1);

   LESS := 0; CODE := 0;
   REPEAT
      LESS := LESS + 1;
      STR(LESS,SNUM);
      IF (LENGTH(SNUM) < LENGTH(STR2)) THEN
         SNUM := CONCAT(SNUM,BLANK);
      IF (SNUM = STR2) THEN CODE := 1;
      UNTIL (CODE = 1);

   FRM := 0; CODE := 0;
   REPEAT
      FRM := FRM + 1;
      STR(FRM,SNUM);
      IF (LENGTH(SNUM) < LENGTH(STR3)) THEN
         SNUM := CONCAT(SNUM,BLANK);
      IF (SNUM = STR3) THEN CODE := 1;
      UNTIL (CODE = 1);
END;

PROCEDURE OUT;

(* THIS PROCEDURE SETS UP THE MESSAGE TO PASS TO THE MODULE
 AND TELLS THE USER WHAT DISKETTE TO INSERT.*)

BEGIN
   WRITE(CHR(12));
   WRITELN;

   (* PASS THE STRING VALUE WITH A LEADING BLANK OR WITH*
IF (LENGTH(STR1) = 1) THEN STR1 := CONCAT(STR1, BLANK);
IF (LENGTH(STR2) = 1) THEN STR2 := CONCAT(STR2, BLANK);
IF (LENGTH(STR3) = 1) THEN STR3 := CONCAT(STR3, BLANK);
IF (LENGTH(STR4) = 1) THEN STR4 := CONCAT(STR4, BLANK);

(*SETUP MESSAGE WITH VALUES*)

MESSAGE := CONCAT(FIRSTNAME, STR1);
MESSAGE := CONCAT(MESSAGE, STR2);
MESSAGE := CONCAT(MESSAGE, STR3);
MESSAGE := CONCAT(MESSAGE, STR4);
MESSAGE := CONCAT(MESSAGE, INPUT);
MESSAGE := CONCAT(MESSAGE, 'D');
IF (REV = 'R') THEN BEGIN
  STR(MODULE, STR1);
  STR(LESSON, STR2);
  STR(FRAME, STR3);
  IF (LENGTH(STR1) = 1) THEN STR1 := CONCAT(STR1, BLANK);
  IF (LENGTH(STR2) = 1) THEN STR2 := CONCAT(STR2, BLANK);
  IF (LENGTH(STR3) = 1) THEN STR3 := CONCAT(STR3, BLANK);
  MESSAGE := CONCAT(MESSAGE, 'R');
  MESSAGE := CONCAT(MESSAGE, STR1);
  MESSAGE := CONCAT(MESSAGE, STR2);
  MESSAGE := CONCAT(MESSAGE, STR3);
  MODULE := MODU;
END;

>(* THE COMMAND TO PASS THE MESSAGE*)

SETCVAL(MESSAGE);

>(* THE NEXT PROGRAM TO CHAIN TO*)

CASE MODULE OF
  0 : SETCHAIN("#5:MODULE0");
  1 : SETCHAIN("#5:MODULE1");
  2 : SETCHAIN("#5:MODULE2");
  3 : SETCHAIN("#5:MODULE3");
  4 : SETCHAIN("#5:MODULE4");
  5 : SETCHAIN("#5:MODULE5");
  6 : SETCHAIN("#6:MODULE6");
END;

WRITELN;

CASE MODULE OF
  0 : WRITELN("INSERT DISKETTE 0 INTO DRIVE #2");
1 : WRITELN("INSERT DISKETTE 2 INTO DRIVE #2");
2 : WRITELN("INSERT DISKETTE 3 INTO DRIVE #2");
3 : WRITELN("INSERT DISKETTE 4 INTO DRIVE #2");
4 : WRITELN("INSERT DISKETTE 5 INTO DRIVE #2");
5 : WRITELN("INSERT DISKETTE 6 INTO DRIVE #2");
6 : WRITELN("INSERT DISKETTE 7 INTO DRIVE #2");
END;

WRITELN;
WRIT("PRESS RETURN AFTER INSERTING DISKETTE ");
READLN(RESPONSE);
GOTOXY(7,21);
WRITE("PLEASE WAIT A FEW MOMENTS ");
END;

