

## **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.

**University  
Microfilms  
International**

300 N. Zeeb Road  
Ann Arbor, MI 48106



8528021

**McDaniel, John William**

**A MODEL FOR INSTRUCTIONAL SOFTWARE PROGRAMMING CONCEPTS**

*Kansas State University*

**Ph.D. 1985**

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



A Model for Instructional Software Programming Concepts

by

JOHN WILLIAM MCDANIEL

B.S. East Central State College - Ada, Oklahoma, 1972

M.S. Oklahoma State University - Stillwater, Oklahoma, 1975

---

A     D I S S E R T A T I O N

submitted in partial fulfillment of the

requirements for the degree

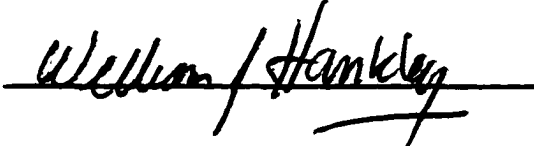
DOCTOR OF PHILOSOPHY

Department of Computer Science

Kansas State University

1985

Approved by:

  
A handwritten signature in cursive script, reading "William Hambley", is written over a horizontal line.

Major Professor

**A Model for Instructional Software Programming Concepts**

**JOHN WILLIAM MCDANIEL**

**1985**

## ABSTRACT

This dissertation presents a paradigm by which specific computer programming concepts can be conveyed to beginning students in a game-like environment. This model emphasizes student-computer interaction, rather than student-teacher interaction. The model integrates four instructional modes which have been effective separately in other systems: a tutorial mode, which is used in conventional CAI, an interactive graphic model, which has been the key to popular video games and newer CAI programs; and program execution mode, which has been the cornerstone for programming laboratory work; and a reductionistic (incremental) structure, which is the accepted structure for texts in computer science. To cover all programming concepts in a beginning course, numerous topic learning modules would have to be developed. (They are not all covered in this work.) To illustrate the paradigm, three specific modules are developed: a Puzzle Module covering sequence and iteration, a Business Module covering procedures and procedure concepts, and a Grade Point Average Calculator Module covering array notation. Two of the modules (The Puzzle Module, and The Business Module) have been implemented to run on micro-computers and one of them (The Puzzle Module) has been evaluated in a classroom environment. Finally, guidelines are included for implementing further modules.

## TABLE OF CONTENTS

0	Title
0.1	Abstract
0.2	Table of Contents
0.3	List of Figures
0.4	Acknowledgments
1	Chapter 1 - Introduction
1.1	The Problem: the need for CAI in teaching programming
1.2	The Requirements: Necessary Characteristics for CAI for programming
1.3	Contribution: A model which integrates these CAI features
1.4	Organization of this paper
2	Chapter 2 - Literature Review and Foundation Concepts
2.1	Introduction
2.2	General Concepts of CAI
2.2.1	Introduction
2.2.2	Advantages of CAI
2.2.3	Disadvantages of CAI
2.2.4	Conclusion
2.3	CAI as an Instructional or Tutorial Mode for Teaching Programming.
2.3.1	Tools for Software Education
2.3.2	Interactive Environments
2.3.2.1	Logo
2.3.2.2	Karel the Robot
2.3.3	Structured Environments
2.3.3.1	Programming Made Simple
2.3.3.2	Cornell Program Synthesizer
2.3.3.3	MacPascal
2.3.3.4	A Programming Environment vs A Structured Environment
2.3.3.5	Conclusion
2.4	CAI Graphics
2.4.1	Graphics as a teaching tool
2.4.2	Categories of Graphics Packages
2.4.3	Conclusion
2.5	Reductionism vs Structuralism
2.5.1	A reductionistic approach
2.5.2	Wholism (or Structuralism )
2.6	Conclusions
3	Chapter 3 - The Model
3.1	Introduction
3.2	Modules Built According to the Model
3.2.1	The Puzzle Module
3.2.2	An Overview of the GPA Module
3.2.3	The GPA Module
3.2.4	The Business Module
3.3	Justification
4	Evaluation of One Programming Module
4.1	Introduction
4.2	Experiment Decisions
4.3	Details of the Experiment



- 4.3.T1 The attitudes of the two groups were the same, a priori
- 4.3.T2a Positive Reaction by the Basic Students (stu-ques-class)
- 4.3.T2b Positive Reaction by the Basic Students (ques-stu-class)
- 4.3.T3a FL/I Class had a positive change in attitude (stu-ques-class)
- 4.3.T3b FL/I Class had a positive change in attitude (ques-stu-class)
- 4.3.T4 The Change For the Basic Class was greater than the FL/I Class
- 4.4 Conclusions
- 5 Conclusions of this work
- 5.1 Review of the work
- 5.2 Guidelines
- 5.3 Future Work

#### Appendices

- A Instructions for the operation of the micro-computer.
- B The questionnaire given to all of the students in both classes
- C A listing of the responses
- D A summary of the data by question.
- E The results from the statistical analysis.
- F The data with the responses of those students who omitted their Social Security Number on June 15 removed.
- G The results of the responses to the questions by the FL/I class.
- H The results of the responses to the questions by the Basic class.
- I Application for Approval to use Human Subjects
- J The classification of the various questions in the questionnaire.

#### Bibliography

## LIST OF FIGURES

### Figure Title

- 3.a. An overview (of the levels and the phases within each level).
- 3.b. Sequence of events for proceeding (or digressing) from one level to another level.
- 3.c. Initial display on the screen, showing the courier
- 3.d. The screen after the courier has retrieved a purchase order
- 3.e. The screen after the data base has been updated
- 3.f. The screen after the inventory has been adjusted
- 3.g. Initial display of the screen for level 2, showing the data base
- 3.h. The screen after the purchase order has been retrieved
- 3.i. The screen after the inventory record has been retrieved
- 3.j. The screen after the inventory record has been updated
- 3.k. The screen after the reorder form has been generated
- 5.a. The set of concepts that are taught at a given level should be "closed".

## Acknowledgments

I would like to take this space to express my appreciation to Dr. William Hankley, my major professor, for his council during the writing phase of this dissertation.

## Chapter 1 - Introduction

### 1.1 The Problem: the need for CAI in teaching programming

Decreasing costs and immediate feedback have combined to make electronic games increasingly popular in recent years. More and more educators have come to the conclusion that teachers in a variety of disciplines can capitalize on the popularity of electronic games when conveying new concepts to less than interested students [Harr 81].

It is recognized that learning computer programming generally requires strong supervision. This supervision is referred to as the 'hidden lab' [Tuts 81]. Unfortunately, the academic community is unable to give adequate individual supervision because of the huge enrollments in many computer science classes.

The large class sizes in computer science can be attributed to the following causes:

a) The increase in the number of computer science majors over the past several years. Dennings article is dated but the following trend still remains. The number of undergraduate majors in computer science doubled during the period 1975 to 1981 [Denn 81] and a similar rate of increase is expected in the future.

b) The increase in other disciplines requiring introductory computer science as a core course. More and more educators (from schools and departments other than computer science) are recognizing the need for their majors to have an increasing amount of computer literacy as it relates to

their disciplines [Pres 83].

c) Not many graduate PhD's in Computer Science select teaching as a career. The half life of the material that is to be taught (with respect to other disciplines like math or physics) combined with the 'hidden lab' associated with lower level programming courses has made teaching computer science unattractive. During the period 1975-1979, there was virtually no increase in the number of new PhD's taking academic positions at PhD granting institutions, whereas the undergraduate enrollment increased from 15 to 20 percent, annually. This has resulted in some universities cutting enrollments and limiting class sizes [Denn 81], while other universities allow the number of students per class to grow without bound. The resultant increase in the teaching workload has caused many to retire from academic life [Tuts 81]. Denning has described the manpower shortage in the computing field as severe [Denn 81b].

The above factors have resulted in large, nearly un-manageable computer science classes with a concomitant increase in the teacher workload. These factors, along with the reluctance of some university administrators to allocate scarce resources in an area as new and volatile as computer science, create a serious problem in Computer Science education. The solution to this problem requires a shift to computer aided instruction of introductory computer science concepts.

#### 1.2 The Requirements: Necessary Characteristics for CAI for programming

The question of this research then is: What is needed for CAI for teaching computer science concepts? My answer to this question was found

by an examination of the hardware (and software) in current use, along with an examination of the literature in the areas of computer science and education, specifically computer aided instruction. Four ideas stand out as essential features for CAI packages: a reductionistic organization, an interactive graphics model, an instructional mode, and an execution mode. It is my conclusion that not all systems have these four modes. They became evident after studying the literature and searching for instructional patterns.

A brief description of the four modes follows:

- Reductionism is used as the basis of most textbooks that teach computer science and is the generally accepted approach. A reductionistic approach is one that introduces key concepts one concept at a time, in isolation and then uses these concepts to build on, as the student is exposed to more advanced and difficult material.
- Computer Graphics offer a high bandwidth for transfer of information. This is evident from the literature and from the emerging technology of instructional packages on micro-computers.
- An Instructional mode is necessary with any package used as a supplement to classroom instruction.
- An Executable mode is a requirement due to the nature of the discipline. A student cannot do any useful work without a programmable system, and execution of programs forms the cornerstone of computer science.

The foundation of these four requirements and the background literature supporting them is given in Chapter 2 of this dissertation.

### 1.3 Contribution: A model which integrates these CAI features

This dissertation presents a model which integrates reductionism, instruction, graphics and execution into one coherent package, an approach not currently being utilized in the instructional programs in the computer science curriculum.

Important characteristics of the model are the following:

- (1) The model is designed to provide motor-learning experiences utilizing immediate feedback that encourages adaptation and imitation.
- (2) Capitalizing upon the appeal of the interactive nature of computers, the model uses a 'friendly' environment in which concepts familiar to the student are translated, automatically, into more formal programming statements and then these formal statements are, subsequently, displayed on a student terminal.
- (3) The model is designed around the idea that beginning programmers should learn their craft by example rather than by trial and error. A problem is presented and the student solves the problem using simple or already familiar concepts. While the student is solving the problem, in familiar terms, a 'program' is introduced, automatically, which shows an equivalent solution using a more formal notation. In this way, program syntax, and some semantic rules, can be conveyed on a real-time basis without an inordinate amount of student-teacher interaction.

(4) This model consists of several levels. Each level (consisting of a sequence of nine phases) is designed to convey more advanced concepts than the previous level. The student will work through the phases of a given level until the completion of that level. The phases are designed so that the student progresses (in a reductionistic manner) by building on previously learned material. Each phase has been designed to 'walk' the student through a solution of a problem using a computer, beginning with the student's own primitive solution and finishing up with a formal algorithmic text. Within the design is a 'backing up' feature that allows the student to see previously covered material at a lowered numbered level for review or for reinforcement.

This model is not what is commonly referred to as CAI (Computer Aided Instruction). CAI generally has two alternating phases: a tutorial phase followed by a question/answer phase. The package described in this paper presents a computer directed motor learning experience that 'guides' the user through several phases (at various levels) culminating in the development (by the user) of syntactically and semantically correct computer algorithms.

It is recognized that several modules are required to build the package described in the previous paragraph. This dissertation describes the implementation of a subset of the modules necessary to cover the entire spectrum of computer science concepts taught at the introductory level.

There are two modules implemented according to the model, a Puzzle



module and a Business module (involving a data base). These modules will be used to convey different computer science concepts. The model can be successfully used in the design of a variety of interactive modules.

The following interactive modules have been designed:

The Puzzle Module - a jig-saw puzzle (with square corners and 4 pieces) appears on the screen. The student is shown the use of several keys on the keyboard (and/or other items of hardware that is available - a joystick, for example) and directed to move the pieces about the screen using the keys (or joystick) until the puzzle has been assembled. This exercise is used to relate puzzle manipulations to algorithmic processes, and to introduce looping, decision and input/output programming constructs. The Puzzle module is implemented on an ATARI micro-computer and was used by an instructor of CS-200 in a classroom environment. An analysis of the responses of the students, regarding the suitability of the package appears in a final chapter of this paper.

The Business Module - the computer displays a hypothetical and quite primitive office, with an input, a database manager, an inventory and a mail box. The student is directed to move sheets of paper (records) from the input to the database manager and then on to the inventory manager or to the mail box. These 'office operations' have counterparts in many programming languages and are considered to be useful programming concepts. This module introduces procedures and parameter passing, and was fully implemented but was not tested in a classroom environment.

The Grade Point Average Module - the student is directed to key in (on a keyboard) 'the number of hours' and 'letter grade received' (using a four point system). The program will display (upon command) total grade points received and a total GPA for the student. This approach is widely used in conventional programming classes and is used to convey to the student the following concepts: 'assignment statements', arrays, input/output, formatting and editing. The design for this module appears in Chapter 3 of this dissertation, but the module was neither implemented nor tested in a classroom environment.

Walker [Walk 76] states that classroom instruction is highly ego satisfying. The classroom instructor will not be willing to step down as the central authority figure (and the undisputed source of knowledge in a classroom) to the role of class monitor. Therefore, the package described in this paper is intended to be used as a supplement to classroom instruction.

#### 1.4 Organization of this paper

This paper is organized in the following manner. The introduction establishes that there exists a problem of educating increasing numbers of students in computer science with an (almost) non-changing number of faculty. Also, appearing in the introduction are the requirements of designing a CAI package to be used by a computer science instructor. Chapter 2 is a review of related work involving Computer Aided Instruction, especially as it relates to game-playing. Chapter 3 gives a detailed description of the model, followed by a description of the modules built according to the model. In Chapter 4, there is a detailed evaluation of the Puzzle module to determine the effectiveness of the software package. Guidelines for an implementation are then given followed by a conclusions chapter and a bibliography.

## Chapter 2 - Literature Review and Foundation Concepts

### 2.1 Introduction

Initially, the literature was examined to find existing work in CAI for teaching about computer science, particularly programming, and to find guidance and principles for the model presented in Chapter 3. Four principles were identified for the model. In order to show how the current literature relates to the model described in this dissertation, this chapter is organized following the four principle goals of the model:

- general concepts of CAI
- CAI for teaching programming
- CAI graphics
- CAI reductionistic structure

Each section concludes with the principles that are used for the model in chapter 3.

### 2.2 General Concepts of CAI

#### 2.2.1 Introduction

CAI (Computer Assisted Instruction) is a natural electronic extension of PSI, a Personalized System of Instruction developed by Fred Keller [Kell 68]. PSI involves the following concepts: 1) the student uses, primarily, written materials, 2) the student masters one level of material before continuing on to the next level, 3) the text is self-paced, in the sense that the student is allowed to proceed at his or her own rate, and

4) there are proctors available to provide someone help. With the introduction of computers, Keller's ideas became even more widespread and are used extensively in many CAI packages.

CAI is defined to be "a teaching process directly involving the computer in the presentation of instructional materials in an interactive mode to provide and control the individualized learning environment for each individual student" [Spli 79]. Belt, et.al., adds that the student is online to a computer, and that a "stimulus is presented, responses are accepted and processed, feedback is provided to the student, and the computer maintains various degrees of control over the sequencing of material" [Belt 76].

In contrast, CMI (Computer Managed Instruction) is defined to be: "an instructional management system utilizing the computer to direct the entire instructional process, including perhaps CAI as well as traditional forms of instruction.... CMI has some or all of the following characteristics: organizing curricula and student data, monitoring student progress, diagnosing and prescribing, evaluating learning outcomes, and providing planning information for teachers" [Spli 79].

Initially, the advocates of CAI borrowed an idea from individualized instruction. This idea was that the process be self-paced, i.e. the learner proceeds at his or her own rate and according to his or her own ability, and usually under the guidance of a teacher [Blak 69]. This idea is still predominant in the packages available at this time, and the package proposed in this dissertation reflects this same approach.

Van Hees [VanH 76] identifies six types of CAI that are in current use. These six types are shown below. The types of CAI that are of interest and are incorporated into the design of the model in chapter 3 have been marked with an asterisk.

- |   |                        |
|---|------------------------|
|   | 1 drill and practice   |
| * | 2 tutorial instruction |
|   | 3 inquiry              |
| * | 4 problem-solving      |
|   | 5 simulation           |
| * | 6 gaming               |

#### Six Types of CAI

Drill and practice sessions have questions and answers similar to:  
Press the correct letter to signify which answer is the capital of Texas?  
a. Dallas   b. Austin   c. San Antonio   d. Houston

Drill and practice is one of the earliest applications for teaching with computers. This approach does not appear in the model described in chapter 3 of this dissertation.

With tutorial instruction, a block of material is displayed on the screen, with the explanation of some feature of, say, a software package. A given command is displayed, then the results of the command appears on the screen as if the user had pressed the command. The tutorial approach is not interactive and therefore was not used in the model appearing in chapter 3.

Inquiry systems are simple question and answer systems. The user can key in some keyword (Kansas, for example) and the capital (or some other

information) will be displayed.

Early problem solving systems were used to teach math students fundamentals of algebra. A typical question is:  $x + 7 = 9$ . What is  $x$ ? If the student keys in the correct answer, he/she will progress to more advanced material, if not then previous material will be reviewed.

Simulation systems for a variety of disciplines are wide-spread. An example system allows the user to enter the topology of a given electronic circuit, and the properties of the circuit are displayed. Changes can then be entered allowing the user to experiment with different designs.

Gaming includes any type of program that pits the student against an adversary, whether it be the machine or another human. In order to win the game, the user is required to apply a prescribed strategy, preferably one that relates to the course of material.

All of the above types may include an authoring mode, a program or a set of programs which allow an individual teacher to enter new lessons into the system at their own discretion without programming [Stol 68]. The package described in this dissertation does not include an authoring mode.

### 2.2.2 Advantages of CAI

Chambers lists several advantages of CAI over other conventional learning methods [Cham 80]:

- It involves the individual actively in the learning process.
- It is self-paced.

- Reinforcement is immediate and automatic.
- It allows investigation into problems which would be costly or not possible without the use of computers.
- CAI frees the faculty from redundancies (such as bookkeeping) and allows them more time to devote to personal human considerations of the students.
- CAI helps in remedial training of bilingual and other disadvantaged students.
- CAI can be cost saving, in some cases[see also Hart 71 & Solo 74].
- CAI reduces learning time when compared to the regular classroom.

Riedesel lists eight additional advantages of CAI, particularly in comparison with other programmed devices [Ried 67]. In summary, the eight advantages are:

- The computer carefully controls the learning sequence of each student and requires the student to comprehend each frame.
- The computer can judge constructed responses for accuracy.
- The computer may offer a more stimulating learning situation than is sometimes provided by programmed texts.
- The computer can utilize background information on each student for constructing learning sequences and judging responses.
- The computer is more versatile than the programmed text. It can teach a wider variety of tasks and employ a wider range of auxiliary stimulus-presentation equipment.
- There is a great deal of interest in the use of a guided discovery approach to teaching in today's schools. The author of a CAI se-



quence is able to use the same type of guided questioning that is typical of guided discovery patterns. In fact, it has an advantage over a guided discovery discussion, since each learner must respond to each question, experiencing discovery himself.

- The computer offers data on the entire learning session as well as summary information. These data can be useful in revision of a programmed sequence as well as for school records and research purposes. Students can use their 'type-outs' for study. Also, student records can be very useful in analyzing the thinking patterns of students.
- The computer is a long-term investment that may be used for a variety of purposes, such as data processing. It may be less expensive and less space-consuming than programmed texts.

### 2.2.3 Disadvantages of CAI

The CAI approach is not the panacea the educational community anticipated. The spectrum of CAI materials is incomplete. There is no standard, high-level, complex CAI language which is machine independent, and which combines authoring aids with graphics capabilities [Chen 80]. Sharing of quality CAI materials is virtually non-existent, and the resources needed to develop an appropriate CAI package exceeds the budget of many potential users. Much more applied research at the interface between cognitive science and computer science is needed for the creation of CAI tools [Dunc 81].

Finally, it is not clear that widespread acceptance of radical teach-

ing practices will be forthcoming. It appears that instructional paradigms will evolve, meeting the educational needs of the population and adjusting to changes in the demographics as they occur. Experience with PLATO and TICCIT confirms this belief [Alde 78].

The following paragraphs describe two example CAI systems currently in use by the academic community.

PLATO (Programmed Logic for Automatic Teaching Operations) has been described as "a large educational computing network" [Alde 78], although it is not a network in the conventional sense. A centralized computer, located at the University of Illinois in Urbana, services nearly a thousand terminals that are attached directly (or with a common carrier) to a CDC Cyber 73-74.

The designers of Plato decided to present the material in small units called lessons (or modules). Several lessons constitute a course -and sets of courses make up a curricula. The central ideas behind PLATO is one of providing 1) courseware for helping students learn a variety of topics (accounting, biology, chemistry, English, math, etc) and 2) a means whereby individual teachers can enter new lessons into the system at their own discretion, using an authoring system called TUTOR. The majority of the instructors judged the lessons to be adequate for their students. As might be expected, the PLATO system did not completely replace the teachers in the classroom. The lessons were either integrated into the study plan or were used as supplementary teaching aids. In Alderman's study, less than one-third of the material was initially covered using CAI mode. [Alde 78].

As a result, Plato is moderately but not overwhelmingly successful. One of the problems, as perceived by this author, is one of distribution of information. As Ullmer argues, whenever a learning activity requires centralization (as typified by a library system or a centralized computing system) then the student is hindered from exposure to the material. "These diverse and disconnected types of activities, having little room for participation and control by students, simply do not constitute a desirable and efficient form for an instructional system." [Ullm 69]

Communication with the centralized computer requires, in many cases, a long-distance phone call (to Illinois), an expense many schools find undesirable.

With the proliferation of low-cost micro-computers, obviating the need for a centralized computer, the acceptance of PLATO and like services are becoming more widespread.

Davidson [Davi 81] describes Pilot as "the most widespread small computer CAI language" that has a calculation mode, graphics capabilities and that is completely machine independent. In the latter regard, Pilot has been implemented on many micro-computers. According to the author, Common Pilot is easy to learn and there are courses in Pilot that teach Pilot, a desirable attribute.

#### 2.2.4 Conclusion

The following list contains those ideas from the various sources that relate to CAI that have influenced or have been incorporated into the

design of the software package described in chapter 3 of this dissertation.

- micro-computers are used,
- there is some tutorial instruction,
- there is some problem-solving involved,
- there is some gaming involved,
- the material is self-paced,
- the teaching process directly involves the computer in the presentation of instructional materials in an online, interactive mode,
- stimuli are presented, responses are accepted and processed,
- feedback is provided to the student, reinforcement is immediate and automatic,
- the computer carefully controls the learning sequence of each student and requires the student to comprehend each frame,
- the individual is actively involved in the learning process,
- the student masters one level of material before continuing on to the next level,
- The computer judges constructed responses for accuracy,
- the program is written in Pascal and Basic making the software somewhat machine independent.

## 2.3 CAI as an Instructional or Tutorial Mode for Teaching Programming.

### 2.3.1 Tools for Software Education

Donovan [Dono 76] describes some of the tools that have been developed that relate specifically to software education. The tools include such packages as simulators, graders, compilers and monitors. The article is

dated, and there has been significant progress since 1976, but the instructional goals and problems mentioned are basically the same.

(a) Simulators - Donovan's group developed a simulator for IBM/360 Assembler that would mimic not only user-level 360 instructions but also privileged instructions and detailed input/output and timing. The goal was to teach problem solving in assembly language, using primarily, a digital computer and staying within the bounds of reasonable costs. According to their report, their efforts were successful. This approach has not been used in this dissertation.

(b) Graders - Corresponding to each problem given to the students, there exists a grading program. This program handles all I/O for the student, furnishes all the sample input (test data), and compares the output with known answers, to be used for grading. Actually, Donovan's group developed a 'meta' grading program "with which an instructor can generate his own grading program...for a large spectrum of student problems" [Dono 76, p. 431]. This approach has not been used in this dissertation.

(c) A language compiler, not necessarily friendly, is the programmers basic tool, without which very little computing would be accomplished. Some of the early compilers produced very little useful diagnostic help. On the IBM 1130, for example, some I/O errors were indicated by: 1) the program in a stopped state, and 2) the error displayed in the console lights on the CPU. Some of the compilers developed during the last few years are somewhat more helpful to the student.

(d) Donovan's group developed program monitors which allow the batch-

ing of student compilations and executions as a single job step, thereby avoiding some step overhead associated with each student run, reducing costs. The monitors also were able to recover from most programming errors so as to continue on with the batch run.

### 2.3.2 Interactive Environments

In the last few years, development of packages for software instruction has taken quite a different turn. The packages are more interactive, rather than batch oriented, and they tend to offer graphic displays. and the environments are becoming more structured in a reductionistic fashion. Parenthetically, these newer packages are not CAI systems as described earlier, but, rather, are special language systems.

This section surveys special computer aided systems for teaching programming.

Simple interactive environments with graphics displays are widespread. Two of the more popular packages (Logo, and Karel the Robot) are described below.

#### 2.3.2.1 Logo

Seymour Papert headed a group at MIT that developed a user-friendly problem solving language they called LOGO. [Abel 82b]. Logo was designed to be simple, to give people control over powerful computing resources, and, concurrently, to teach 'advanced' programming concepts to novice programmers. These concepts include the following - procedures, recursion,

list processing (specifically, hierarchy of objects), and language extensibility. Logo has been implemented on a variety of micro-computers including the TI99/4A, the TRS-80 and the Apple II, and features Turtle Graphics, a term coined by the authors that refers to the computer simulation of a robot [Nels 81].

The turtle is a triangular shaped icon appearing on the screen that can be manipulated by the user. The turtle is an abstraction of a 'real' turtle, a hemispherical shaped mechanical device (with three wheels) that can be made to move around on the floor.

A basic set of primitive commands that are used to control the turtle are listed below. For a more complete list of Logo commands, refer to the August 1982 issue of Byte Magazine (p.282).

- forward n
- left d
- right d
- back n
- penup
- pendown
- hide turtle
- repeat n [ commands ]
- if condition then command
- output
- print
- request
- make

- readword

Given the basic commands provided, the ability to extend the set of recognizable commands with user defined procedures has also been provided.

to square

```
repeat 4 [ forward 45 right 90 ]
```

end

Procedure invocation would be accomplished by entering the command 'square'.

Input parameters can be specified

to square : size

```
repeat 4 [ forward:size right 90 ]
```

end

For this example, procedure invocation would be akin to:

'square 100'.

Output parameters can be specified:

to sum :a :b

output :a + :b

end

A syntactically correct statement that invokes the sum is:

```
sum 4 5
```

Logo is a dialect of LISP, and therefore shares some of the list processing features of LISP. Given the following list: [a[b[c d]]] the com-



mands on the left would give the results shown on the right.

command	result
first	a
but first	[b[c d]]
fput [y z] [b[c d]]	[[y z] b[c d]]
sentence [y z] [b[c d]]	[y z b[c d]]
make 'x [a[b c]]	x := [a[b c]]
first but first x	b
make 'x : x + 1	x := x + 1

The strong point in favor of Logo is the modularization that can be achieved with the procedural approach to problem solving. This feature, combined with recursion, makes Logo a reasonably powerful programming language.

Abelson [Abel 82b, p.112] writes: "If we can dispel the delusion that learning about computers should be an activity of fiddling with array indexes and worrying about whether x is an integer or a real number, we can begin to focus on programming as a source of ideas. For programming is an activity of describing things."

Criticisms of Logo: It appears that Papert's group has gone full circle. Logo was designed for teaching a subset of programming concepts, which is a fine motivation, but the designers were sidetracked; and, as a result, Logo does not offer strong type checking. I believe that they are wrong. Following several years of research to determine those principles that make a programming language worthwhile the computing community intro-

duced several, rather strongly typed, languages (PL/I, Pascal, Ada, etc). Although, at first blush, Logo appears to be in the same category as Pascal, it is evident that the design philosophy is not consistent with modern principles of software construction.

Mismatching types is an all too common source of errors (both at compile time and at run time) and, therefore, should be detected by the compiler as early as possible. Also the student should realize, as soon as possible, the difference between different data types (such as integer and boolean) in a given programming language. Therefore, the above concepts have been included in the software package described in this dissertation.

Logo allows the creation of a special purpose sub-language not provided in the original definition. The existence of a multitude of widely distributed user sublanguages is detrimental to the advancement of a standard language (and a standard set of teaching tools) in the academic environment. Sometimes, one despairs.

#### 2.3.2.2 Karel the Robot

Karel is an abstract robot that inhabits a flat land (the TV screen) consisting of horizontal streets (running North, East, South and West) and two impenetrable walls running along the X and Y axes of the first quadrant of Karel's world. These two walls retain Karel within the bounds of the first quadrant.

There are five primitive commands which can be used to cause Karel to react: move, turnleft, pick beeper, put beeper, and turnoff.

The five commands are combined in various ways (by the student) to perform many complex operations (not necessarily related to, nor limited to, Karel's simple world). The programming constructs (and concepts) that are introduced to the users (via Karel the Robot) are:

begin...end blocks

if... then

if... then... else

nested if's

iterate n times

while C do ...

define new instruction ... as

libraries

invariants and preconditions

procedures

Not covered are the following concepts which are usually covered in a CS-200 course:

variables

arrays

files

input/output

case statement  
assignment  
type statements  
parameter passing

Criticisms of Karel: There is no 'immediate mode' feature that allows experimentation. An entire syntactically correct program must be keyed in before it can be run. The user is required to have mastered the rules of syntax (especially the rule concerning the semi-colon), the rules concerning keywords and program structure before the first program can be run [Patt 81] [Krau 82].

The concepts not covered by Karel can be covered with a package following the design contained herein. Provision has been made to cover all of the concepts listed above within the design.

The above packages (Logo and Karel) have been well received and the student responses have been positive. Pattis reports that Karel the Robot has been used extensively in classes at Stanford University and at the University of California at Berkley [Patt 81]. Accordingly, when the students at the United States Air Force Academy were required to switch from the interactive package to traditional programming in a Batch mode, they requested that the interactive system be reinstated. [Krau 82]

### 2.3.3 Structured Environments

The structured environments described in these paragraphs, are software packages that are used for teaching, but also they constrain the

user in some ways that Logo and Karel do not. The environments are becoming more structured in a reductionistic fashion. Among other things, they all have a syntax checker. Parenthetically, these newer packages are not CAI systems as described earlier, but, rather, are special language systems.

#### 2.3.3.1 Programming Made Simple

This package consists of a series of programs (i.e. Pascal compiler/simulators) that support larger and larger subsets of Pascal. As the student progresses through the course, advanced compilers are used and more of the features of the language are made available. The advantage of this scheme is that the user will not be exposed to error messages referring to features of the language with which he/she is unfamiliar, a common problem with PL/C and like packages.

#### 2.3.3.2 Cornell Program Synthesizer

One form of CAI developed at Cornell University is the Cornell Program Synthesizer (CPS) [Teit 81]. This package is an integrated system that allows the creation, editing, execution and debugging of programs by the student. The synthesizer is syntax directed: both the editor and the execution module are guided by the syntactic structure of the programming language.

The grammar is embodied in a collection of templates of the statements so that entering typographical errors is impossible. When the user wishes to key in a specific statement, a short code is keyed (like using the .i

for the IF statement) and the statement is then displayed, with proper indentation. The user has to key in only the condition and the single statement.

### 2.3.3.3 MacPascal

MacIntosh Pascal was developed by Think Technologies for the Apple Corporation. This package has a data entry mode, with a built in syntax checker, and a run mode that has a single step feature that displays the changing variables in a program.

### 2.3.3.4 A Programming Environment vs A Structured Environment

Logo and Karel offer a programming environment to the student. With this environment, the language is available to the student, usually in a batch mode, and errors are revealed at the end of the run. MacPascal, CPS, and PMS offer a more interactive error checking mode, whereby errors are detected immediately (in MacPascal) or not allowed to be entered at all (CPS).

### 2.3.3.5 Conclusion

The above sections describe several packages that exhibit many excellencies in design: structured programming environments, an interactive mode, and an execution mode. Those same ideas have been incorporated into the design of the software package described in Chapter 3 of this dissertation. In addition, the model also provides a syntax checker that will accept certain programming statements, and none others. Keying errors are

detected, and the user informed, as soon as the return button is pressed.

## 2.4 CAI Graphics

The key to understanding and mastering coding (in any language) is one of knowing how to convert the solution to a real world problem into a form that consists of abstract 'primitives' whose meanings can be replicated by a computing machine.

This conversion can be represented by an isomorphic mapping from the real world to an imaginary world of a computer model. A given task is accomplished within the bounds and limits of the computer and then the result is mapped back into the real world. The model usually given in typical computer science courses is the box model. The memory cells of the computer are represented as little boxes, pigeonholes, and as the program is simulated (by the instructor) the contents of the boxes are changed. This model has been used in various forms since the early days of teaching computing and certainly stands the test of time. It helps to see a model, and a picture as an abstract model helps people to understand programming. When graphics facilities were made available in the computing laboratory, some of the same models (with boxes) made their way into the programs that were used to teach computer science. MacPascal, described earlier, shows a run time computer that shows boxes for variables.

### 2.4.1 Graphics as a teaching tool

This section reviews the issue of the use of graphics, and, in particular, using graphics when teaching computer programming concepts.

Glinert lists several advantages that graphical displays have over written text [Glin 84].

- 1) There is usually a small 'translation distance' from image to comprehension.
- 2) Images are usually recalled as single units of information, which is due to the powerful parallel processing capability of the human mind. This results in a high bandwidth for man-machine communication.
- 3) Images (such as international road signs) are more universal than written text, and can span national, as well as cultural, borders.
- 4) The cost of graphics hardware is dropping, dramatically, along with an increase in performance. These factors make visual communication economically feasible.

#### 2.4.2 Categories of Graphics Packages

Several models using graphic aids designed specifically for the purpose of teaching computer science concepts have been developed. There are essentially 3 kinds of graphics models used: the two dimensional boards, the box mode displaying the contents of individual memory locations, and the new graphics models. In Chapter 5, the various packages are compared with the design proposed in this dissertation.

An example of an early graphics system using a two-dimensional display of memory is CARDIAC. A package developed at Bell Labs and distributed by the local phone company primarily to high-schools unable to purchase a computer, CARDIAC (a Cardboard Illustrative Aid to Computation) is a machine



language based cardboard computer designed to illustrate the operations of a computer and to serve as an introduction to machine language/assembler language programming [Educ 69]. CARDIAC has an accumulator, instruction register, 100 memory cells, an input/output system and a repertoire of ten instructions(read, write, load, store, add, subtract, goto, branch to subroutine, etc). Cardiac is a cardboard computer, but it is very graphic, the student can see the machine (and its memory) and get hands on experience. However, CARDIAC suffers from the fact that all operations are done by hand, therefore, program execution becomes quite tedious. Aside from manual execution, CARDIAC serves the student well.

The second category of graphics packages available are systems that display the contents of some of the key variables in a program while the program is in execution. MacPascal is an example of that type of system. The user can display not only the variables, but also the line of code that is currently being executed is highlighted, giving the user a visual display of program operation.

The third category of graphics packages available are systems trying to move into the area of graphic languages. An example of these newer graphics models is PICT.

Glinert and Tanimoto describe an interactive graphical programming environment they call PICT that permits users to learn programming concepts without the burden of learning the syntax of a conventional programming language [Glin 84]. They describe a system that takes advantage of 1) the increased availability of graphics displays on personal computers, and 2) the high bandwidth of information transfer when in a graphic form (as op-

posed to text). Programming in PICT involves selecting and/or composing icons, moving them on a screen, and connecting the icons to demonstrate the flow of control. Additional features include 1) a menu driven editor, a simulator with animation, help messages, a syntax checker and recursion. The programs are in a form similar to a conventional flowchart where different icons in the flowchart represent different conceptual entities (type, var, if, while, etc). In the conclusion, the authors outline a method for extending the PICT system to generate Pascal code for novices, an idea expanded upon in this work.

Gilbert described PICT as follows: "Icons denoting the various operations needed to perform the calculations must be selected and placed in the module; these icons must then be connected by paths to indicate the desired flow of control".

In some respects, PICT is similar in nature to the package described in this paper. Both schemes display a graphic problem and require the student to solve it. But in terms of depth of understanding and learning curve, PICT requires the student to be more advanced in terms of problem solving. The Business module (described in this dissertation) is similar to a 'real world' situation. The software guides the user from that situation to Pascal-like code. On the other hand, PICT abstracts a solution to an external problem, and the user immediately begins manipulating the flowcharting symbols. With the introduction of connecting paths, the situation is ripe for the inclusion of goto's, a coding feature that runs counter to current trends in programming methodology.

### 2.4.3 Conclusion

With the advantages of graphic displays known, the decision was made to use what graphics facilities were available in the presentation of the material for the model described in Chapter 3 of this dissertation.

## 2.5 Reductionism vs Structuralism

This section reviews the concepts of reductionism and structuralism and demonstrates how reductionism can be used by instructors when teaching computer science concepts in a classroom environment.

### 2.5.1 A reductionistic approach

This section reviews the concept of reductionism as manifest in a laboratory environment for teaching computing.

What is reductionism? The reductionistic method is, specifically, the method of dividing and conquering. Within the context of a classroom, it involves breaking a large subject down into small well defined modules, and then studying each module. For example, all (or most) text books in computer science follow a reductionistic format. "Traditionally, ... structure has been used as an analytical concepts to break down sets into their constituent elements, an essentially atomistic exercise." [Lane 70] Textbooks, therefore, are reductionistic. A single concept is presented, then questions are included to ensure mastery of the subject.

Structured programming is reductionistic [Crow 80]. Most modern programming languages (that include Begin-End blocks, If..Then..Else and Case

statements) are modular in nature in that they encourage the division of the programming task into several simple procedures, such that each procedure can be completed, in turn, until the entire problem has been solved. The interlocking relationships between these procedures can then be consciously minimized.

Implementors of software packages that teach computer languages are beginning to use this approach in the transmission of computer science concepts. Pascal Made Simple (see above) is modular. The writers broke the subject matter down into small, easy to digest, pieces each of which is introduced in a laboratory environment with a different interpreter.

#### 2.5.2 Wholism (or Structuralism )

In contrast, there is the concept of wholism. Wholism initially was concerned with the structure of languages, but is used by many as a method of analysis. "As such, its many facets and different uses make it a subject of various interpretations. No simple or single definition applies to it except in very general terms. One could say a structure is a combination and relation of formal elements which reveal their logical coherence within given objects of analysis. ... "Thus, structuralism attempts to uncover the internal relationships which give different languages their form and function." [Ehrm 70]

"The most distinctive feature of the structuralist method is the emphasis it gives to wholes, to totalities. a new importance has been given to the logical priority of the whole over its parts. They insist that the whole and the parts can be properly explained only in terms of the rela-

tions that exist between the parts. The essential quality of the structuralist method...lies in its attempt to study not the elements of a whole, but the complex network of relationships that link and unite those elements. Hence ... we are given ... an account of the relations of myths to one another, and the relations of episodes to wholes." [Lane 70]

The wholistic approach is located at the opposite end of the spectrum from the reductionistic approach. Wholism aids in the understanding of a system but not necessarily in isolation. The wholistic view is appropriate as an overview and can be used to introduce new material only if one can relate the new material to something with which the student is familiar. To use the wholistic method effectively 1) a readily understood concept is introduced and then generalized to cover the entire model, or 2) the overview is explained using terms with which the student is already familiar. In the first method, an example can be given (say in the context of a programming assignment) and then the example can then be used as a basis when covering new concepts.