(* MAIN LINE *)
BEGIN
GETCVAL(MESSAGE);
BLANK := ' ';

(* IF LENGTH IS ZERO THEN STARTING OUT, OTHERWISE
   IT IS RETURNING FROM A MODULE*)
IF (LENGTH(MESSAGE) = 0) THEN BEGIN
   QUERY;
   READFILE;
   IF (IRESULT <> 0) THEN NEW;
   MENU;
   OUT;
END
ELSE BEGIN
   WRITE(CHR(12));
   NEWS := FALSE;
   BREAK;
   MENU;
   MESSAGE := ' ';
   IF (RESPONSE = '2') THEN OUT;
   SETCVAL(MESSAGE);
END.
END.
PROGRAM MODULE1;

(*AUTHOR BOB DUTILLY
DESCRIPTION THIS IS THE FIRST MODULE FOR THE
CAI GRAPHICS PACKAGE*)

USES CHAINSTUFF;

VAR

  MESSAGE : STRING[80];
  FIRSTNAME, NAME : STRING[20];
  STR1, STR2, STR3, STR4, STR6, STR7, SNUM : STRING[2];
  LESSON, CODE : INTEGER;
  BLANK, DIRECTION, REVIEW : STRING[1];
  RESPONSE : STRING;

PROCEDURE BREAK;

(*THIS PROCEDURE TAKES THE MESSAGE STRING APART AND
DETERMINES THE ACTION REQUIRED*)

BEGIN
  FIRSTNAME := COPY(MESSAGE, 1, 20);
  STR1 := COPY(MESSAGE, 21, 2);
  STR2 := COPY(MESSAGE, 23, 2);
  STR3 := COPY(MESSAGE, 25, 2);
  STR4 := COPY(MESSAGE, 27, 2);
  NAME := COPY(MESSAGE, 29, 20);
  DIRECTION := COPY(MESSAGE, 49, 1);
  REVIEW := COPY(MESSAGE, 50, 1);
  LESSON := 0; CODE := 0;

  (* DETERMINE WHICH LESSON IS NEEDED*)
  REPEAT
    LESSON := LESSON + 1;
    STR(LESSON, SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR2)) THEN
      SNUM := CONCAT(SNUM, BLANK);
    IF (STR2 = SNUM) THEN CODE := 1;
  UNTIL (CODE = 1);

  (* IF USER IS REVIEWING, THEN BREAK REST OF MESSAGE*)
  IF (REVIEW = 'R') THEN
    BEGIN
      STR5 := COPY(MESSAGE, 51, 2);
      STR6 := COPY(MESSAGE, 53, 2);
      STR7 := COPY(MESSAGE, 55, 2);
    END;

  (* THIS PR O C E D U R E  TA KES T H E  M E SSA G E  S T R I N G A P A R T A N D
DETERM IN ES T H E  A C T I O N R E Q U I R E D*)

BEGIN
  FIRSTNAME := COPY(MESSAGE, 1, 20);
  STR1 := COPY(MESSAGE, 21, 2);
  STR2 := COPY(MESSAGE, 23, 2);
  STR3 := COPY(MESSAGE, 25, 2);
  STR4 := COPY(MESSAGE, 27, 2);
  NAME := COPY(MESSAGE, 29, 20);
  DIRECTION := COPY(MESSAGE, 49, 1);
  REVIEW := COPY(MESSAGE, 50, 1);
  LESSON := 0; CODE := 0;

  (* DETERMINE WHICH LESSON IS NEEDED*)
  REPEAT
    LESSON := LESSON + 1;
    STR(LESSON, SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR2)) THEN
      SNUM := CONCAT(SNUM, BLANK);
    IF (STR2 = SNUM) THEN CODE := 1;
  UNTIL (CODE = 1);

  (* IF USER IS REVIEWING, THEN BREAK REST OF MESSAGE*)
  IF (REVIEW = 'R') THEN
    BEGIN
      STR5 := COPY(MESSAGE, 51, 2);
      STR6 := COPY(MESSAGE, 53, 2);
      STR7 := COPY(MESSAGE, 55, 2);
    END;