For the most part, laboratory work in computer science has been wholistic, a problem noted by Donovan [Dono 76]. When this term is used in a programming context, the use implies that a novice programmer has to know many things before he/she can do any one thing.

A typical student's program consists of several programming statements in a subset of some programming language. The student will be using a language where different features relate to one another. If the student uses a production compiler, it is hard for the instructor to restrict the student's environment to just a small subset of the language. As an exam-

ple, in most laboratory environments, the students will get error messages about advanced features of the language, features they do not know anything about. PL/C is particularly bad about this. Therefore, the environment provided in lab work is wholistic in nature; and, as a result, the students have problems debugging their programs. They either follow the (sometimes) misleading corrective statement blindly, or make random changes and resubmit the program.

Many students are unable to reconcile the fact that most computer science texts are reductionistic while the programming labs are structuralistic.

Many of the manuals provided by any given vendor are wholistic. It is a common occurrence that in searching a manual for clues to the operation of a given statement, cross-references are encountered to features of a language that have not been covered in class or are not relevant to the problem at hand. Many concepts are explained in terms of other more advanced ideas, and the student is forced to try to learn everything in order to do one thing.

This dissertation describes a package that provides an instructional laboratory environment that is reductionistic in nature.

## 2.6 Conclusions

After a review of literature, the decision was made to incorporate the following features in the design of the model:

- it will offer graphic displays with icons representing the entities,

- it is to be instructional, i.e. there will be a tutorial mode,
- the interpreter will enforce a structured programming discipline,
- and it will present the material in a reductionistic manner.

## Chapter 3 - The Model

### 3.1 Introduction

This paper describes the design of a model by which numerous programs can be written for the expressed purpose of teaching computer science concepts to students at the introductory level. Flake identifies four components of a simulated game [Flak 75]. These four components are:

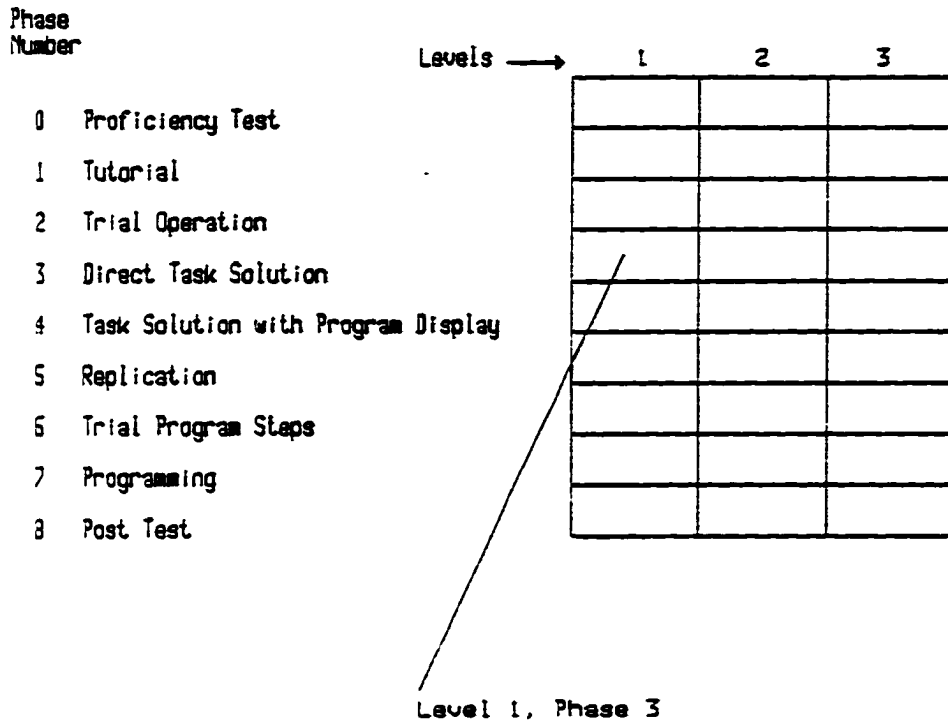
- 1 an abstraction of an environment - this is the model, perhaps consisting of a system of models;
- 2 a series of rules for how the model behaves, or, models interact - this is simulation;
- 3 the freedom for the intern to interact with the simulation to develop strategies - this is the game; and
- 4 'reality' feedback - this is what makes it come 'alive'

These four components have been expanded to produce a nine phase model for teaching Computer Science concepts.

The model is divided into several levels and each level consists of nine phases (zero through eight, see Figure 3.a.), each phase, of which, serves a useful and progressive purpose [Fox 75].

The higher the number of a level the more advanced concepts that is covered. The student is required to successfully complete all nine phases of a given level before progressing onto the next level [Ried 67] [Gibb67],





An overview (of the levels and the phases within each level)  
of the model

Figure 3.a.

where the same nine phases are repeated with more advanced concepts. Each level, then, consists of the same nine phases, with only the content changing from one level to the next.

Within the design, there is a provision (that may or may not be included) which allows the student to skip phases if he/she feels that the material is too easy, or to back up and repeat phases that are difficult to understand. These features allow the student to progress at his or her own rate.

Mozeico lists five stages of development among users of an interactive package [Moze 82]. They are:

- 1) using the system to obtain desired results with minimal prerequisite knowledge
- 2) learning the basics of the graphic system
- 3) progressing to more independent use of the system
- 4) probing into the more subtle or difficult features
- 5) producing quality results within known system constraints

Following Mozeico's guide, the design of the package introduced in this paper is one of guiding the student through progressively more advanced ideas in such a way that each phase is a logical extension of the immediately preceding phases.

All of the packages proposed (the Puzzle module, the Business module,

etc) utilize the idea of multiple levels consisting of nine phases per level.

The nine phases are as follows:

Phase 0) - Review Test

A proficiency review/test is taken within 24 hours after the first exposure to (or if need be, the review of) the concepts presented at the previous level. The student should pass this proficiency exam before he/she can proceed with the current level. This approach ensures a certain level of proficiency with the material and guarantees that progress is being made. If the student fails the review/text, then an earlier phase should be repeated until mastery has been achieved.

Phase 1) Tutorial

The student is tutored in the use of the various keys available (on the keyboard) for use at this level and what each key accomplishes. For an implementation using other equipment (akin to a joystick) then appropriate instructions on the use of the equipment is given.

The purpose of the keys may be unfamiliar to many of the beginning students. For example many introductory level students fail to realize that every line entered must be followed by a line termination character (usually a carriage return). A short tutorial on the use of the keys to be used for successful completion of the assignment is then considered appropriate.

Phase 2) Trial Operations

The student tries out the keys, one at a time, and the keys pressed are immediately executed. This phase has been included for reinforcement. Phase 1 explains the use of the various keys, and Phase 2 allows the student to practice with the keys to gain confidence (with a motor-learning experience) before proceeding with the exercise.

There need be no meaning to the sequence of experiments with the keys. Given the availability of a joystick, the first two phases introduce the joystick and allow the user enough time to become familiar with the physical properties.

#### Phase 3) Direct Task Solution

During this phase, the student accomplishes a simple task using the keys (or Joystick). At this point, a conceptually simple game is introduced along with the keys on the keyboard that can be used to 'play' the game. The games proposed are quite simple in nature, and can be described in a short paragraph of text, i.e. Jig-Saw Puzzle, Grade Point Average, etc. The student is already familiar with a solution to the game and playing the game involves learning the functions of the keys on the keyboard and using these keys to play the game, i.e. accomplish an already familiar task. In this phase, the student accomplishes some predefined and easy to understand task using the concepts covered in Phases 1 and 2 to manipulate the cursor.

#### Phase 4) Task solution with program display

This phase is the previous phase combined with a program display. As

the student manipulates the cursor, a 'program', reflecting the actions, appears at the bottom of the screen. The program is written and displayed by the computer.

At this stage, two events should occur. 1) The student should begin to see the relationship between the actions and the code that is appearing on the screen. Ausebel refers to this technique as 'the incorporation of a new learning task into existing cognitive structure so that a meaningful relationship is established' [Ause 63]. Mayer describes this step in the learning process as crucial for novices [Mayer 79], especially for creative tasks, a category in which Computer Science falls. Winston [Wins 80] argues that this method is referred to as learning by analogy, i.e. when the student generates a constraint description in one domain, given a constraint description in another.

The second event that should occur is 2) the formal syntactic structure of the object language statements are shown to the student. When an individual is involved with learning a new language, he/she forms rules concerning the semantics of the various syntactic constructs (Craik 80]. If the rules the person forms are wrong, then when exceptions to the rules are encountered, either modifications to the rules must be made in order to allow for the exception, or the rule must be discarded. Research evidence supports the hypothesis that it is much harder for a person to relearn something he/she has learned incorrectly the first time [Allen 69]. Therefore, in order to minimize the time it takes to learn programming rules, the proper syntax and proper syntactic and semantic rules should be part of the learning experience, early on.

It is not as if the package was designed to force the student into a particular way of thinking, it's just that the rules of programming in any given language are precise and the sooner the student has exposure to 'correct' syntax, the better.

It is for this reason that proper syntactic structure is introduced in Phase 4.

One of the early decisions was one of determining the introductory programming language to be used. The Kansas State University Computer Science Department has the responsibility of teaching Pascal, Fortran and PL/I to approximately 700 students per semester, and any one of the above languages could have been chosen for inclusion into the software package. Holt, et.al., advises one to resist adopting YATL (Yet Another Teaching Language) for teaching programming concepts [Holt 77]. They argue that the following criteria should be met when selecting a language to be used for pedagogical purposes:

- (1) It should be (part of) a 'real' programming language used in business, science, and government and should be appropriate for introducing programming concepts used in those areas,
- (2) it should encourage systematic problem solving and structured programming,
- (3) it should be a small, convenient, easy to master language, and
- (4) it should be easy to support by economical, diagnostic language processors on a variety of machines including minicomputers.

With the above criteria in mind, there is a dilemma. If Pascal had

been chosen, then the package would be worthless to Fortran and PL/I students. If several packages that would be 'all things for all programmers' were designed, then much duplication of effort would have resulted. The decision was made to teach YATL, and in doing so, three of the four criteria were met.

The difference between Phases 3 and 4 are subtle, but important. Phase 3 displays 'primitive' commands, one at a time, reflecting the actions of the user. Phase 4 combines multiple instances of each primitive command into a single statement.

For example, in the Jig-saw Puzzle module, if the user presses the up-arrow key six times (or presses up on the 'joystick'), at phase 3 the student's actions would be represented by:up up up up up up. At phase 4, when more formal programming concepts are introduced, the students actions would be represented by the following code:

```
REPEAT 6 TIMES MOVE UP.
```

The difference is that six commands have been combined forming one command, but more importantly, the student has been introduced (albeit slowly and carefully) to a more formal and useful syntactic structure.

#### Phase 5) Replication

During this phase, not only is code generated, but the student is required to key in the program statements before he/she can change commands and continue to make progress. The responses are also judged for accuracy,

and an indication of a syntactic error is displayed if an incorrect response is given [RIED 67].

#### Phase 6) Immediate execution

The student keys in program statements which are executed immediately. This step is similar to step 2), but the idea, in this phase, is to show the student, by example, exactly how to write the code to a problem for which he/she is intuitively familiar. At this point in time, actual programming commands are being replicated by the user, a motor-learning experience not to be under-rated.

At this point in time, more formal programming statements are now being used by the student. The primitive commands (of phase 2) have been abandoned in favor of more sophisticated and formal language constructs.

The basic idea behind levels 4,5 and 6 is one of automatically translating student actions into several lines of code and showing that the code so generated translates back into the original actions. Once the idea that programs control computers has been absorbed, a milestone in learning about computer operation has been reached.

The option is left open, at this point, to allow phases 3,4,5 & 6 to be combined together in 2 or 3 phases (at the discretion of the implementor) as the situation warrants. If an implementor feels that the material/concepts to be covered lend themselves toward a more advanced and accelerated pace, then phase 3, for example, can be omitted. This allows for a more flexible format.



## Phase 7) Programming

Several statements are keyed in and then the word 'run' is keyed to get program execution. Also, each statement is displayed (at the bottom of the screen) before it is actually 'executed', reinforcing the student's understanding of the concepts involved. This type of reinforcement is more stimulating to the student than running a program in 'batch mode' [Reid 67].

Phase seven corresponds to the usual programming mode as exists on many current micro-computers. The advantage inherent within this scheme is that two or more statements can be entered and then 'run' entered, resulting in program execution. The relationships between the statements are displayed and the 'program' concept is reinforced, i.e. several commands can be used, collectively and in sequence, to accomplish a given task.

## Phase 8) Post Test

A few questions are directed to the student to ensure that key programming concepts have been absorbed. Also, a hard copy of the pertinent material is distributed for review at a later date. The nine phases are used to introduce a few (usually less than 5) programming concepts and must be repeated if other concepts are to be mastered by the student. It is to this end that the concepts of several levels of learning is introduced. The package is designed so that a typical student 'passes through' all nine phases of Level 1 (which will result in the mastery of a few identifiable skills), then proceed to level 2, say, a week later. The student then 'passes through' all nine phases of level 2, and proceeds to level 3. This

process is repeated for  $n$  levels, where  $n$  is sufficiently large for most important computer science concepts to be mastered.

Once a student has successfully completed all nine phases of this level and (presumably) mastered two to four programming concepts, he/she is then ready to proceed to a more advanced level. The model allows for several levels of semantic information to be covered utilizing the same game. At the second level, the student ensures his/her preparedness, first, by successfully completing the proficiency test. Once this has been done, the student receives a tutorial on the various keys used at the second level. After that, this level is essentially the same as Level 1 except for the different programming concepts covered. Extrapolating, there can be as many levels (semantic units) as is deemed necessary by an implementor to cover all important programming concepts.

#### Suggested Schedule of Levels

Within the normal sequence of events, the student progresses from one phase to the next and from one level to the next as shown in Figure 3.b. However, provision has been made for the student to backup a phase (or a level, if necessary) if review of previously covered material is found to be necessary. This backing up is at the discretion of the student.

The author recommends that the student complete approximately one (semantic) level of material per week (usually at one sitting).

### 3.2 Modules Built According to the Model

Sequence of events for proceeding (or digressing)  
from one level to another level

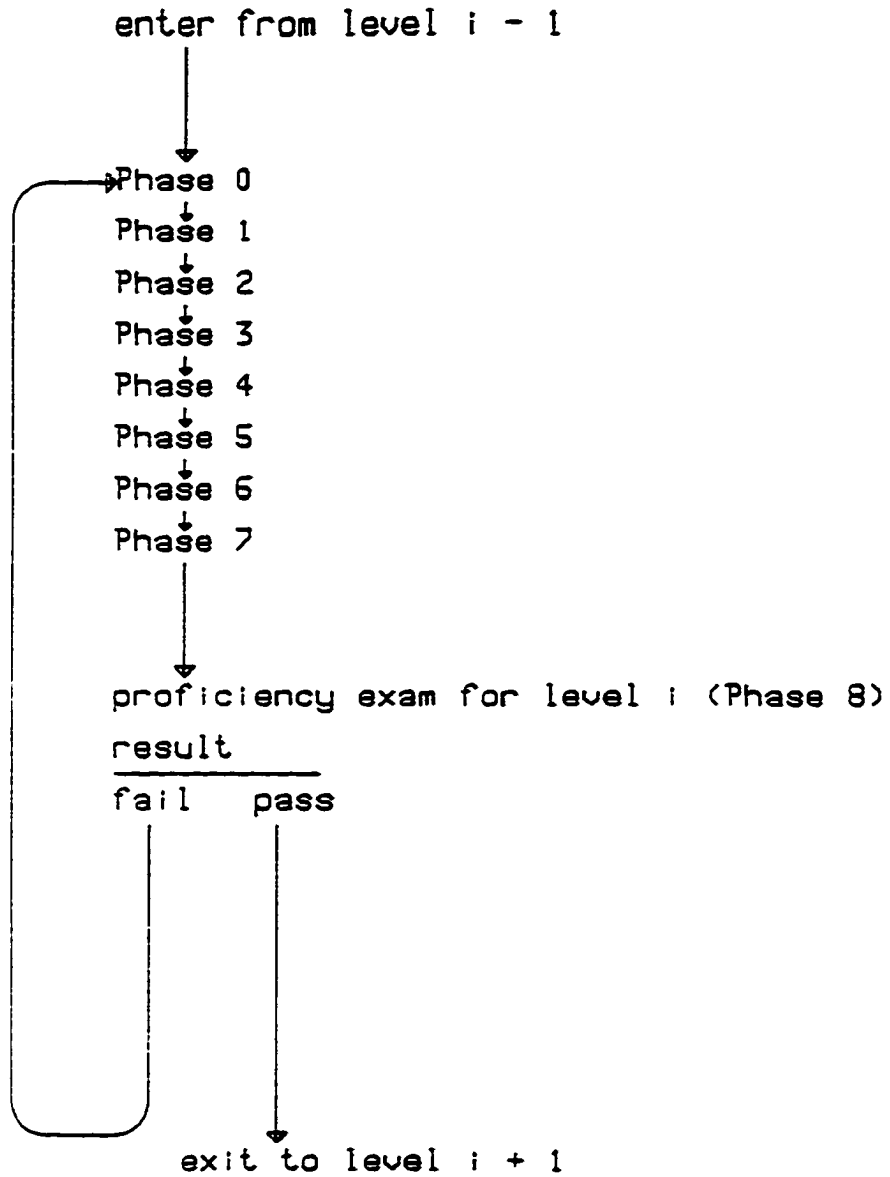


Figure 3.b.

The following is a description of three modules implemented using the design described in the previous pages. The three modules are: A Jig-Saw Puzzle Module, a Grade Point Average module and a Business Problem (Files and Records).

The Puzzle module was implemented on an ATARI micro-computer. It was used by an instructor for CS-200 in a classroom environment and evaluated. The Business module was fully implemented but was not tested in a classroom environment not evaluated with respect to effectiveness. The design for the Grade Point Average module appears in Chapter 3, however the module was neither implemented nor tested in a classroom environment.

### 3.2.1 The Puzzle Module

The Jig-Saw Puzzle module is the first example of a game designed to teach computer science concepts following the model. This section gives a detailed description of the several levels (and the nine phases within each level) used to convey the ideas selected.

The implementation of this (one) module consists of several BASIC programs, with graphic facilities, that allow the user to manipulate (i.e. move up, down, left and right) the pieces of a jig-saw like puzzle. The manipulations are translated (by the program) into an algorithm which is then displayed. A display of a more formal representation of the student's actions convey algorithmic concepts and also provide an example for emula-

tion by the student. The Puzzle Module has the following characteristics :

- It is conceptually simple
- It has immediate reinforcement with feedback
- it is self-learning
- it enables the student to progress from the known to the unknown
- it is implemented on a micro-computer (when complete)
- it uses (and conveys) reductionistic techniques
- it differentiates between application semantics and formal computer science vocabulary and concepts
- it has a visual aspect

The Puzzle Module - Level 1 Initially, Phase 0 checks to make sure the student understands how to turn the machine on.

Phase 1 introduces the following keys (and equipment) with a short explanation of their respective meanings (and purposes):

- |             |   |
|-------------|---|
| up arrow    | moves the cursor up 1 row.                |
| down arrow  | moves the cursor down 1 row.              |
| left arrow  | moves the cursor to the left one column.  |
| right arrow | moves the cursor to the right one column. |

hook                    attaches the cursor onto the piece over which it resides. From this point on (and until an Unhook key is pressed) whenever the cursor is moved, the puzzle piece moves with it. If an implementor decides not to have the hook command available, then the cursor can be permanently attached to one of the pieces.

unhook                de-attaches the cursor from a puzzle piece. Moving the cursor does not cause a puzzle piece to be moved.

right turn            causes the piece hooked to the cursor to be rotated 90 degrees to the right. This feature is optional.

combine                If the piece attached to the cursor is (everywhere along one edge) adjacent to another piece (i.e. if they fit) then combine the two puzzle pieces into one piece by erasing the line that signifies their edge of adjacency. From this point on, the two pieces are considered (and are treated) as one piece.

<return>    TURN        None of the other keys are 'executed' until <return> is pressed.

A 'joystick with button' can be attached to the Atari. This joystick is used by the student to manipulate the cursor (up, down, left, right). The button on the joystick is used to combine the pieces together.

When assembling the puzzle, there is a display consisting of:

- 1) the pieces of the puzzle,
- 2) the cursor,

- 3) the code generated at phases 3 through 8, and
- 4) the status of the Hook switch.

These items are placed on the screen in such a manner as to be aesthetically pleasing, and logistically workable.

The Hook command.

If the cursor is 'over' a puzzle piece then the Hook command reverses the 'hook switch' located in the upper left corner of the screen. (I.e. if the switch is off then it is turned on, if it is on then it is turned off.) If the switch is 'on' then subsequent manipulation of the cursor results in a piece of the puzzle (the piece under the cursor) moving also. If the switch is off then moving the cursor causes no other action to occur. Initially the 'hook switch' is turned off.

Phase 2

At the beginning of this phase, the student is shown (on the screen) four puzzle pieces to manipulate. This phase has been included to allow the student to 'play' with the keys (described in Phase 1) and to ensure that, indeed, they work as described.

Phase 3

At the beginning of this phase, the student is shown several (4) disjointed puzzle pieces and is directed to put the puzzle together using the keys and equipment described in Phase 1. As the student manipulates the cursor, commands reflecting his/her actions appear (across the bottom of

the screen). This is the first appearance of an algorithm that reflects the students actions.

#### Phase 4

At the beginning of this phase, the student is, again, shown several puzzle pieces and instructed to put the puzzle together (in a similar fashion as before) in as few steps as possible. However, in terms of learning programming syntax, this phase is crucial. Several like commands are combined into one looping statement, such as: N UP commands are displayed at the bottom of the screen as -

```
LOOP N TIMES MOVE UP
```

furnishing a precise, yet logically equivalent command. This command is to be one of THE commands that correspond to one of THE programming constructs designated to be conveyed at this semantic level. The new result is that an important programming concept (in this case looping) has been introduced using ideas and terminology with which the user is already familiar. This practice of going from what is known to what is unknown is be used extensively throughout this design effort.

#### Phase 5

Phase 5 used code generation from Phase 4 plus replication (copying). If the user keys the UP key 4 times then the following command would appear at the bottom of the screen:

```
LOOP 4 TIMES MOVE UP.
```



If the student pressed the up key 1 more time then the 4 would be changed to a 5, but if the student pressed any other command key (say left arrow) then he/she would be instructed to replicate the command :  
LOOP 4 TIMES DO UP within the area on the screen designated as the program area. In other words, the student keys in a program that solves the puzzle with a visual display showing how to do it (what to key in).

#### Phase 6

Once the student has had some exposure to statements akin to:

```
LOOP 4 TIMES MOVE UP
```

then he/she is allowed this phase to reinforce the syntax/semantic meaning of this command and like commands with a motor-learning experience. Phase 6 gives the student a chance to review these new formal programming constructs and to 'play with' the various statements in an immediate execution mode. The student keys in one statement then presses <return>. The statement is immediately executed, providing immediate reinforcement.

#### Phase 7

Of all the phases, this one may be the most difficult for the student. The same puzzle as given in Phase 6 is displayed by the computer. At the beginning of this phase, the student is directed to key in ALL of the commands necessary to solve the puzzle. After ALL of the commands have been entered he then types in 'RUN' and watches the result of his intellectual effort appear on the screen. Each line, as it is being executed, is highlighted to enable the student to relate the actions on the screen with

the commands in the program.

#### Phase 8

This phase consists of a short quiz on the key programming concepts covered at this level, i.e. Sequence (one statement at a time, in order, top to bottom) and Looping. The student is then given a hard copy of the primitives (up, down, etc) with a list of formal programming constructs covered, differentiating the two. The student is told that the 'primitives' being used will eventually be discarded, and that he/she should not consider these primitive operations as key concepts within the broader area of computer science.

#### Other levels

Different programming concepts can be covered at the different levels. The following list shows the programming concepts covered at each level using the Puzzle Module.

Level	Concepts
1	sequence, looping
2	variables(numeric and boolean), if then else, assignment
3	decision (if then)
4	input/output
5	case statement
6	arrays (tables)
7	files and record structures

The Puzzle Module - Level 1

## Phase 0

At this level, the concept of the 'hook' variable (that appears as a box in the corner of the screen) is introduced. At the previous level (level 1) the cursor is permanently hooked to an arbitrary piece and the student is unable to unhook the cursor from that piece. At the current level, if the hook option is chosen, it is possible to move the cursor around without actually moving a puzzle piece. When the cursor is 'hooked' onto a puzzle piece a 'yes' appears in the slot marked hook (on the screen), else a 'no' appears. The concepts covered, at this level, involve the following:

- slots exist that can contain a value (in this case a 'yes' or a 'no')
- these slots are called variables
- the student has direct control over the contents of the slots (i.e. he can hook or unhook the cursor to a piece at will)
- the contents of the slot has an effect on the actions appearing on the screen (i.e. the program)

## Phase 1

s.p

During the phase, the user is allowed to operate the joystick and see the results appear on the screen. Any operations to 'hook' onto a piece will result in the variable hook displaying a value of on.

## Phase 2

The student assembles the puzzle, using the newly defined 'hook' and

'unhook' commands along with the previously defined 'combine' command. The contents of the hook variable appears in the top left corner of the screen.

### Phase 3

In this phase, the student assembles the puzzle as before. The commands that appear at the bottom are:

- H for hook,
- U for unhook,
- C for combine.

### Level 4

The 'high level' commands that appear at the bottom of the screen are the following:

- H hook cursor
- U unhook cursor
- C combine with the adjacent piece

in addition to the statements covered at the previous level. Also appearing within the code are the following statements: HOOK = YES when a hook command has been specified, HOOK = NO when an unhook command has been specified.

### Phase 5

The student is required to key in the new command (plus all of the old commands) that appears at the bottom of the screen.

### Phase 6

This phase has been provided for a practice phase. The student is

directed to 'try out' the new statements covered at this level (plus any statements covered at previous levels).

#### Phase 7

The student is directed to attempt to solve the puzzle using all of the 'high level' commands covered by this level and previous levels.

#### Phase 8

A short quiz on hook and unhook can be given for reinforcement and to ensure student confidence.

#### Level 3

The student is informed that in order to put the puzzle together he/she must rotate one or more pieces clockwise for the proper sides to match. Except for this one change, the phases of Level 3 correspond to like phases of the previous levels.

There is a lot missing. It is difficult for one module, of this nature, to convey every programming concept covered in an introductory programming course. The module does not lend itself, naturally, to teaching every programming concept, therefore a variety of games will have to be implemented, in order to cover the entire spectrum of ideas. Therefore, two other modules, inherently different from the Puzzle module, are also described in this dissertation.

The Puzzle Module is written in Basic for an ATARI-800 micro-computer in BASIC-A. The minimum configuration is 16K RAM, one 5 1/4 inch floppy

disk drive, a color monitor and a 'joystick'. The package is divided into several programs that were 'chained' together to form a complete module.

The following list gives the approximate number of lines of BASIC code necessary to implement the Puzzle Module on the ATARI-800. The code includes some necessary internal documentation, which makes up about 3 percent of the text.

tutorial	tut1.lst	250 lines
practice	pl1.lst	460 lines
generate code	gen1.lst	300 lines
replicate code	rep1.lst	340 lines
immediate mode	pra1.lst	275 lines
run mode	run1.lst	370 lines
review concepts	rev1.lst	47 lines

Initially, the movement of the icons on the screen was too fast for a novice, and delay statements had to be inserted into the code so the users would not run the icons off of the screen. In the later phases of the modules, necessary calculations in the code provided the delay.

Production of instructional packages are, by their very nature, time consuming. Unfortunately, there are no 'instructional meta packages' available that provide a template that will allow an implementor to enter example data and automatically create a learning module. All of the modules had to be written by hand, a task hindered by the fact that BASIC-A is not structured. Many hours were consumed debugging a relatively simple program

Scheduling for the only available ATARI-800 was a problem (although minor). The machine was made available for program development thrice weekly for an hour or two. With this constraint, implementation of the Puzzle Module took three to four months for completion.

There were language dependent features that made a difference in both execution time and program development time. For example, the BASIC-A interpreter on the ATARI-800 searches for line numbers in a linear fashion starting at the top. Therefore, a given program runs faster if program documentation (appearing in REM statements) is placed at the bottom of the program rather than at the top. The graphics functions used were completely machine dependent because there are no standard graphics functions available on micro-computers.

### 3.2.2 An Overview of the GPA module

The Grade Point Average Module involves the use of the input command. This command allows the user to key in information from the keyboard while the program is running. The various commands (and keys) available with this program appear below along with an explanation of their function.

Input Hours	these commands allow the user to key in a number and a
Input Grade	letter separated by some delimiter
?	A ? in the TGP column displays the total grade point
	accumulated. A ? in the CCPA column displays the
	cumulative GPA.

The programming concepts covered with this module include the following: Input, Output, Sequence, Variables, Case, Tables, Assignment, and 2-D arrays.

The screen layout for the GPA problem is very much like the following:

	hours	grade	total	total	cumulative
			hours	grade	GPA
row 1	3	A	3	12	4.00
row 2	4	B	7	24	3.42
row 3	2	C	9	28	3.11

### 3.2.3 The GPA module

#### The GPA module - Level 1

##### Phase 0

During this phase, the student is instructed in the use of the INPUT and the ? instructions.

##### Phase 2

During this phase, the user is given the opportunity to test the functions described in the previous phase. The concept of moving data from a keyboard 'into' the computer is an important one, and this phase gives the student the opportunity of reinforcing a particularly important idea.

##### Phase 3



The user is directed to key in a set of 'hours/grade' pairs for GPA computation. Within this level an phase, each column within the table is given an explicit name.

#### Phase 4

As the student keys in pairs of 'hours/grade', (and a ? in the appropriate columns) the program appears with the array elements appearing as: Row j of column-name.

#### Phase 5

During this phase the student replicates the generated program.

#### Phase 6

During this phase, the 'high-level' statements keyed in by the student are immediately executed.

#### Phase 7

The student is required to key in an entire program and run it, for the completion of this phase.

#### Phase 8

The review quiz covers the major concepts listed in a previous paragraph: assignment, 2-D arrays, & input.

### The GPA module - Level 2

The second level of the GPA Module follows the form and content of the

first level with the following exceptions.

Phase 1

The student is informed that columns can be referred to by number (rather than by name, as was the case in the previous level).

Phase 4

The notation for referencing elements of a table is changed to: Row  $j$  of column  $k$ .

#### The GPA Module - Level 3

The third level of the GPA module follows the form and content of the second level with the following exceptions.

Phase 1

The student is informed that the entire table can be given a name and elements within the table are (in this level) referred to by: Table (row  $j$ , column  $k$ )

Phase 4

The code generated by the current phase displays array element references using the following scheme: Table(row  $j$ , column  $k$ ).

#### The GPA Module - Level 4

Phase 0

During this phase, the following convention is adopted: the first

number, within the parenthesis of a Table reference, refers to the the row and the second number refers to the column.

#### Phase 4

References to table elements have the notation: Table(i, j).

#### 3.2.4 The Business Module

The following section of this dissertation outlines a third game which is based on the model described earlier. The motivation for the approach used is the following: a newcomer in a business office (the user) is directed to hand carry a document to a data base manager and then (if also directed) to an inventory manager. During this process, the user acquires an understanding of the methods and mechanisms used to keep accurate records of items in the inventory. The game was implemented for the following reasons: 1) it is an abstraction of a procedure performed by an office clerk, i.e. moving papers from one desk to another. Then, as the papers are moved, the user views the actions taking place at each station; 2) the problem is relatively simple, allowing the user to concentrate on what has to be done rather than having to be concerned with the (sometimes) myriad of details of a more complex system; 3) the inventor management problem lends itself to the understanding of RECORDS and the 'dot notation', an abstraction found in many of the new programming languages (Pascal, Ada, PL/I, etc); 4) at one level, the user 'sees' the actions of the icon representing the messenger, and at another level the user sees the actions of the data base manager. This approach allows the introduction of main programs and procedures, reflecting the two views previously intro-

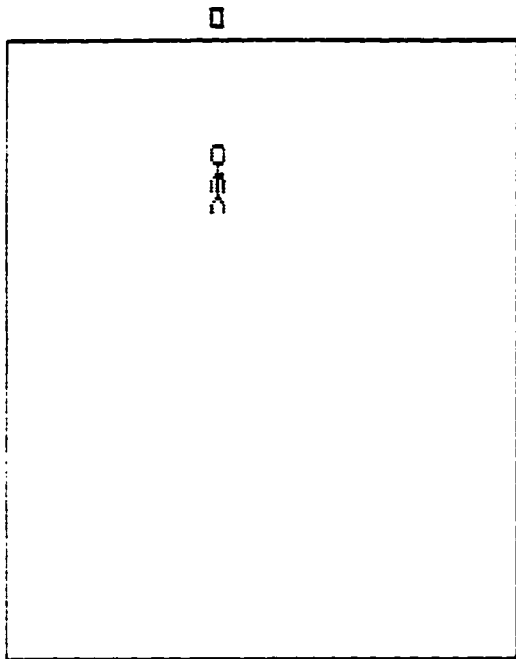
duced; 5) although not specifically designed with the package, the concepts of a structure (a record having several unlike fields) will be conveyed, albeit as a by-product. The presumption is that the user will have been exposed to records previous to exposure to this level of instruction; and 6) the separate notions of office procedures and selection from a menu list are intuitive in nature and can serve as a springboard for introducing new concepts.

The user must have successfully completed the sequence of modules preceding the current one, and have a basic understanding of the concepts that were covered. Specifically, the concept of a variable (that can hold a value) should, at this point, be understood along with the concept of a field within a record.

The business game displays several icons (as described below). The purpose of the 'game' is to move documents (which are also icons) from one icon position to another or to select a task to be performed using a menu selection option. The icons are:

- a document carrier (which represents a person)
- a data base manager (which represents a person)
- an inventory supervisor (which represents a person)
- a mail box

At each step in the process, goals will be provided at the bottom of the screen. Initially, the screen will display a carrier icon and the directions to move the icon to the top of the screen (see Figure 3.c.). After retrieving a purchase order, the user will be directed to move the



Move the icon to the top of the box  
to fetch and display the order

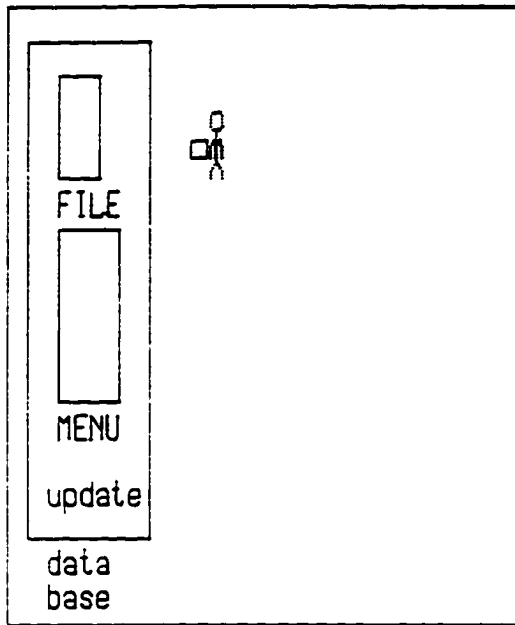
Figure 3.c.

purchase order icon (representing a courier carrying a document) from the point of entry to the icon representing the data base manager (see Figure 3.d.). The contents of the purchase order document will be similar to the following:

item	rectangle
amount requested	3
result	—

Initially, the purchase order will be displayed at full size, but when the user moves the icon across the border, the purchase order will diminish in size (only the envelope will be shown) so as not to cause logistic problems during the remaining sequence of events. The contents of the purchase will be displayed on the screen and will remain there, continually, for user reference.

During the period of time that the icon is near the database, the database is updated reflecting the directives on the document. If the number of items originally requested is less than or equal to the number of items recorded in inventory, then the data base manager will stamp the purchase order with an "ok" and return it to the courier. The user will then be directed to move the purchase order to the inventory manager icon. Once this has been done, several items (as reflected on the original request) are removed from inventory (they disappear out of the bottom of the



```
item:    rectangle  
amt requested: 3  
result:   reorder  
purchase order (p.o.)
```

move the icon down and to the left  
to submit the invoice to the data base

Figure 3.d.

screen). If the number of items originally requested is greater than the number of items recorded in inventory then the data base manager will issue a Reorder Form and give it to the courier (see Figure 3.e.). No trip to the inventory manager will be necessary; however, the user is requested to move the order request icon to the mail box icon (see Figure 3.f.).

Although, the concepts of procedure invocation and procedure definition are introduced across two levels of instruction, in order to describe the overall picture, the description of both levels has been included, intermixed, in the following discussion.

Various phases -

#### Phase 1 - tutorial

A mouse (a small, handheld, boxlike attachment, containing a wheel or like device and one or two buttons) can be attached to the micro-computer to manipulate the icons. If a mouse is not available directional keys can be used. The game has been designed around the mouse (or the directional keys); hence, a short tutorial on the use of the mouse (or the directional keys) is provided in this phase. At the second level, the user accesses a menu driven routine by pressing function keys therefore the user will be given a tutorial on the use of the function keys during phase 1.

#### Phase 2 - Trial Operations

During Phase 2, the student will practice with the mouse (and the



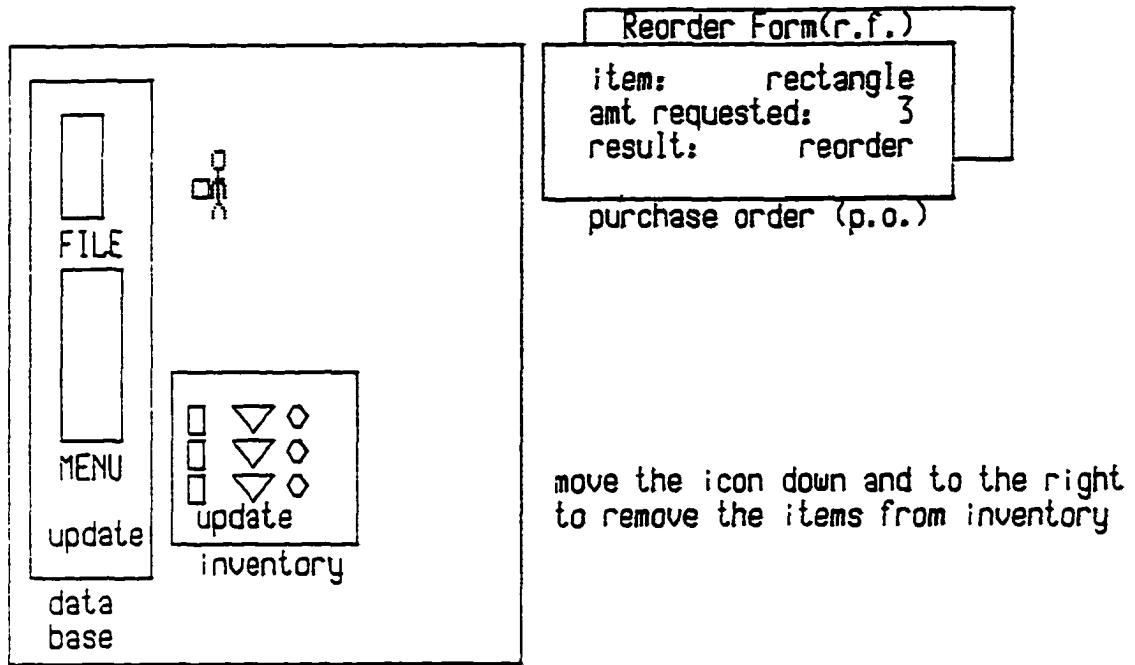
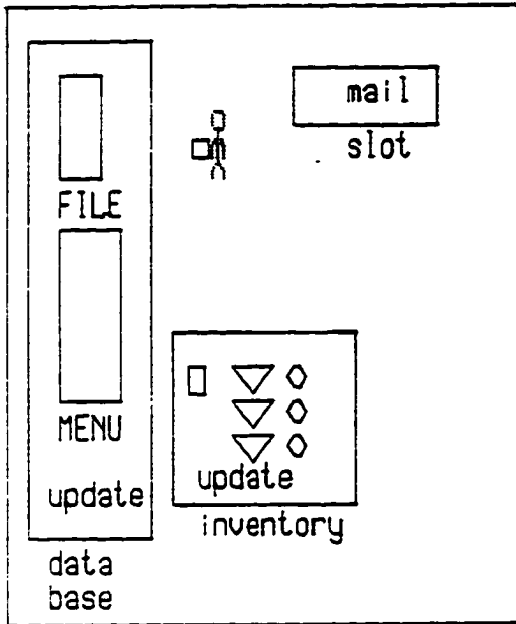


Figure 3.e.



item: rectangle  
 amt reordered: 5

reorder form (r.f.)

move the icon up and to the right to  
 mail the reorder form

Figure 3.f.

function keys during the second level of instruction) in the course of solving some simple problem. The student is directed to move icons around the screen (using the mouse) in the following manner. The student will move the icon to a particular area on the screen and, in the process, will accomplish a given task.