  (* THIS PR O C E D U R E  TA KES T H E  M E SSA G E  S T R I N G A P A R T A N D
DETERM IN ES T H E  A C T I O N R E Q U I R E D*)

BEGIN
  FIRSTNAME := COPY(MESSAGE, 1, 20);
  STR1 := COPY(MESSAGE, 21, 2);
  STR2 := COPY(MESSAGE, 23, 2);
  STR3 := COPY(MESSAGE, 25, 2);
  STR4 := COPY(MESSAGE, 27, 2);
  NAME := COPY(MESSAGE, 29, 20);
  DIRECTION := COPY(MESSAGE, 49, 1);
  REVIEW := COPY(MESSAGE, 50, 1);
  LESSON := 0; CODE := 0;

  (* DETERMINE WHICH LESSON IS NEEDED*)
  REPEAT
    LESSON := LESSON + 1;
    STR(LESSON, SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR2)) THEN
      SNUM := CONCAT(SNUM, BLANK);
    IF (STR2 = SNUM) THEN CODE := 1;
  UNTIL (CODE = 1);

  (* IF USER IS REVIEWING, THEN BREAK REST OF MESSAGE*)
  IF (REVIEW = 'R') THEN
    BEGIN
      STR5 := COPY(MESSAGE, 51, 2);
      STR6 := COPY(MESSAGE, 53, 2);
      STR7 := COPY(MESSAGE, 55, 2);
    END;
PROCEDURE CALL;

(* THIS PROCEDURE DETERMINES WHICH LESSON TO CALL
    AND SENDS THE MESSAGE DOWN TO THAT LESSON*)

BEGIN
    WRITE(CHR(12));
    GOTOXY(2,15);
    WRITE('LESSON COMING UP - WAIT A MOMENT');
    CASE LESSON OF
        1: SETCHAIN('#5:LESSON1');
        2: SETCHAIN('#5:LESSON2');
        3: SETCHAIN('#5:LESSON3');
        4: SETCHAIN('#5:LESSON4');
        5: SETCHAIN('#5:LESSON5');
        6: SETCHAIN('#6:LESSON6');
    END;
    SETCVAL(MESSAGE);
END;

PROCEDURE NEXT;

(* THIS PROCEDURE IS NEEDED TO CALL THE NEXT MODULE - A
    LATERAL MOVEMENT*)

BEGIN
    MESSAGE := CONCAT(FIRSTNAME,STR1);
    MESSAGE := CONCAT(MESSAGE,STR2);
    MESSAGE := CONCAT(MESSAGE,STR3);
    MESSAGE := CONCAT(MESSAGE,STR4);
    MESSAGE := CONCAT(MESSAGE,NAME);
    MESSAGE := CONCAT(MESSAGE,'D. ');

    (* IS USER IS STILL REVIEWING, THEN SEND THIS ON*)
    IF (REVIEW = 'R') THEN
        BEGIN
            MESSAGE := CONCAT(MESSAGE,'R ');
            MESSAGE := CONCAT(MESSAGE,STR5);
            MESSAGE := CONCAT(MESSAGE,STR6);
            MESSAGE := CONCAT(MESSAGE,STR7);
        END;

    WRITE(CHR(12));
    WRITELN; WRITELN;
    WRITELN('NOW INSERT DISKETTE 3 INTO DRIVE 2');
    WRITELN;

WRITE('AND PRESS RETURN \');
READLN(RESPONSE);

SETCVAL(MESSAGE);
SETCHAIN('.#5:MODULE2.');

WRITE(CHR(12));
GOTOXY(5,15);
WRITELN('NEXT MODULE - WAIT ONE MOMENT');
END;

PROCEDURE RETURN;

(* THIS PROCEDURE IS USED FOR RETURNING TO THE MAIN PROGRAM *)

BEGIN
SETCVAL(MESSAGE);
SETCHAIN('#4s:THESIS');
WRITE(CHR(12));
GOTOXY(1,12);
WRITELN('RETURNING TO THE MENU - WAIT ONE MOMENT');
END;