### Phase 3 - Direct Task Solution

In phase 3, the user is given the opportunity to view the complete picture of all of the events on the screen.

The particular 'business' problem is introduced within this phase (see above). The reader is reminded that the game is divided into two levels. At the first level, the purchase order icon is moved about the screen by the user. If the amount requested is in inventory then the purchase order will be stamped "ok" and will be delivered to the inventory manager. If the amount requested is greater than the amount in inventory, then the purchase will be stamped "reorder" thereby converting the purchase order to an order request, and will be delivered to the mail box.

At the first level of instruction the contents of the data base is displayed in a form similar in nature to the following:

<u>item</u>	<u>number</u>
rectangle	5
triangle	7
hexagon	3

The user is reminded that these numbers reflect the inventory, but are not the actual inventory items.

Within this phase of the second level, the student views the problem from a different perspective, i.e. from the perspective of the data base manager rather than from the point of view of the caller. (see Figure 3.g.). The concept of procedures is introduced in this manner. In the first level, the user will see, essentially, the procedure invocation (i.e. from the outside-in). In the second level, the user will see the procedure definition (i.e. from the inside-out), although no code will be generated. Also in the second level, a menu of options that allow the user to perform selected operations (get a record, update a field, release the record) will appear, and the student will be required to select the correct option (by moving the cursor from the current location to a point near the individual items within the menu and pressing the button on the mouse or pressing return). The menu will appear as follows:

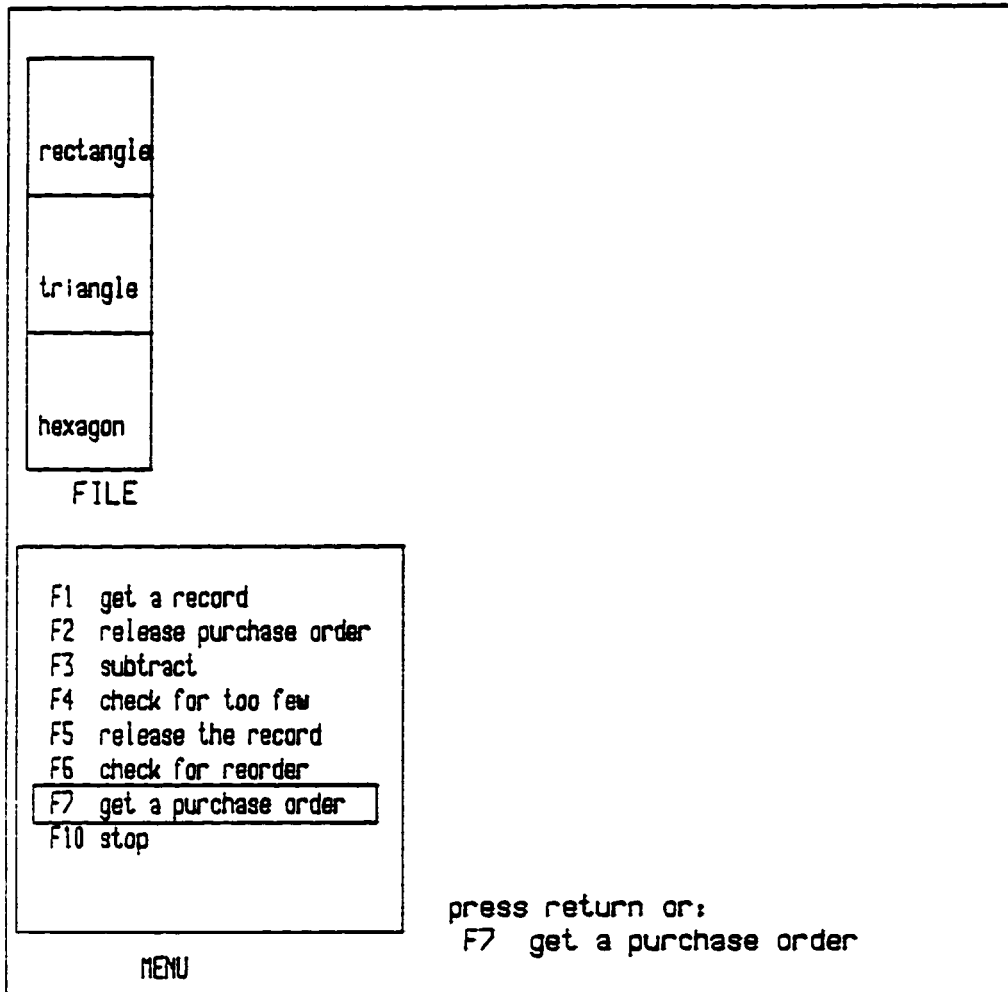


Figure 3.g.

F1	get a record
F2	release purchase order
F3	subtract
F4	check for too few
F5	release the record
F6	check for reorder
F7	get a purchase order
F10	stop

In the second level, and after option F1 is selected, the user will see the purchase order. (see Figure 3.h.). Also, in the second level, the user will begin to see the actual record used within the data base to store the information concerning the inventory. (see Figure 3.i.). This record will be represented in the form of a structure not unlike structures in PL/I or Pascal, and will be similar to the following:

item	rectangle
count	3
reorder	5

#### Phase 4 - Task Solution with Program Display

Within this phase, as the student moves the icons from place to place, representative code reflecting the actions performed will be generated and displayed at the bottom of the screen. Since code will be generated, after the second phase, the user is required to press the RETURN button on the

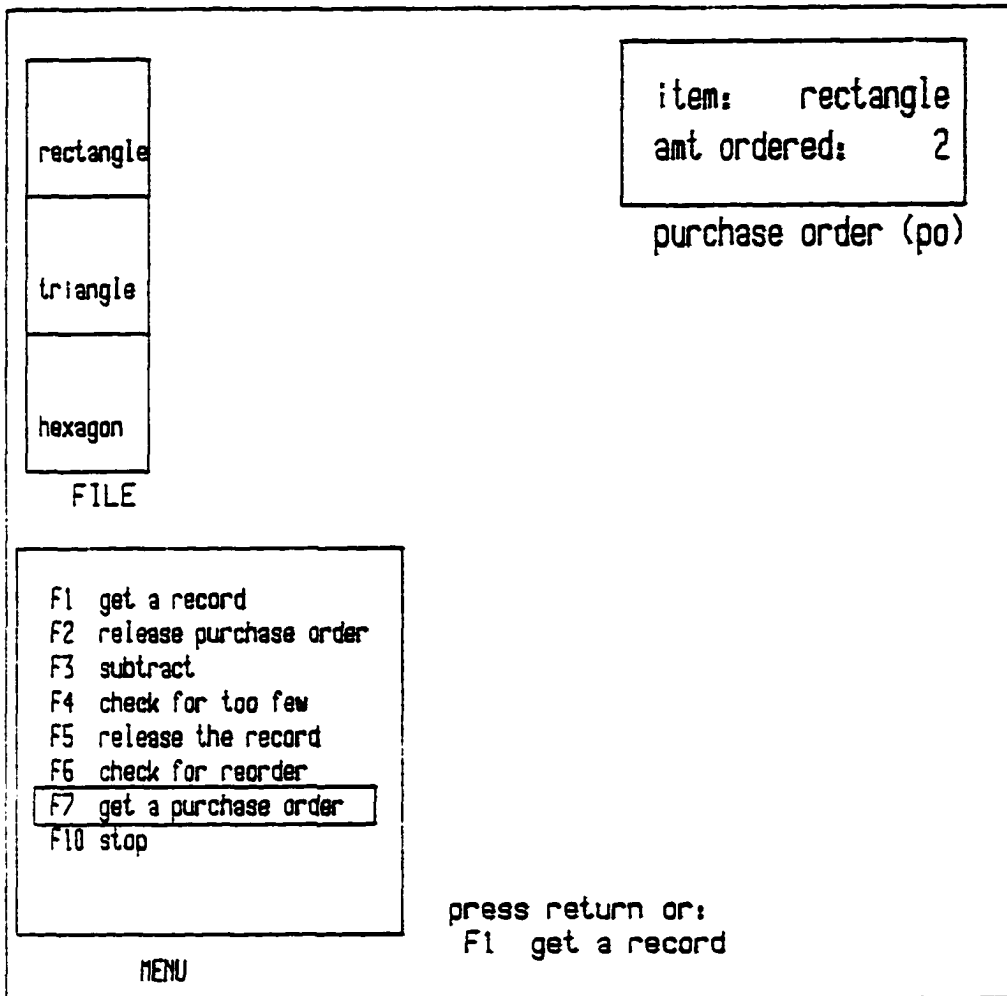


Figure 3.h.

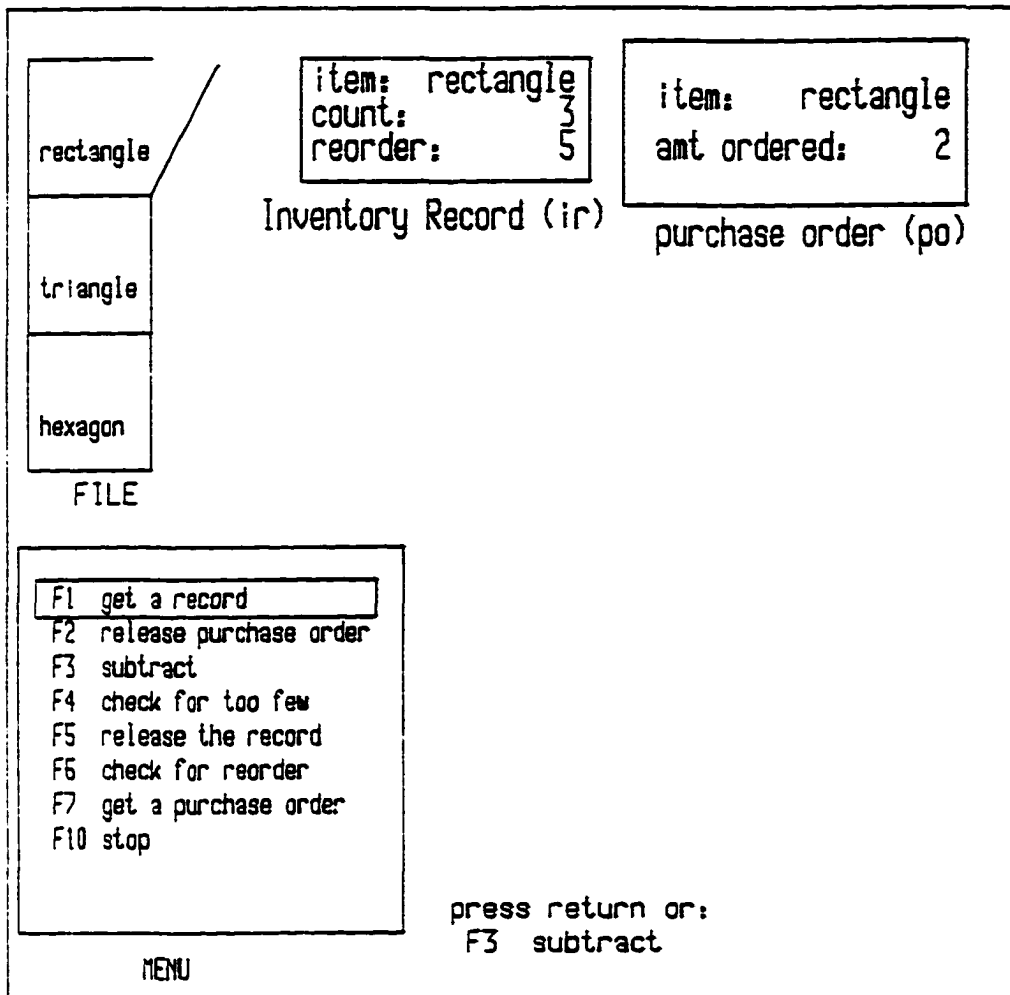


Figure 3.i.



console. This action is consistent with current operating methodologies which dictate that every line entered is to be terminated with a carriage return.

The software generates one or more instructions that change individual fields of a record (in this case, the COUNT field of the RECTANGLE record is changed).

```
USING RECTANGLE DO COUNT = COUNT - 2
```

As the code shown above is generated, the data base record is updated, as shown in Figure 3.j.

When the user selects the F6 option indicating that he/she wants to check the inventory to see if it is time to reorder, and if the number of items ordered exceeds the number of items in inventory then a reorder form is generated, as shown in Figure 3.k. A typical sequence of commands generated is as follows:

```
MOVE ICON TO DATA BASE MANAGER  
INVOKE PROCEDURE UPDATE_DATA_BASE (IN PURCHASE ORDER, OUT RESULT)  
IF RESULT = OK THEN MOVE ICON TO INVENTORY MANAGER  
IF RESULT = ORDER_REQUEST THEN MOVE ICON TO MAIL BOX  
IF RESULT = ERROR THEN MOVE ICON TO ENTRY
```

At the third level, typical commands generated are the following:

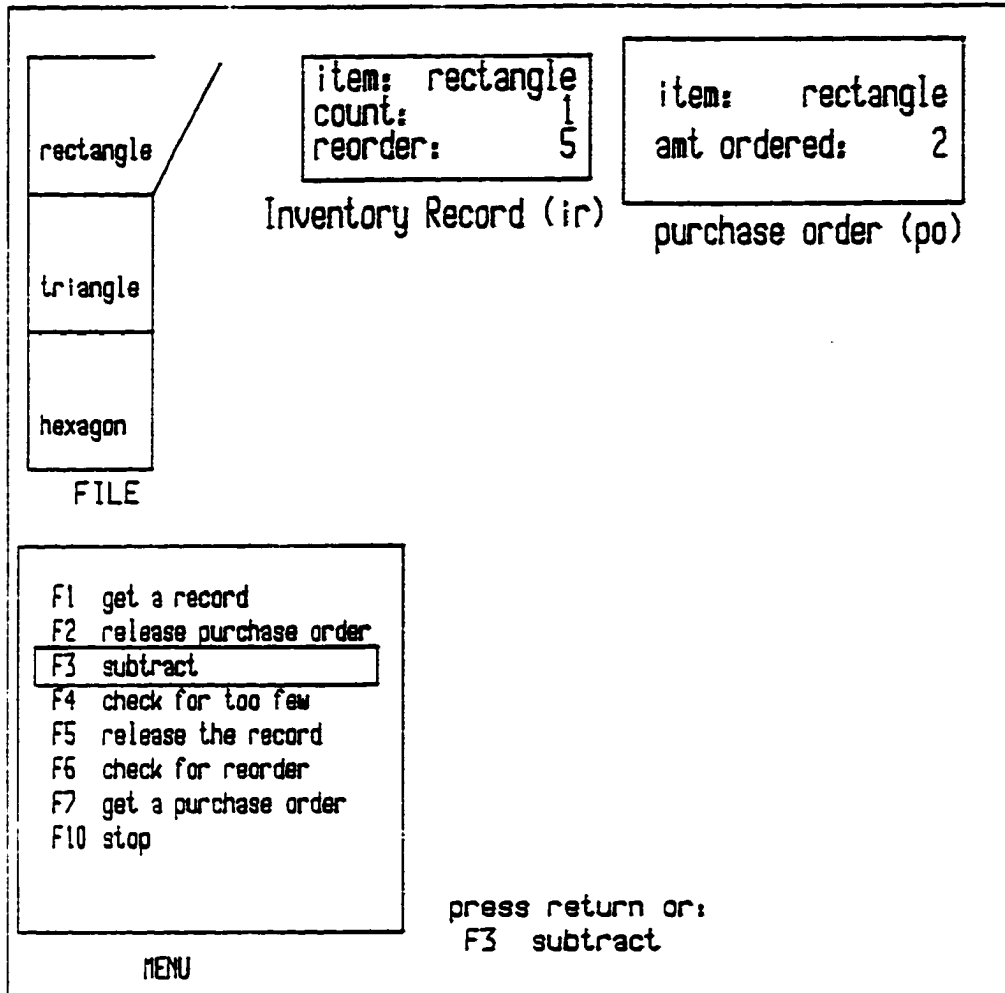


Figure 3.j.