(* MAINLINE *)

BEGIN
BLANK := ' ';
GETCVAL(MESSAGE);
BREAK;

(* DETERMINE WHAT DIRECTION THE USER IS GOING
  U - FOR UP TO MAINLINE
  D - DOWN TO A LESSON
  N - FOR NEXT MODULE *)

IF (DIRECTION = 'N') THEN NEXT;
IF (DIRECTION = 'D') THEN CALL;
IF (DIRECTION = 'U') THEN RETURN;
END.
(*$S++*)

PROGRAM LESSON1;

(*AUTHOR       BOB DUTILLY
DESCRIPTION THIS IS THE FIRST LESSON FOR THE
CAI GRAPHICS PACKAGE  *)

USES CHAINSTUFF,TURTLEGRAPHICS,OVERHEAD,
(*$U #5:GRAPH1,LIBRARY*) UNIT1,UNIT2;

VAR
  RESULTS : FILE OF RECORD
  H : ARRAY [1..8] OF INTEGER;
END;
MESSAGE : STRING[80];
STR1,STR2,STR3,STR4,SNUM : STRING[2];
MODULE,LESSON,RECNUM,CODE,M,L,F : INTEGER;
RESPONSE : STRING;
BLANK,REVIEW : STRING[1];
NAME : STRING[20];

PROCEDURE BREAK;

(* THIS PROCEDURE BREAKS APART THE MESSAGE*)

BEGIN
  GETCVAL(MESSAGE);
  FIRSTNAME := COPY(MESSAGE,1,20);
  STR1 := COPY(MESSAGE,21,2);
  STR2 := COPY(MESSAGE,23,2);
  STR3 := COPY(MESSAGE,25,2);
  STR4 := COPY(MESSAGE,27,2);
  NAME := COPY(MESSAGE,29,20);
  REVIEW := COPY(MESSAGE,50,1);
  MODULE := 1;
  LESSON := 1;
  FRAME := 0;
  CODE := 0;
  (* DETERMINE WHICH FRAME TO CALL*)
  REPEAT
    FRAME := FRAME + 1;
    STR(FRAME,SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR3)) THEN
      SNUM := CONCAT(SNUM,BLANK);
    IF (STR3 = SNUM) THEN CODE := 1;
  UNTIL (CODE = 1);
RECNUM := -1;
CODE := 0;

REPEAT
    RECNUM := RECNUM + 1;
    STR(RECNUM, SNUM);
    IF (LENGTH(SNUM) < LENGTH(STR4)) THEN
        SNUM := CONCAT(SNUM, BLANK);
    END;
    IF (STR4 = SNUM) THEN CODE := 1;
UNTIL (CODE = 1);

(* IF REVIEWING, THEN GET REST OF MESSAGE*)

IF (REVIEW = "R") THEN
BEGIN
    STR1 := COPY(MESSAGE, 51, 2);
    STR2 := COPY(MESSAGE, 53, 2);
    STR3 := COPY(MESSAGE, 55, 2);

    M := 0;
    CODE := 0;
    REPEAT
        M := M + 1;
        STR(M, SNUM);
        IF (LENGTH(SNUM) < LENGTH(STR1)) THEN
            SNUM := CONCAT(SNUM, BLANK);
        END;
        IF (SNUM = STR1) THEN CODE := 1;
    UNTIL (CODE = 1);

    L := 0;
    CODE := 0;
    REPEAT
        L := L + 1;
        STR(L, SNUM);
        IF (LENGTH(SNUM) < LENGTH(STR2)) THEN
            SNUM := CONCAT(SNUM, BLANK);
        END;
        IF (SNUM = STR2) THEN CODE := 1;
    UNTIL (CODE = 1);

    F := 0;
    CODE := 0;
    REPEAT
        F := F + 1;
        STR(F, SNUM);
        IF (LENGTH(SNUM) < LENGTH(STR3)) THEN
            SNUM := CONCAT(SNUM, BLANK);
        END;
        IF (SNUM = STR3) THEN CODE := 1;
    UNTIL (CODE = 1);
END;
PROCEDURE CALL;