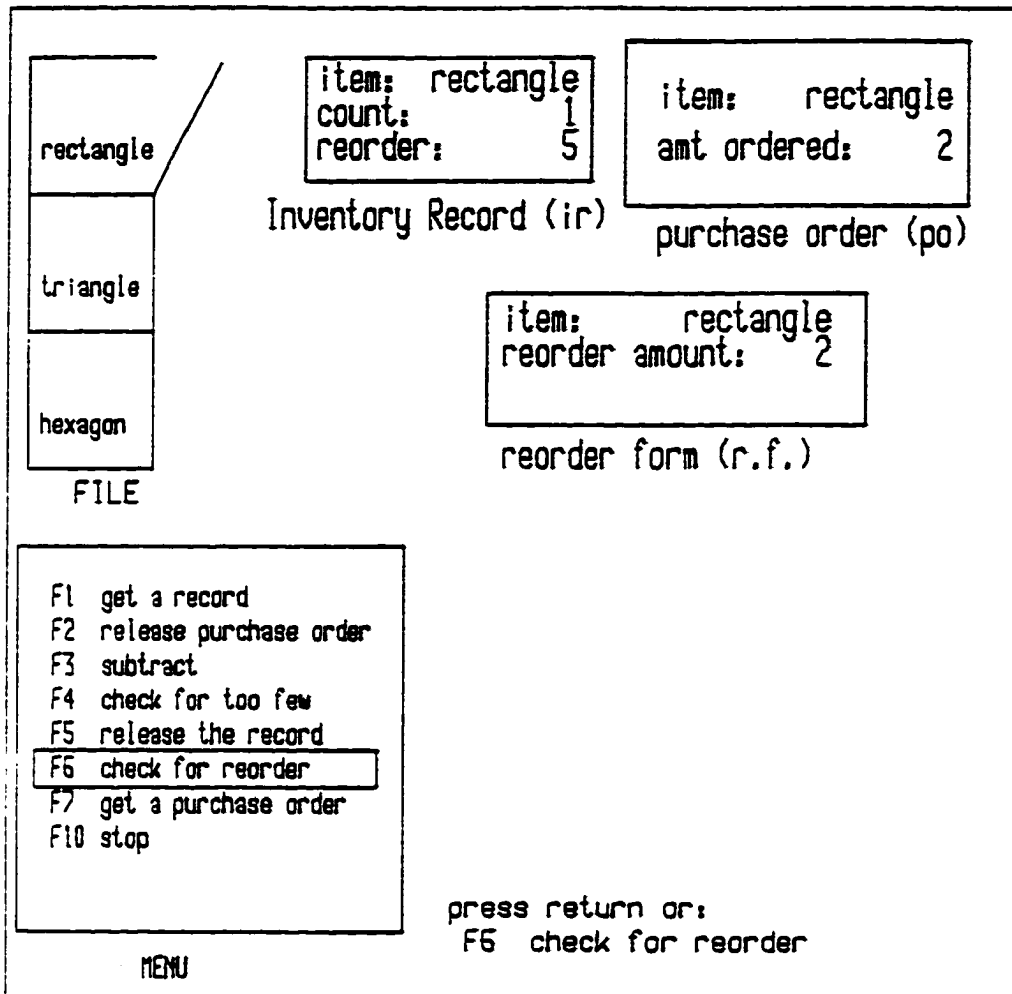


Figure 3.k.

```

PROCEDURE UPDATE_DATA_BASE (IN PURCHASE-ORDER, OUT RESULT)
BEGIN
    GET A REQUEST
    GET RECORD FOR TRIANGLE
    IF PURCHASE-ORDER.AMOUNT > INVENTORY.AMOUNT THEN RESULT = ERROR
    INVENTORY.AMOUNT = INVENTORY.AMOUNT - PURCHASE-ORDER.AMOUNT
    IF INVENTORY.AMOUNT <= REORDER_LEVEL THEN STAMP REORDER
END PROCEDURE UPDATE_DATA_BASE

```

#### Phase 5 - Replication

The student is requested to solve basically the same problem as described in the previous phase, with one difference. Code reflecting the student's action is generated, but within this phase, the student is required to key in the code displayed before progressing.

#### Phase 6 - Immediate Execution

In this phase and the following phases, the mouse is not used. The high-level commands to update the various fields are entered and are "executed" immediately, thereby reinforcing the users knowledge of record update commands. A template of the form:

```
USING ITEM DO STATEMENT
```

appears at the bottom of the screen as in:

USING RECTANGLE DO AMOUNT = AMOUNT - 1.

Syntax errors are detected and an indication of the error is displayed. The user is then required to re-enter any invalid statement.

#### Phase 7 - Programming

The entire program sequence is then entered at this time. A "partial program" can be run by keying in the keyword "RUN", and the sequence of commands is displayed, at the bottom of the screen. The icons will be moved as per the users instructions.

#### Details of Implementation

The Business Game is implemented on a Zenith micro-computer running DOS with 16k of RAM, at least one 5 1/4 inch floppy disk drive, and a color or monochrome monitor. The program utilizes the four directional keys and the 10 function keys available on the keyboard. The package is divided into two programs that form a complete module. This module covers the topics of procedure invocation and procedure definition. The following list gives the approximate number of lines of Pascal code required to implement the Business Module on the Zenith.

level 1     1850 lines

level 2     1631 lines

The package is written in its entirety in Turbo/Pascal, an economical Pascal Editor and Compiler distributed by Borland International. Among other features, Turbo offers a convenient debugging tool usually not available with other Pascal Compilers. If an error occurs in compilation, the editor

is loaded and the line number of the error detected by the Turbo Compiler is transferred to the Turbo Editor, and the users program is displayed on the screen with the cursor on the line containing the error. With the facilities available, syntax errors can then be systematically located and removed.

Pascal, as an implementation language, contains many excellencies in design over and above those features offered by other programming languages provided with micro-computers (such as BASIC). The program can be divided up into separate procedures, each procedure can be developed and then incorporated into the main program. This procedure was followed during the implementation phase of the Business module.

There are language dependent features appearing in the code. The graphic functions (Plot and Draw) are Turbo/Pascal dependent. As a result, these graphics commands were the source of most of the programming problems that occurred during the development phase of the project.

The hardware offers the following graphics modes:

character mode      25 lines of 80 characters per line

hi res mode          200 lines of 640 pixels each

In addition to the two graphics modes, there were two other factors that caused problems. One was the fact that characters are written to the screen only on line boundarys. There was no convenient way to place an individual character on a non-line boundary. Another factor which caused problems was that the location of pixels in even-numbered rows (in Hi Res Mode) are 8192 bytes down (in memory) from the corresponding pixels in the

odd-numbered rows. Movement of icons from one place on the screen to another involved calculating the location of vertically adjacent pixels, making sure not to erase a pixel already moved. A byte in memory is used to store the status of eight horizontally adjacent pixels; therefore two horizontally adjacent pixels (on the screen) may be in the same byte or they may not. Calculation, in a more or less elegant manner, of the location and displacement of given pixels turned out to be quite a chore.

There was some discussion, early on, of using two video display screens to display the multitude of icons necessary for the completion of the tasks. For practical reasons, the idea of using two screens was abandoned, and one screen was used throughout the sequence. With this constraint the icons were tightly packed onto the screen, leaving little room for displaying text on line boundaries. This situation forced a decision of aligning characters on non-line boundaries, hence, the problem described above emerged.

Implementation of the Business module took approximately five months, devoting two to three hours a day to the actual programming.

### 3.3 Justification

Mayer writes that "the principle justification of the existence of instruction is that it assists an individual in learning something better than he would have learned it by himself [Mayer 67]. It is the burden of instruction to demonstrate the value of its existence by demonstrating its power to facilitate learning." The author asserts that, with respect to

the justification of the software package described in the previous section, a serial, reductionistic approach to learning is far more efficient than other known methods.

The programmers tool is a set of primitives. These primitives may be anything from machine language instructions to powerful operating systems, but they are primitives, at least in the sense that they are items that are combined together in various combinations to accomplish some purpose. They are also primitives in the sense that a group of building blocks is available and the act of combining them together into a single package provides the power.

The key to understanding and mastering coding (in any language) is one of knowing how to convert the solution to a real world problem into a form that consists of abstract 'primitives' whose meanings can be replicated by a computing machine.

This conversion can be represented by an isomorphic mapping from the real world to an imaginary world of a computer model. A given task is accomplished within the bounds and limits of the computer and then the result is mapped back into the real world. (The notion of accomplishing some task by appearing to do something else is not a new one.)

The mental process required, for the conversion from the real world to the machine and back again, represents a quantum leap for many students of programming.

That quantum leap is the focus of this dissertation.



## Chapter 4 - Evaluation of One Programming Module

### 4.1 Introduction

This chapter presents an evaluation of the puzzle programming module covered in Chapter 3. The organization of this chapter is as follows: Section 4.2 contains a description of the decisions that were made during the course of the experiment and why they were made; Section 4.3 contains the questions, the details of the experiment, and the interpretation of the results.

### 4.2 Experiment Decisions

During the summer term of 1984, the Puzzle module was used to introduce CS-200 students to computer programming concepts. There were two groups involved in the experiment, a BASIC programming class of 44 students (the experimental group) and a PL/I programming class of 38 students (the control group). The control group was given a conventional programming assignment. The experimental group was given the Puzzle module as a substitute for the first programming assignment. Since there was only one micro-computer available at the time of the experiment, a signup sheet was distributed and each student signed up for a 50 minute slot on the machine. The typical time for completion was approximately 20 minutes. Sometimes, a monitor was available to answer questions in case the students had difficulty with the software or the hardware.

To evaluate the Puzzle module, two measures were chosen: attitudes of the students in both classes were determined from the drop rates of both

classes and from questionnaires completed by both groups before and after the experiment took place. The drop rate turned out to be insignificant (5 in one class, 6 in the other), therefore it is not discussed further. The questionnaires contained attitude statements about computer science and CAI. The mix of questions include 18 questions about attitudes of the students toward computer science, 24 questions about student attitudes toward computer aided instruction, one question concerning the student attitudes toward the instructor of the course (in retrospect, more questions in this category would have been appropriate), and six questions indicating the students own perception of his/her ability to learn faster with a CAI package. The students were required to answer the questionnaire before exposure to the Puzzle module, and then again after exposure to the module (10 days later). A statistical analysis (using a t-test on the questions and Paired Test on the students) was performed with the results of the questionnaires to determine if there was a significant change in attitude resulting from the use of the Puzzle module.

There were six analyses performed on the data as described in Section 4.3. A summary of the six tests follow.

T1 A t-test was used on the statistical means of the student responses for each question to determine if the attitudes of the students in both classes at the beginning of the semester were about the same.

T2a A Sign Test with a Paired Sample was performed on the responses of the students to determine whether there was a positive reaction by the students in the Basic class. Particular interest was placed in the change of attitude recorded by each student for each question

within each class.

T2b An analysis of the responses of the students in the Basic class was performed as in 2a, with the following change: the analysis was performed for each question for each student within each class.

T3a A Sign Test with a Paired Sample was performed on the responses of the students to determine whether there was a positive reaction by the students in the FL/I class. As in 2a, the area of interest was the change of attitude recorded by each student for each question within each class.

T3b An analysis of the responses of the students in the FL/I class was performed as in 3a, with the following change: the analysis was performed for each question for each student within each class.

T4 A t-test was used on the means of the student responses for each question to test the attitudes of the students in the Basic class (the experimental group) had a more positive shift than the attitudes of the students in the FL/I class (the control group).

This section contains a description of the decisions that were made during the course of the experiment. Many of the decisions were compromises, due to the timing of the experiment.

(1) Why was just one module evaluated? The experiment does not validate the entire model, but only one module within the model. Extrapolating the time it took to develop just one module, the complete package covering the entire set of computer science concepts would take several years to be completely developed. In order to demonstrate the concept

of the model, the committee decided that two modules should be implemented and one evaluated in a classroom environment.

- (2) Why was the experiment done during the summer? The issue of timing was discussed with the committee in a meeting during the Spring of 1984. It was the considered opinion of the committee that the experiment should be done as early as possible for the following two reasons: 1) the design could be evaluated before additional models were developed, and 2) any adverse impact of the package on the subjects (in case the modules were found unacceptable) would be minimized. The package was not ready for evaluation during the Spring semester of 1984, and the completion of the Puzzle module coincided with the summer months. Certainly, the summer term is not a good time to do experiments. Nevertheless, the consensus of the committee was that an evaluation should proceed before an additional module was developed, and that it was accepted that the experiment be performed during the summer.

As it turned out, the four CS-200 programming classes that were offered during the Summer term were all different languages (Pascal, Fortran, PL/I and Basic). I picked two classes at random and a class of students learning BASIC (the experimental group) was compared with a class learning PL/I (the control group). Also, there were two different instructors for the two classes.

- (3) Were there differences between instructors or between the populations of the two classes? In retrospect, a selection of two classes taught by the same instructor over the same subject matter would have been preferred. But during the summer semester of 1984 there were four CS-

200 classes taught, all covering different computer languages. An attempt was made to minimize instructor bias, given the fact that two different teachers were involved. Two out of the four were chosen based on the sex (male), age (early twenties) and experience of the instructors (both were graduate students working on a Masters Degree in Computer Science). The decision was supported by the drop rate of the two classes (about the same) and the attitudes of the students relating to their respective instructors as indicated by the results of the analysis of the questionnaires (see question 7).

(4) Why was student attitudes measured? Chambers, et al, [Cham 80] list several criteria for evaluating CAI products, including:

- 1) the amount of learning that initially takes place,
- 2) the amount retained over a given time period,
- 3) the drop rate of the individuals,
- 4) the attitude changes of the learners towards Computer Science and towards computers as an instructional medium,
- 5) the transportability of the material,
- 6) the acceptance of the materials by others involved in classroom instruction.

Dr. Jackson Byars, a member of the committee, suggested using attitude questionnaires early during the design. Using student attitude as a measure of success is an acceptable metric [Gall 77] [Educ 78]. In that regard, a dissertation by Bickerstaff [Bick 76] was used as a model for the questionnaire. The questions were rephrased to center on CAI in computer science.

- (5) Why were there just 47 questions? Initially, 92 questions were drawn from Bickerstaff's dissertation. After some consultation with Dr. Corwin Bennet [Benn 84] on the suitability of the questions, about half of the questions were discarded. The reasoning was that the students would tire of answering so many questions and just start checking questions at random, invalidating the results.
- (6) Why have a monitor present during the evaluation? Cognizant of the limitations of most micro-computers and on the advice of Dr. C.E. Bennet [Benn 84], the decision was made to have a monitor available to help the students if the hardware or software should malfunction. It was decided that there would be less bias if differing amounts of help were provided to the students. In some cases, there was no one to help, in others, someone was in the room and available if something should go wrong. And in others, the monitor was sitting directly alongside the student to give help when asked, in the manner of an informal conversation. The decision to make someone available to (and sitting alongside the) students was not universally accepted, as was discerned from student comments. In fact, some of the students expressed the desire to work entirely alone. In retrospect, having someone in the room (but not necessarily looking over the students shoulder) is the best course of action in similar situations.
- (7) What statistical tests were used? For tests T1 and T4, the class sizes were large enough to apply a statistical test that requires an approximately normal distribution of the sample population, therefore a t-test was deemed appropriate. For tests T2 and T3, the evaluation included

comparisons of attitudes of individual students before and after exposure to the software package, therefore, a A Sign Test with a Paired Sample was chosen.

#### 4.3 Details of the Experiment

The 47 questions that were given to the students are shown below. Appearing before each question are three groups of characters. The first group of characters are the digits that make up the question number. Those items with an 's' in the second group signify questions that pertain to the students attitude toward computer science; those marked with an 'i' pertain to CAI, those marked with an 'f' indicate that the student perceives he/she learns faster with CAI; the questions marked with a 'c' were cross-check questions; and the one marked with an 'n' relates the students attitude toward the instructor. The third group consists of just one character, either a plus (+) or a minus (-). The plus indicates that the question is phrased positively, the minus indicates that the question is phrased in a negative manner.

- 1 ..f.. + The availability of CAI helped me to learn the course material faster.
- 2 s..c. + I feel the Computer Science Department is doing an excellent job of helping the students.
- 3 s.... - The Computer Science Department is not very attentive to the student's needs.
- 4 s..c. - The Computer Science Department shows only token concern for the student.
- 5 s.... + The feeling that I have toward Computer Science is a good feeling.
- 6 s.... - In the Computer Science Department I have had a hard time finding someone to answer my questions.
- 7 ..... - I feel uncomfortable about seeing my Computer Science instructor.
- 8 .i... + If CAI were available in Computer Science I would use it regularly to do my homework.
- 9 .i... + Each student should be required to work through at least one CAI homework module during the semester.
- 10 .i.c. + Computer-assisted instruction is better than traditional instruction.
- 11 ..f.. - My mind goes blank, and I am unable to think clearly when working on computer programs.
- 12 ..f.. - I learn faster by doing programming problems than by CAI.
- 13 .if.. - I learn best by working closely and directly with a teacher.
- 14 s.... + I really like computer science.
- 15 .i... + A person can concentrate better on learning the material with CAI than in a classroom with other students present.
- 16 s.... + I feel a definite positive reaction to computer science; it's enjoyable.
- 17 .i... + Using CAI to do homework is an efficient use of students' time.
- 18 s..c. + I feel at ease in computer science, and I like it very much.
- 19 s.c. + Computer Science is fascinating and fun.
- 20 ..... + I have a good memory.
- 21 s.... + I am much happier in computer science class than in any other class.
- 22 .i... - I like the idea of a teacher standing by to give me help whenever I feel I need it.
- 23 .i... + I would enroll in a recitation section that required homework to be done by CAI.
- 24 .i... - I am not in favor of CAI because it depersonalizes instruction.

Table 4.1.a Questionnaire



- 25 .if.. + Learning by CAI is like having a private tutor.
- 26 .i... + If a person knowledgeable in CAI and Computer Science were present at the microcomputer I would do my homework via CAI.
- 27 s..c. - I do not like Computer Science and it scares me to take it.
- 28 .i... - I have never liked computers, and it is my most dreaded subject.
- 29 .i... - Even otherwise interesting material would be boring when presented by CAI.
- 30 s..c. - Computer Science makes me feel as though I'm lost in a jungle of strange terminology and can't find my way out.
- 31 .i... - A person becomes more involved in running the machine than in learning the material (with CAI).
- 32 s.... - It makes me nervous to even think about having to do a computer program.
- 33 .i.c. - Computer-assisted instruction does not allow a person to learn quickly.
- 34 .i... + Material which is otherwise boring can be interesting when presented by CAI.
- 35 s.... - I approach computer science with a feeling of hesitation, resulting from a fear of not being able to understand how computers work.
- 36 .i... + The ideal learning situation involves one student and one teacher.
- 37 s.... + Computer Science is something I enjoy a great deal.
- 38 s.... + I would probably take some advanced Computer Science courses, even if they are not required in my curriculum.
- 39 .i... - Not being able to ask a teacher questions is a definite disadvantage of CAI.
- 40 .i... - Whenever I have to operate a machine, I become nervous.
- 41 .i... + I would recommend to other Computer Science students that they try to do homework via CAI.
- 42 s..c. - I am always under a terrible strain in a Computer Science class.
- 43 .i... + The advantages of using CAI in Computer Science outweigh the disadvantages.
- 44 s..c. + Computer Science is very interesting to me and I enjoy computer courses.
- 45 .ifc. + I learn faster with CAI than with conventional classroom instruction.
- 46 .i.c. - I learn slowly with CAI.
- 47 .i... + A person can work at his own pace with CAI.

Table 4.1.a Questionnaire

A summary table of the various categories and the number of questions per category is shown below.

	+	-	C+	C-
s	6	4	4	4
i	12	8	2	2
f	2	3	1	0
n	0	1	0	0

A Summary Table of Question Categories

where s indicates student attitude  
 where i indicates CAI  
 where f indicates learns faster  
 where n indicates instructor

Numeric values have been assigned to the various responses as follows:

- 1 strongly disagree
- 2 disagree
- 3 undecided
- 4 agree
- 5 strongly agree

The following notation is used:

- u1 = represents the attitudes of the FL/I class, on June 5
- u2 = represents the attitudes of the FL/I class, on June 15
- u3 = represents the attitudes of the Basic class, on June 5
- u4 = represents the attitudes of the Basic class, on June 15

Tests were done for each question using the normalized mean  $uk[i]$  for question[i], where  $i = 1..47$ , for each class data set,  $k = 1..4$ .

Using this notation, the six tests can be restated as:

- T1.  $u1 = u3$  that the attitudes of the students in both classes, at the start of the experiment, were statistically the same. A t-test was used for the a priori test.

- T2a.  $u_2 > u_1$  that there was a positive reaction by the students (to the overall material) in the Basic class. A Paired Test was performed on the responses of the students for each question in each class.
- T2b.  $u_2 > u_1$  that there was a positive reaction by the students (to the overall material) in the Basic class. A Paired Test was performed on the responses of the students for each question for each student for each class.
- T3a.  $u_4 > u_3$  that there was a positive reaction by the students (to the overall material) in the FL/I class. A Paired Test was performed on 1) the responses of the students for each question in each class. This assertion will be shown to be false.
- T3b.  $u_4 > u_3$  that there was a positive reaction by the students (to the overall material) in the FL/I class. A Paired Test was performed on the responses of the students for each question for each student for each class.
- T4.  $u_4 > u_2$  that the reaction by the students in the class that used the CAI package was more positive than the reaction of the students in the control group. In fact, the test is for:  $(u_4 - u_3) > (u_2 - u_1)$ . Assuming that the classes have the same initial attitudes (i.e.  $u_3$  is approximately equal to  $u_1$ ), then the test determines the attitudes of the students posteriori. A t-test was used.

4.3.T1 The attitudes of the two groups were the same, a priori

Test 1 was used to determine the outcome of the following hypothesis:

$H_0 : u_1 = u_3$

where  $H_0$  is the hypothesis being tested,  $u_1$  is the composite value of the responses given by the students in the Basic class before the experiment was run, and  $u_3$  is the composite value of the responses given by the PL/I students before exposure to the software package.

The algorithm used for case T1 is listed, below, in pseudo-code. The following Pascal record is indicative of the input to the algorithm used for all of the cases.

```
questions :  
  array [1..47] of  
    record  
      number      : integer;  
      sign        : ('+', '-');  
      response    : array [june5, june15] of  
        record  
          x       : array[1..5] of integer; { PL/I }  
          y       : array[1..5] of integer; { Basic }  
        end  
      end  
    end
```

Record layout of the input data used

```

for {each question} q := 1 to 47 do
  with questions[q].response[day] do
    begin
      for i := 1 to 5 do
        begin
          xbar := xbar + i*x[i];
          ybar := ybar + i*y[i];
          xtotal := xtotal + x[i];
          ytotal := ytotal + y[i]
        end;

      xbar := xbar / nx (number of x's);
      ybar := ybar / ny (number of y's);

      for i := 1 to 5 do
        begin
          variance_of_x := variance_of_x+(xbar-i)**2 * x[i];
          variance_of_y := variance_of_y+(ybar-i)**2 * y[i]
        end;

      variance_of_x := variance_of_x / (nx - 1);
      variance_of_y := variance_of_y / (ny - 1);

      z := (ybar - xbar)/sqrt(variance_of_x/nx +
        variance_of_y/ny);

      print number,
      print '11',
      for i := 1 to 5 do print x[i],
      print xtotal, xbar, varx, z, xbar-ybar, conf_int
      print 'basic',
      for i := 1 to 5 do print y[i],
      print ytotal, ybar, vary;
    end
  end
end

```

#### The Algorithm used for cases T1 and T4

The results of the above analysis are given in appendix E and are summarized in Table 4.1.a. The z value (the normalized differences of the mean) indicates whether the corresponding means are significantly different. A test procedure is said to have level of significance alpha (in this case .1), or to be a level alpha test if the probability of a type 1

error  $\leq \alpha$  [Devo 82, p103]. A type 1 error consists of rejecting the null hypothesis,  $H_0$ , when it is true. [Devo 82, p101]

There were two sections with 23 and 36 students (in PL/I and Basic, respectively) responding. The degrees of freedom, therefore, are 22 and 35, respectively. In a t-test with an alpha of .1, any result, such that the z value is greater than 1.65 (for the Basic class) or 1.717 (for the PL/I class), is considered significant. Since alpha equals the probability of rejecting  $H_0$  when it is actually true and since alpha is chosen to be .1 or 10%, on a random basis, approximately five questions might show significance through chance error. Test T1 compares the attitudes of the two groups of students before the experiment took place. The test shows six questions significant. Those questions are: 32, 33, 35, 41, 42, and 45.

The results of Question 32 and question 35 indicate that the students in the BASIC class were more nervous than the students in the PL/I class, a finding one would expect (if the results are not due to chance.) It is likely, that, most of the BASIC students were first time programmers, whereas most (or many) of the PL/I students were not, hence the PL/I students would display more confidence the first day of class.

The conclusion is, therefore, that the attitudes of the students in the two classes with regards to the categories tested were about the same.

Question 7 was used to determine the existence of instructor bias. On June 5, the measure of significance (z-value) of the difference between the two means of the two classes was -0.09, i.e. no significant difference.

- - - - - J u n e 5 - - - - -					- - - - - J u n e 5 - - - - -				
	- m e a n -		sign	z-value		- m e a n -		sign	z-value
	pl1	basic				pl1	basic		
1	3.00	3.00	+	0.00	24	2.87	2.66	-	-1.51
2	2.76	2.95	+	1.08	25	3.27	3.25	+	-0.20
3	2.87	2.89	-	0.11	26	3.15	3.30	+	1.10
4	2.92	2.89	-	-0.32	27	2.85	2.55	-	-1.40
5	3.26	3.52	+	1.17	28	2.46	2.32	-	-0.70
6	2.95	3.02	+	0.71	29	2.90	2.77	-	-0.95
7	2.67	2.68	-	0.09	30	2.90	2.82	-	-0.41
8	3.36	3.36	+	0.03	31	2.92	2.89	-	-0.36
9	3.15	3.16	+	0.03	32	3.28	2.70	-	-2.47 *
10	2.95	3.00	+	0.28	33	2.95	2.75	-	-2.05 *
11	2.95	2.75	-	-1.07	34	3.13	3.02	+	-1.05
12	2.95	3.05	-	0.85	35	3.23	2.82	-	-1.69 *
13	3.64	3.45	-	-0.91	36	3.08	2.84	+	-0.93
14	3.15	3.23	+	0.43	37	2.87	3.07	+	1.11
15	3.13	3.11	+	-0.10	38	2.85	2.86	+	0.08
16	3.03	3.18	+	0.85	39	3.36	3.27	-	-0.63
17	3.21	3.16	+	-0.38	40	2.36	2.20	-	-0.83
18	2.90	3.07	+	1.02	41	2.87	3.11	+	2.37 *
19	3.23	3.30	+	0.36	42	2.95	2.68	-	-1.76 *
20	3.62	3.89	+	1.36	43	3.03	3.09	+	0.62
21	2.74	2.64	+	-0.68	44	3.00	3.14	+	0.81
22	4.00	3.95	-	-0.22	45	2.92	3.07	+	2.19 *
23	3.23	3.18	+	-0.39	46	3.00	2.95	-	-0.53
					47	3.48	3.27	+	-1.43

Results of the Sign Test for both classes on June 5  
Table 4.1.a

- - - - June 15 - - - -				- - - - June 15 - - - -					
	- m e a n -		sign	z-value		- m e a n -		sign	z-value
	pl1	basic				pl1	basic		
1	3.06	3.24	+	1.25	24	2.72	2.62	-	-0.62
2	3.06	3.07	+	0.08	25	3.08	3.12	+	0.22
3	2.67	2.83	-	0.87	26	3.17	3.50	+	1.86 *
4	2.94	2.66	-	-1.56	27	2.78	2.17	-	-2.57 *
5	3.31	3.67	+	1.42	28	2.58	2.19	-	-1.61
6	2.83	3.24	+	1.86 *	29	2.78	2.40	-	-2.14 *
7	2.36	2.10	-	-1.28	30	3.03	2.64	-	-1.49
8	3.22	3.38	+	0.94	31	3.11	3.12	-	0.03
9	3.19	3.45	+	1.41	32	3.25	2.74	-	-2.10 *
10	2.78	2.88	+	0.51	33	2.81	2.50	-	-1.98
11	2.94	2.14	-	-3.86 *	34	3.19	3.31	+	0.79
12	3.06	2.83	-	-2.11 *	35	3.31	2.76	-	-2.23 *
13	3.58	3.52	-	-0.28	36	3.56	3.10	+	-1.87 *
14	3.06	3.51	+	1.87 *	37	2.75	3.24	+	2.00 *
15	2.86	3.05	+	1.24	38	2.53	2.90	+	1.40
16	3.28	3.52	+	0.96	39	3.50	3.79	-	1.40
17	3.08	3.33	+	1.73 *	40	2.81	2.26	-	-2.71 *
18	3.00	3.37	+	1.45	41	3.03	3.17	+	0.96
19	3.22	3.45	+	0.93	42	2.89	2.33	-	-2.91 *
20	3.47	3.90	+	2.19 *	43	3.03	3.17	+	1.08
21	2.44	2.50	+	0.25	44	2.92	3.40	+	2.00 *
22	3.81	3.71	-	-0.37	45	3.00	2.95	+	-0.49
23	3.06	3.02	+	-0.19	46	2.92	2.84	-	-0.71
					47	3.25	3.70	+	3.08 *

Results of the Sign Test for both classes on June 15  
Table 4.1.b



#### 4.3.12a Positive Reaction by the Basic Students (stu-ques-class)

This test was used to determine whether there was a positive reaction by the students (to the overall material) in the Basic class. Tests 2 was analyzed using two approaches. A Sign Test with a Paired Sample was performed on the responses of the students. (The input to both is shown in appendix F.) For this approach, the number of students was tallied who 1) indicated a positive change in attitude (for positively phrased questions), or 2) who indicated a negative change in attitude (for a negatively phrased question). For this case, particular interest was placed in the change of attitude recorded by each student for each question within each class.

The hypothesis:

$$H_0 : u_2 = u_1$$

$$H_a : u_2 \neq u_1$$

where  $H_0$  is the hypothesis being tested, that there was no change in attitude in the Basic class at the end of the experiment.  $H_a$  is the alternative hypothesis, that there was a change in the attitudes of the students in the Basic class. In this case, if  $H_0$  is rejected then  $H_a$  will be accepted.  $u_2$  represents the responses given by the students in the Basic class after the experiment was run, and  $u_1$  represents the responses given by the Basic students before exposure to the software package.

For clarity, the above hypotheses is rephrased to demonstrate that there are 47 separate hypotheses tested:

$h0[i] : u2[i] = u1[i], \text{ for } i = 1..47$

It will be shown that there occurred 13 rejects, therefore the conclusion is:

ha:  $u2 > u1$

the data record and the algorithm used in Tests T2 and T3 are as follows:

```
data :
  record
    class : array [basic..pl1] of
      record
        student : array [1..number_of_students] of
          record
            day : array [june5..june15] of
              record
                responses : array [1..47] of integer;
              end
            end
          end
        end
      end
    end
  end
```

Layout of the Record used in Tests T2 and T3

```

for {each question} q := 1..47 do
begin
with questions[q] do write(number, sign)
for {each student} s := 1 to number_of_students do
begin
t := data[basic.student[s],day[june15].responses[s]]
- data[basic.student[s],day[june5].responses[s]];
if t > 0 and sign[question] = '+'
or t < 0 and sign[question] = '-'
then pcount := pcount + 1
else ncount := ncount + 1
end;
z := (pcount/total - 0.5 - 0.5/total) / sqrt(.25/total);
print pcount, ncount, total, z
end

```

The Algorithm used in Tests T2a and T3a

The results of this analysis are shown in appendix G and are summarized below.

basic class: number of questions  
that signal a significant change

significantly positive	13
not shown significant	32
significantly negative	2
<hr/>	
total	47

A Summary of Appendix G (Basic Class)

For the Basic class, a statistically significant positive shift in attitude appears in questions 1, 7, 9, 11, 12, 16, 17, 18, 27, 33, 34, 42 and 47. This group of questions represent a mix of attitudinal questions about Computer Science, CAI and one question concerning the instructor. There

was one statistically significant negative shift in attitude. This question concerned teacher accessibility and did not relate to CAI.

H<sub>0</sub> is rejected and the conclusion is that the students reaction to CAI, and Computer Science in general, is more positive as a result of using the CAI package.

#### 4.3.12b Positive Reaction by the Basic Students (ques-stu-class)

To further check validity, the responses in the Basic class analyzed by question for each student within each class. A Sign Test [Uken 78, p.297] was used to determine the changes in attitude that took place for each student in the Basic class.

```

print 'basic',
for each student do
begin
print student number,
for each question [1..47] do
begin
t := data[question #, after] - data[question #, before];
if t > 0 and sign[question] = '+'
or t < 0 and sign[question] = '-'
then pcount := pcount + 1
else ncount := ncount + 1
end;
total := pcount + ncount;
z := (pcount/total - 0.5 - .5/total) / sqrt(.25/total);
print pcount, ncount, total, pcount/total, z
end

```

The Algorithm used in Tests T2b and T3b

The results are shown in appendix H and summarized below. With respect to individual students, almost 1/2 (16/36) of the BASIC students registered a shift.

basic class: number of students that signal a significant change	
significantly positive	16
not shown significant	18
significantly negative	2
<hr/>	
total	36

A Summary of Appendix H (Basic Class)

Almost half of the students in the Basic class (16 out of 36) had a positive change in attitude at the end of the ten day period.

4.3.T3a PL/I Class had a positive change in attitude (stu-ques-class)

This test was used to determine whether there was a positive reaction by the students (to the overall material) in the PL/I class. A Signed Test with a Paired Sample was used on the responses of the students to measure the change of attitude recorded by each student for each question within each class. (The input is shown in appendix F.) The algorithms used are the same as in test T2a, with the exception that the class is PL/I.

The hypothesis:

$H_0 : u_4 \neq u_3$

$H_a : u_4 \neq u_3$

The results of this analysis are shown in appendix G and summarized below.

PL/I class: number of questions that signal a significant change	
significantly positive	1
not shown significant	38
significantly negative	8
<hr/>	
total	47

A Summary of Appendix G (PL/I Class)

For the PL/I class, there is a noticeable positive shift in attitude

only on question 36. Since alpha = .1, there could be up to five items (at random) that would show a significant change. Overall there is no noticeable positive shift in attitude for the PL/I class during the ten day period, if anything, there is a noticeable negative shift in attitude (8 out of 47), posteriori. Therefore, H0 is accepted.

4.3.T3b PL/I Class had a positive change in attitude (ques-stu-class)

To ensure validity, there was an analysis of the responses for each question for each student within each class (see T2 above). The results are shown in Appendix H and are summarized below.

PL/I class: number of students that signal a significant change	
significantly positive	7
not shown significant	12
significantly negative	4
-----	
total	23

A Summary of Appendix H (PL/I Class)

With respect to individual students, less than 1/3 (7/23) of the PL/I students registered a noticeable positive shift in attitude, whereas almost 1/2 (16/36) of the BASIC students registered a shift; therefore, H0 is rejected.

4.3.T4 The Change For the Basic Class was greater than the PL/I Class.

This test determines that the shift in attitude of the students in the

Basic class was greater than the shift in attitude of the students in the FL/I class.

The conventional statistical notation used is as follows:  $H_0$  is identified with the hypothesis of no change (from current opinion), i.e. no departure (from current beliefs). In scientific investigations  $H_0$  is often the 'status quo' claim. The burden of proof will rest with  $H_a$  in the sense that  $H_0$  will be accepted as true unless the experimental evidence strongly suggests otherwise.

The hypothesis for case 4:

$$H_0 : u_4 = u_2$$

$$H_a : u_4 \neq u_2$$

The t-test evaluation for case T4 is the same one used for case T1, with the exception that the data reflect the attitudes of the students at the end of the testing period. The results of the analysis are given in appendix E and are summarized in Table 4.1.b (already shown). The z value (the normalized differences of the mean) indicates whether the corresponding means are significantly different.

The results of the same questions after the experiment took place indicate that the FL/I students show more confidence than the BASIC students ( $z = -2.10$  and  $z = -2.23$ ). A significant change in attitude between the two classes, at the end of the ten day period, was detected. There are a total of 18 questions where the responses show a significant statistical difference between the two classes. The results (on June 5) from Questions 33, 41 and 45 display a positive attitude toward CAI, a priori. There are



24 questions that relate to CAI, and for a 10% test, two to three of the questions may show significance, due to chance. If the results are conclusive, then the attitude of the students (on June 15) dropped slightly with respect to the above three questions. This is more than offset by the results of June 15, where the responses were statistically significant (in favor of CAI) in questions 12, 17, 25, 29, 33, 40, and 47.

The students response to Question 42 is significant before and after the questionnaire was given. The students in the BASIC class felt even less stressful than the students in the FL/I group, after the ten day period. A significant change in attitude between the two classes, at the end of the ten day period, was detected. There are a total of 18 questions where the responses show a significant statistical difference between the two classes. The results from Questions 33,41 and 45 display a positive attitude toward CAI, a priori. If the results are conclusive, then the attitude of the students dropped slightly with respect to the above three questions. This is more than offset by the results of June 15, where the responses were statistically significant (in favor of CAI) in questions 12, 17, 25, 29, 33, 40, and 47.

The students response to Question 42 is significant before and after the questionnaire was given. The students in the BASIC class felt less stressful than the students in the FL/I group, especially after the ten day period.

Question 7 was used to determine the existence of instructor bias. On June 5, the measure of significance (z-value) of the difference between the two means of the two classes was -0.09, i.e. no significant difference. On

June 15, the z-value was 1.28, a small, but insignificant, positive shift. Ten of the students in the PL/I class failed to write their identifying numbers on their response cards on June 15. This left a sample size of 23 in the PL/I class as opposed to 36 in the BASIC class. Had the sample size from the PL/I group been larger, a significant result may have been obtained for question 7. The responses, if any, of the students to both teachers was positive, reducing the impact of instructor bias in the results. The conclusion, therefore, is that the students reacted to the different instructors in a like manner.

#### 4.4 Conclusions

A statistical test of user attitudes based on responses from a questionnaire was performed for the Puzzle module. After some study, the conclusion was drawn that the change in student attitudes was significantly positive and that the experiment could be considered a success for the following reasons: The attitudes of the two classes on June 5 are about the same. The Basic class had a positive shift in attitude, the PL/I class demonstrated a slightly negative shift in attitude, and the shift in attitude of the Basic class was more positive than the shift in attitude of the PL/I class.

In all experiments of this nature, there comes a moment of truth, a time when the question must be asked, 'Was the experiment successful?' After a review of the results, one concludes that there seems to be a change. Now the important question, 'What could have caused it?' Was the change in attitude the result of the students' exposure to the Puzzle

module, or was there some other factor? Some of the possibilities are:

>> There may have been Instructor Bias? Using the results listed above, Instructor Bias was ruled out.

>> The result may have been due to the Hawthorne Effect, i.e. the performance from individuals cognizant of their participation in an experiment has been shown to be different.

>> The students may not have had the same assignments? As the questionnaires were passed (on June 5) the students were informed of their involvement in such a way as to minimize student bias with respect to the experiment. The PL/I class was told that they did not HAVE to use the software package, and the Basic class was told that they GET to use the software package in lieu of a regular homework assignment.

Half of the questions related directly to CAI, and the package that the students used was considered (at least, by them) as a CAI package, and a statistically significant number of the questions that show significance were CAI questions, therefore the conclusion is that the change in attitudes was CAI related.

In summary, the statistical results were not due to randomness of the data, nor to instructor bias. The Puzzle module seems to have helped with the attitude of the students about computing.

## Chapter 5 - Conclusions of this work

### 5.1 Review of the work

This dissertation has presented a reductionistic model used for conveying computer science concepts to students with little or no expertise in computer science.

Based on the initial investigations and research, four principles emerged for the development of this work. These four principles make up the minimum requirements that an instructional CAI package should have: a reductionistic presentation, direct manipulation graphics, an instructional mode, and an execution mode. Justification for these requirements follow.

- Reductionism is the basis and the structure of most textbooks that teach computer science and is a generally accepted approach.
- Graphics facilities offer a high bandwidth for transfer of information as is evident from the literature and from the emerging technology of instructional packages on micro-computers.
- An Instructional mode is necessary with any package used as a supplement for instruction.
- Because of the nature of the discipline, an Executable mode is a requirement. A student cannot do any useful work without a programmable system, and execution of programs forms the cornerstone of Computer Science.

At the time of writing, there were no completely integrated CAI packages that combine instruction, direct manipulation graphics and execution

together into one coherent program.

The contribution of this work then is a model that integrates instruction, direct manipulation graphics, and execution into one package, an approach not currently used in the instructional programs of the Computer Science curriculum.