(* THIS PROCEDURE ACTUALLY CALLS EACH FRAME IN THE
LIBRARY - UNIT 1 AND UNIT 2*)

BEGIN

(*$N++*)

REPEAT

CASE FRAME OF
  1: FRAME1;  2: FRAME2;  3: FRAME3;
  4: FRAME4;  5: FRAME5;  6: FRAME6;
  7: FRAME7;
END;

(* CANNOT HAVE THE SAME STRING VARIABLE IN BOTH
UNITS, THEREFORE HAVE TWO STRINGS THAT RETURN
A VALUE TO THE LESSON*)

REPEAT
  IF (FRAME < 5) THEN RESPONSE := R1 (*UNIT1 OR 2 RESPONSE*)
  ELSE RESPONSE := R2;
  RESPOND := RESPONSE;
  CODETYPE := CODE;
  CHECK(RESPOND,CODETYPE); (*CHECK WHAT WAS INPUT*)
  CODE := CODETYPE;

  (* DETERMINE WHAT ACTION TO TAKE*)

  CASE CODE OF
    1 : ; (*CARRIAGE RETURN*)
    2 : FRAME := FRAME - 2; (*BACK UP*)
    3 : BEGIN (*IN DEPTH CONCEPTS*)
      IF (FRAME = 3) THEN FRAMED3;
      IF (FRAME = 5) THEN FRAMED5;
    END;
    4 : BEGIN
      HELP; (*HELP PROCEDURE*)
      FRAME := FRAME - 1;
    END;
    5 : EXIT(CALL); (*QUIT THE PROGRAM*)
    6 : BEGIN (*ERROR IN RESPONSE*)
      WRITELN('SORRY, I DO NOT UNDERSTAND, PLEASE RETYPE.');
      READLN(RESPONSE);
    END;
  END;
END;
UNTIL (CODE <> 6);

FRAME := FRAME + 1;
IF (FRAME <= 0) THEN FRAME := 1;

UNTIL (FRAME >= 8);
(*$N-**)
END;

PROCEDURE OUTPUT;

(* THIS PROCEDURE UPDATES THE RESULTS FILE WITH THE
QUIZ RESULTS FROM LESSON 1*)
BEGIN
(*$I-**)
RESET(RESULTS, 'F4:RESULTS.DAT');
IF (IORESULT <> 0) THEN
BEGIN
CLOSE(RESULTS, LOCK);
OUTPUT;
END;
(*$I++*)
SEEK(RESULTS, RECNUM);
GET(RESULTS);
RESULTS.H[1] := RESULT;
SEEK(RESULTS, RECNUM);
PUT(RESULTS);
CLOSE(RESULTS, LOCK);
END;

PROCEDURE RETURN;

(* THIS PROCEDURE EITHER RETURNS TO THE CALLING
MODULE OR CALLS THE NEXT LESSON FOR LATERAL
MOVEMENT*)
BEGIN
(* NEXT LESSON*)
IF (FRAME >= 8) THEN
BEGIN
LESSON := LESSON + 1;
FRAME := 1;
END;

(* SETUP MESSAGE WITH VALUES*)
STR(MODULE, STR1);
STR(LESSON, STR2);
STR(FRAME, STR3);
STR(RECNUM, STR4);

IF (LENGTH(STR1) = 1) THEN STR1 := CONCAT(STR1, BLANK);
IF (LENGTH(STR2) = 1) THEN STR2 := CONCAT(STR2, BLANK);
IF (LENGTH(STR3) = 1) THEN STR3 := CONCAT(STR3, BLANK);
IF (LENGTH(STR4) = 1) THEN STR4 := CONCAT(STR4, BLANK);

FOR I := 20 DOWNTO LENGTH(FIRSTNAME) + 1 DO
BEGIN
  FIRSTNAME := CONCAT(FIRSTNAME, BLANK);
END;

FOR I := 20 DOWNTO LENGTH(NAME) + 1 DO
BEGIN
  NAME := CONCAT(NAME, BLANK);
END;

MESSAGE := CONCAT(FIRSTNAME, STR1);
MESSAGE := CONCAT(MESSAGE, STR2);
MESSAGE := CONCAT(MESSAGE, STR3);
MESSAGE := CONCAT(MESSAGE, STR4);
MESSAGE := CONCAT(MESSAGE, NAME);

(* IF CODE IS A 5 THEN RETURN TO MODULE 
  OR CALL NEXT LESSON*)

IF (CODE = 5) THEN MESSAGE := CONCAT(MESSAGE, "R");
ELSE MESSAGE := CONCAT(MESSAGE, "D");

IF (REVIEW = "R") THEN
BEGIN
  STR(M, STR1);
  STR(L, STR2);
  STR(F, STR3);
  IF (LENGTH(STR1) = 1) THEN STR1 := CONCAT(STR1, BLANK);
  IF (LENGTH(STR2) = 1) THEN STR2 := CONCAT(STR2, BLANK);
  IF (LENGTH(STR3) = 1) THEN STR3 := CONCAT(STR3, BLANK);

  MESSAGE := CONCAT(MESSAGE, "R");
  MESSAGE := CONCAT(MESSAGE, STR1);
  MESSAGE := CONCAT(MESSAGE, STR2);
  MESSAGE := CONCAT(MESSAGE, STR3);
END;

IF (CODE = 5) THEN SETCHAIN("#5:MODULE1");
ELSE SETCHAIN("#5:LESSON2");

SETCVAL(MESSAGE);

END;
BEGIN
  BLANK := ' ';
  RESULT := -1;
  BREAK;
  CALL;
  IF (REVIEW = 'R') AND (MODULE >= M) AND (LESSON >= L) AND (FRAME >= F)
     THEN REVIEW := ' ';
  IF (REVIEW <> 'R') THEN
    BEGIN
      IF (RESULT <> -1) THEN OUTPUT;
      (* UPDATE THE LOCATION FILE WITH THE NEW DATA
         THE FILE UPDATE PROCEDURE IS IN THE SYSTEM
         LIBRARY UNDER OVERHEAD*)
      UPDATE(RECNUM, MODULE, LESSON, FRAME);
      END;
    RETURN;
  END.
END.
(*$S++*$)

UNIT UNIT1;

(*AUTHOR   BOB DUTILLY
DESCRIPTION  THIS IS UNIT ONE OF THE CAI
COMPUTER GRAPHICS PACKAGE. *)

INTERFACE
USES CHAINSTUFF, TURTLEGRAPHICS;

VAR
  FRAME, I, J, K : INTEGER;
  R1 : STRING;
  FIRSTNAME : STRING[20];

PROCEDURE FRAME1;
PROCEDURE FRAME2;
PROCEDURE FRAME3;
PROCEDURE FRAME4;
PROCEDURE FRAME4A;
PROCEDURE FRAME4B;
PROCEDURE FRAME4C;

IMPLEMENTATION
VAR
  BLANK : STRING[1];

PROCEDURE FRAME1;

(*  THIS IS THE FIRST PROCEDURE IN THE
   GRAPHICS PACKAGE AND EXPLAINS ABOUT
   GRAPHICS.*)

BEGIN
  WRITE(CHR(12)); WRITELN;
  WRITELN('IN THIS FIRST LESSON WE WILL COVER');
  WRITELN;
  WRITELN('THE BASIC CONCEPTS OF COMPUTER GRAPHICS');
  WRITELN;
  WRITELN('SEVERAL DEFINITIONS, SOME EXPLORATIONS');
  WRITELN;
  WRITELN('OF GRAPHIC COMMANDS AND OF COURSE SOME');
  WRITELN;
  WRITELN('QUIZZES ON THE MATERIAL COVERED.');
  GOTOXY(3,19);
  WRITE('PLEASE PRESS RETURN OR H, Q ');
  READLN(R1);
END;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
PROCEDURE FRAME2;

(* THIS PROCEDURE CONTINUES THE EXPLANATION OF GRAPHICS*)

BEGIN
  WRITE(CHR(12)); WRITELN;
  WRITELN("INSTRUCTIONS WILL BE GIVEN TO YOU");
  WRITELN;
  WRITELN("THROUGHOUT THE COURSE,_USUALLY THE");
  WRITELN;
  WRITELN("CARRIAGE RETURN OR A ONE WORD RESPONSE");
  WRITELN;
  WRITELN("IS ALL THAT IS NEEDED. THERE IS AN");
  WRITELN;
  WRITELN("OPTION OF A DEPTH RESPONSE AT VARIOUS");
  WRITELN;
  WRITELN("SECTIONS THROUGHOUT THE COURSE. BY");
  WRITELN;
  WRITELN("THIS OPTION, FURTHER DISCUSSION OF THE");
  WRITELN;
  WRITELN("MATERIAL CAN BE OBTAINED.");
  GOTOXY(2,20);
  WRITE("PRESS RETURN TO CONTINUE OR B, H, Q");
  READLN(R1);
END;

PROCEDURE FRAME3;

(* THIS IS THE BEGINNING OF THE MATERIAL TO BE LEARNED*)

BEGIN
  WRITE(CHR(12)); WRITELN;
  WRITELN("COMPUTER GRAPHICS IS THE AREA OF");
  WRITELN;
  WRITELN("PROGRAMMING IN WHICH DATA, EITHER");
  WRITELN;
  WRITELN("GIVEN OR CALCULATED, IS GRAPHICALLY");
  WRITELN;
  WRITELN("REPRESENTED BY SOME TYPE OF PICTURE.");
  WRITELN;
  WRITELN;
  WRITELN("THIS AREA OF RESEARCH STARTED IN");
  WRITELN;
  WRITELN("1962 WITH THE FIRST SUCCESSFUL");
  WRITELN;
  WRITELN("INTERACTIVE GRAPHICS SYSTEM BY");
  WRITELN;
  WRITELN("IVAN SUTHERLAND.");
  WRITELN;
  WRITELN;
  WRITE("PLEASE PRESS RETURN OR B, D, H, Q");
READLN(R1);
END;

PROCEDURE FRAMED3;

(* THIS IS AN IN DEPTH FRAME THAT THE USER HAS AN OPTION OF USING. IT IS ALSO A GRAPHICS FRAME.*)

BEGIN
(* A BASIC PICTURE IS DRAWN TO DESCRIBE THE INTERACTION*)
INITTURTLE;
MOVETO(15,170);
WSTRING('THE PROCESS OF INTERACTIVE GRAPHICS');
MOVETO(15,155);
WSTRING('INVOLVES A TWO WAY COMMUNICATION');
MOVETO(15,140);
WSTRING('BETWEEN THE USER AND COMPUTER. THE');
MOVETO(15,125);
WSTRING('USER CONTROLS THE OUTPUT BY A SERIES');
MOVETO(15,110);
WSTRING('OF COMMANDS.');
MOVETO(2,90);
WSTRING('INITIAL');
MOVETO(2,82);
WSTRING('PICTURE');
MOVETO(50,87);
PENCOLOR(BLUE);
MOVETO(55,87);
PENCOLOR(NONE);
MOVETO(60,87);
PENCOLOR(BLUE);
MOVETO(65,87);
PENCOLOR(NONE);
MOVETO(70,87);
PENCOLOR(BLUE);
MOVETO(75,87);
PENCOLOR(NONE);
MOVETO(73,90);
PENCOLOR(BLUE);
MOVETO(78,87);
MOVETO(73,84);
PENCOLOR(NONE);
MOVETO(81,87);
WSTRING('COMMAND');
MOVETO(134,87);
PENCOLOR(ORANGE);
MOVETO(158,87);
PENCOLOR(NONE);
MOVETO(155,90);
BEGIN

(*MAINLINE*)

BEGIN

END.
BIBLIOGRAPHY

Books


Holland, James G.; Solomon, Carol; Doran, Judith; Frezza, Daniel A. *The Analysis of Behavior in Planning Instruction*. Philippines: Addison-Wesley, 1976.


Journals and Articles


Unpublished Materials