Several important characteristics of the model are the following:

- (1) The model is designed to provide motor-learning experiences with immediate feedback.
- (2) The model uses a 'friendly' environment in which familiar concepts are translated, automatically, into more formal programming statements which are then displayed on a terminal.
- (3) The model is designed around the idea that while a student is solving a problem, a 'program' (reflecting the students actions) is introduced, using a more formal syntactic notation. Program syntax, and some semantic rules, are conveyed on a real-time basis.
- (4) The model is made up of several levels (consisting of a sequence of nine phases each) where each level is designed to convey more advanced concepts than the previous levels. The phases were designed in such a way that one is able to build on previously learned material while introducing progressively harder material.
- (5) Since the object of the package is to convey Computer Science concepts, it was decided to adopt a language that was not one of

the standard programming languages. The opinion reflected in this particular guideline is not universally accepted. Holt, et al, [Holt 77] argue for teaching a subset of an existing language. They listed the following four criteria for an instructional language: 1) it should be (part of) a 'real' programming language used in business, science, and government and should be appropriate for introducing programming concepts used in those areas, 2) it should encourage systematic problem solving and structured programming, 3) it should be a small, convenient, easy to master language, and 4) it should be easy to support by economical, diagnostic language processors on a variety of machines including minicomputers. Of these, arguments two to four are accepted as valid, and have been incorporated into the design. Argument one was abandoned. There is no desire to allow the student to fall into the trap of confusing the syntax of a new language with the semantics of the statements, a problem discovered by Ledgard [Ledg 80]. Exposure to two syntactically disjoint languages in parallel (one in the classroom and the other from the software package) would aid the student in discerning the difference between the semantics of the concepts and the syntax of a given statement. Hence, the "language" chosen for inclusion within the package is not one of the popular languages, nor a subset of one. The language was one developed during the design phase of the model.

There are three modules defined according to the model, a Puzzle module, a Business module (involving a Data Base) and a Grade Point Average

module. The Grade Point Average module is widely used in conventional programming classes and is designed to convey to the student the concepts of 'assignment statements', arrays, input/output, formatting and editing. There were two modules implemented according to the model, the Business module and the Puzzle module. The Business module was used to introduce procedure 'calls' and procedure declarations to the student. Although implemented, this game was not tested in a classroom environment and therefore no statistical tests validating the game are available. The Puzzle module relates puzzle manipulations to algorithmic processes, and introduces looping, decision and input/output programming constructs. The Puzzle module was implemented on an ATARI micro-computer and was used by an instructor of CS-200 as a supplement to a classroom environment. The responses of the students, regarding the suitability of the package appears in Chapter 4 of this dissertation.

## 5.2 Guidelines

In addition to the four principles for the design of the model, several guidelines evolved during the design phase and have been identified. These guidelines were drawn from the literature and from 12 years experience in dealing with (and observing) students in an Introduction to Programming laboratory.

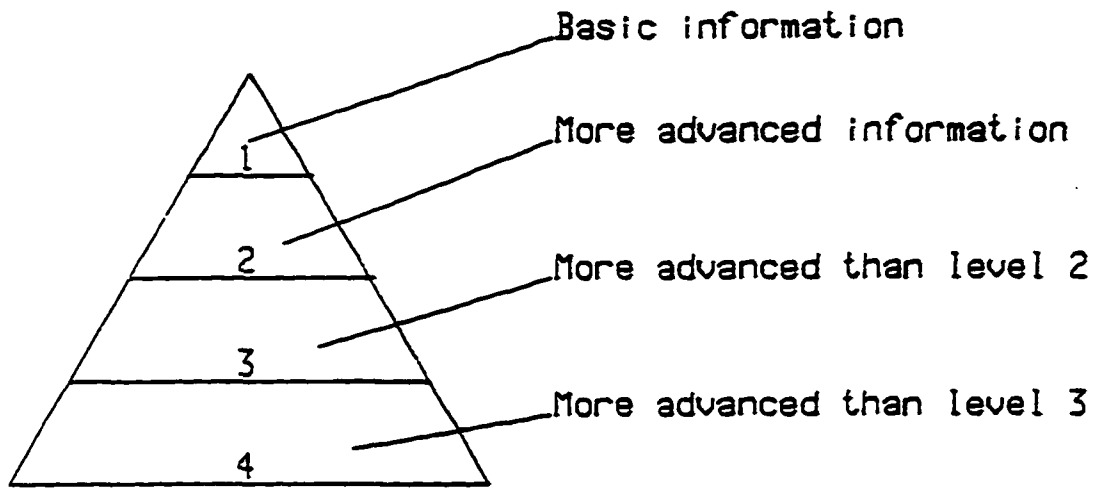
1) The first guideline is that the commands available at each particular level should be closed, i. e. everything the student needs to know to work at a given level has been explained and everything the student needs to do, at that level, can be done with the available tools (commands).

Figure 5.a. displays a hierarchy of concepts to be conveyed to the student. At the top is the simplest material and is explained to the student first. The concepts (at this level) are closed in the sense that dropping to the next (more advanced) level is not required in order to complete the work required at the top level. As an example, in a programming course, the PRINT statement (or its semantic equivalent) should be covered first. With that one statement the student has the ability to 'see' the results of a given command. Also, the student can then use the PRINT statement to find errors in programming later on.

The above idea is related to, but differs slightly from, the idea that statements and concepts can be understood in vacuo - the programmer need not assimilate a great deal of information about the environment of a statement/command in order to understand it. Most of the information, needed for understanding, can be derived from existing information. This approach runs counter to integrated systems. In many cases, where programming is involved, the student is required to learn and understand the majority of the components of a system (and their interlocking relationships) before being able to perform the simplest task [Holt 77]. This holistic view, as described, delays reinforcement, in the form of correctly running programs which leads to discouragement, and results in some dissatisfaction with computer science.

2) Learning should proceed from the known to the unknown. This guideline has a sound theoretical basis. Gibbs holds that 'the learning' process involves the incorporation of a new learning task into existing cognitive structure so that a meaningful relationship is established'





The set of concepts that are taught at a given level should be closed

Figure 5.a.

[Gibb 67]. Mayer views meaningful learning (of technical information) as a process by which the learner connects new material with existing knowledge [Maye 81]. He cites arguments by Ausubel [Ause 63] that learning may be enhanced by using advanced organizers that provide general concepts and ideas prior to exposure to new material. Papert describes learning as the process of assimilation of new facts to existing cognitive models. He states that what one learns and how one learns it depends upon available models [Pape 80, p. vii].

Mayer & Bayman [Maye 81b] argue that beginning programmers develop a mental model of a computer language [Baym 83]. The student who uses the 'glass box' approach (i.e. understands what each command does in terms of a more primitive machine) achieves superior long term retention and superior transfer to novel problems than the student who uses the black box approach (i.e. memorizes the sequence of commands that performs a specific task).

It is for this reason that the strategy of displaying formal programming text in response to actions provided by the student has been adopted. That is to say, the student manipulates icons on a TV screen and these actions are manifest with generated code.

3) The third guideline followed is: the order of doing programming (writing a program from 'scratch') is in reverse order than learning about programming. It is generally accepted that the most effective and least error prone method for creating reliable software is to perform the follow-

ing actions in the order shown:

- a) Requirements Analysis - describe the functions of the components of the system, and their inter-relationships via an i/o analysis - i.e. data flow.
- b) Design - decide upon the components of the system (hardware and software).
- c) Code and Test - develop the components of the system.
- 4) Operate - use the system.

However, learning how to program does not fit this model. "Programmed instruction ... proceeds in small steps" with an active learner (Craik 80). Therefore, the first thing the student should do is to do something with his/her hands (i.e. operate the equipment). Once the student is familiar with the various pieces available, then the components that make up the pieces are described (define the modules). After the modules have been understood then the purpose of each module is given, then the relationships between the modules (data flow) are given.

This order is a workable one that builds from the known to the unknown, and one that deserves wider acceptance within the academic community.

4) The fourth guideline is that Interactive learning aids in retention of key concepts. Immediate feedback 'lets the learner know... whether or not he has mentally processed the new information correctly' [Gibb 67]. A positive response 'reinforces the cognitive activity, and ensures the student that he is progressing satisfactorily' [Gibb 67]. The package

described in this dissertation is analogous to direct manipulation systems described by Shneiderman [Shne 83]. These systems encompass screen editors, Visicalc, video games, computer aided design packages, and any other package that gives the user exactly what is being displayed on a video screen.

5) Learning should be by example and not by trial and error. The concern here is with efficiency. The concepts to be taught in a CS-200 course are known, a priori, and experimentation by the student is not necessary. Every level starts with an example that generates code. The student copies this code, and then later on in the sequence enters a slight variation, or, in some instances, creates new code. At no time is the user at a loss as to what he/she should do with the new material, a method which is appropriate at this level. During the design phase of the model, it was decided that more control be used when conveying computer science concepts to students, an approach that should be adopted in all instructional packages.

This concludes the list of guidelines followed during the design phase of this work.

### 5.3 Future Work

The tools in the commercial arena are moving in the direction of software packages like the one described in this dissertation, and more and more, this approach is becoming accepted. However, no packages exist, yet, that combine direct manipulation graphics and instruction with execution of user programs.

A complete model that covers the entire spectrum of concepts in Computer Science needs to be implemented and tested in a classroom environment.

## Appendices

The contents of the appendices are as follows:

Appendix A - The instructions given to the students (as a hardcopy) for the operation of the micro-computer.

Appendix B - The questionnaire given to all of the students in both classes on both dates. The sign (plus or minus) after the question number indicate whether the question was phrased in a positive manner or a negative manner with respect to Computer Science, CAI, etc.

Appendix C - A listing of the responses given by all of the students in both classes. The first four digits make up the student identifying number (we used the last four digits of the students Social Security Number), the next three characters indicate the class and date.

p05	indicate	PL/I	June 5
b05	indicate	Basic	June 5
p15	indicate	PL/I	June 15
b15	indicate	Basic	June 15

Appendix D - A summary of the data by question. The first column of numbers is the question number, the second column is a + or - (to indicate a positive or negatively phrased question). The next five columns (see 0 1

37 1 0 of the top row of numbers) constitute the total number of students in the HL/I class on June 5 who responded 1,2,3,4, or 5, respectively. The second group of five columns are the totals for the responses from the Basic students on June 5. The next 10 columns are the responses from June 15 ( five columns for HL/I, and five for Basic).

Appendix E - The results from the statistical analysis described in Chapter 4.

Appendix F - A listing of the raw data with the responses from students who omitted their Social Security Number on June 15 removed. We used this data set for the paired sample sign test.

Appendix G - The result from the statistical analysis described in Chapter 4.

Appendix H - The result from the statistical analysis described in Chapter 4.

Appendix I - This form is required for all persons performing experiments on human subjects. Three copies were filed with the University Committee for Research with Human Subjects as per University guidelines.

Appendix J - A table displaying the classification of the various questions appearing in the questionnaire. The areas of interest were Computer Science (cs), Computer Aided Instruction (cai), student learning rate, and

**student responses to the instructor.**



## Instructions for operating the micro-computer

Open the door to the disk drive by pressing the black rectangular button below the small handle.

Insert the disk marked PUZZLE 1 into the disk drive, face up, bottom first.

Close the door to the disk drive.

Turn on the ON-OFF Switch to the Power Strip.

Turn the TV on, if it is not already on, by pulling on the on/off switch.

Turn the disk drive on if the red light on the drive is not already on. Wait for the top busy light to go out.

If READY does not appear on the TV screen, turn on the ATARI-800 machine. The ON/OFF switch is a rocker switch located on the right side of the keyboard. Wait for all sound effects to complete and for READY to appear on the screen.

Type in: DOS <return> where <return> is a single button. Wait for SELECT ITEM OR RETURN FOR MENU to appear.

Type in: L <return> Wait for it to ask for the filename.

Type in: B <return> Wait for - all sound effects to complete and for READY to appear and the busy light on the disk drive to go off.

Appendix A

page 133

Open the door to the disk drive

Remove the disk.

Insert the disk marked 1 SAV into the disk drive, bottom first.

Close the door on the disk drive.

Type RUN "D:TUT1.SAV" <return> Wait for the instructions to appear.

Appendix A

page 134

Directions: Each of the following statements expresses a feeling which a particular person has toward learning Computer Science. Please express, on a five-point scale, the extent of agreement between the feeling expressed in each statement and your own personal feeling. For some of the questions, you will have no opinion. Please mark box 3 for these questions. (Note: CAI is an abbreviation for "Computer Aided Instruction".)

The five points are:

- 1 strongly disagree
- 2 disagree
- 3 undecided
- 4 agree
- 5 strongly agree

#### ITEMS

- 1 + The availability of CAI helped me to learn the course material faster.
- 2 + I feel the Computer Science Department is doing an excellent job of helping the students.
- 3 - The Computer Science Department is not very attentive to the student's needs.
- 4 - The Computer Science Department shows only token concern for the student.
- 5 + The feeling that I have toward Computer Science is a good feeling.
- 6 - In the Computer Science Department I have had a hard time finding someone to answer my questions.
- 7 - I feel uncomfortable about seeing my Computer Science instructor.

Appendix B

page 135

- 8 + If CAI were available in Computer Science I would use it regularly to do my homework.
- 9 + Each student should be required to work through at least one CAI homework module during the semester.
- 10 + Computer-assisted instruction is better than traditional instruction.
- 11 - My mind goes blank, and I am unable to think clearly when working on computer programs.
- 12 - I learn faster by doing programming problems than by CAI.
- 13 - I learn best by working closely and directly with a teacher.
- 14 + I really like computer science.
- 15 + A person can concentrate better on learning the material with CAI than in a classroom with other students present.
- 16 + I feel a definite positive reaction to computer science; it's enjoyable.
- 17 + Using CAI to do homework is an efficient use of students' time.
- 18 + I feel at ease in computer science, and I like it very much.
- 19 + Computer Science is fascinating and fun.
- 20 + I have a good memory.
- 21 + I am much happier in computer science class than in any other class.
- 22 - I like the idea of a teacher standing by to give me help whenever I feel I need it.
- 23 + I would enroll in a recitation section that required homework to be done by CAI.
- 24 - I am not in favor of CAI because it depersonalizes instruction.
- 25 + Learning by CAI is like having a private tutor.

Appendix B

- 26 + If a person knowledgeable in CAI and Computer Science were present at the microcomputer I would do my homework via CAI.
- 27 - I do not like Computer Science and it scares me to take it.
- 28 - I have never liked computers, and it is my most dreaded subject.
- 29 - Even otherwise interesting material would be boring when presented by CAI.
- 30 - Computer Science makes me feel as though I'm lost in a jungle of strange terminology and can't find my way out.
- 31 - A person becomes more involved in running the machine than in learning the material (with CAI).
- 32 - It makes me nervous to even think about having to do a computer program.
- 33 - Computer-assisted instruction does not allow a person to learn quickly.
- 34 + Material which is otherwise boring can be interesting when presented by CAI.
- 35 - I approach computer science with a feeling of hesitation, resulting from a fear of not being able to understand how computers work.
- 36 + The ideal learning situation involves one student and one teacher.
- 37 + Computer Science is something I enjoy a great deal.
- 38 + I would probably take some advanced Computer Science courses, even if they are not required in my curriculum.
- 39 - Not being able to ask a teacher questions is a definite disadvantage of CAI.
- 40 - Whenever I have to operate a machine, I become nervous.

Appendix B

- 41 + I would recommend to other Computer Science students that they try to do homework via CAI.
- 42 - I am always under a terrible strain in a Computer Science class.
- 43 + The advantages of using CAI in Computer Science outweigh the disadvantages.
- 44 + Computer Science is very interesting to me and I enjoy computer courses.
- 45 + I learn faster with CAI than with conventional classroom instruction.
- 46 - I learn slowly with CAI.
- 47 + A person can work at his own pace with CAI.

Appendix B



9141 p05 3333333333 3353333335 3533323333 3333453331 3333333  
5616 p05 3344234322 3342224224 2233324434 3433422243 3332333  
0123 p05 3443143434 4342314224 2533434343 2333452141 3222243  
7540 p05 3333433433 3333333334 3433343333 3333233432 3333333  
5842 p05 2424432221 1545352454 3433422131 2133115442 3235333  
5720 p05 3433432323 3324444444 3433343232 2433423442 3333334  
9361 p05 3243431441 3353134333 3544433334 3433433244 3333333  
6408 p05 3333132553 4342513121 1541434223 3524521244 3331333  
4093 p05 4422332532 4343535241 1532333334 4433432242 3423333  
5852 p05 3422432244 3243443332 3433443234 3434443243 3433333  
1576 p05 3333431333 2333333342 3433332133 3233233332 3333333  
0671 p05 3222223333 2343333334 2433332232 3233243332 3233333  
0579 p05 3333434433 3344343344 3433433223 3433244433 3234333  
0104 p05 3532432333 3343333333 2433332233 3533513132 3233333  
2533 b05 3333533333 2315353554 4233331134 3133215532 3235333  
9254 b05 3333233433 3332323234 2433334334 3433432232 3333333  
1990 b05 3333433333 3333333334 3333333333 3333333333 3333333  
2327 b05 3333333443 4344343344 3433342333 3423433142 3333333  
9883 b05 3333333333 3353333334 3533333334 3433453332 3333333  
7644 b05 3333333334 3333333334 3333333333 3333533333 3333333  
5863 b05 3422413243 1344244454 2532221132 2133124452 3134435  
7133 b05 3332432444 3344244344 2432342222 2223223442 4243334  
5510 b05 3333433333 3333333334 3433333333 3333333333 3333333  
4246 b05 3422432445 2344444452 4441442222 3233333333 3333333  
7882 b05 3144342444 4343424234 1442444334 2423442142 3441334

Appendix C



7535 b05 3333534433 3333333234 3533334233 3333243334 3333333  
7229 b05 3333433443 3353333335 3533342233 3433243442 3333333  
7769 b05 3333233333 4322323224 2433334333 3533432134 3332333  
4530 b05 3422511443 1325354555 3551551111 3133125531 3135333  
7317 b05 3333333331 3333333335 3333333333 3133123433 3333333  
5731 b05 3122343441 2334342444 4432442222 3224224442 4244334  
5812 b05 3333532435 2334343444 3542432122 3234224432 3334334  
5493 b05 3423433433 2343333444 3434243232 3333223243 3334334  
6050 b05 3332533443 3343343334 3433341232 3223223332 3333333  
2242 b05 3322332323 2353334234 2423344334 4423422242 3432334  
8992 b05 3333433313 3333333334 3233332233 3233213432 3333333  
3534 b05 3243243333 3343333334 2433434434 3423443334 3333333  
2861 b05 3333433333 2332433334 3433332233 3433433334 3333333  
9601 b05 3534531333 5534533332 2552431223 3232213451 4233333  
7624 b05 3333532442 2343244332 1432332223 3423443232 3343334  
3638 b05 3333433333 3344333345 2433332232 3233233232 3234333  
8740 b05 3333333333 3343333334 3533333233 3333423332 3333333  
1783 b05 1333333333 3313333331 3133333233 3233223332 3233333  
6648 b05 3333432333 3323333334 2223332232 3233223432 3233333  
3859 b05 3223333443 2333333334 3534451121 3223243342 3233334  
3218 b05 3322233533 2332323324 3533332232 3233253232 3333333  
4785 b05 4244143434 4152422222 2532334434 3433442144 3432333  
1371 b05 3333333333 3333333334 3442343324 3333423432 3233333  
1792 b05 3333532333 3344333344 3332432131 3233224432 3233333  
5028 b05 5333533213 1535353554 3133421111 2131155331 3135333

Appendix C

2104 b05 4333532555 2325555555 2351531151 1115145521 5155515  
3441 b05 3233243343 2343333224 2433332234 3233432233 3432333  
5336 b05 3333233331 5353323134 3534335535 3533513332 3333333  
9549 b05 3333433333 2343333334 3433332233 3233243131 3334333  
1884 b05 3333331433 3353333334 2533333433 3333353141 3333333  
7876 b05 3333233331 3321313225 1532335534 3433422131 3332333  
8046 b05 3333333332 3343333334 3432343133 3233423322 3333333  
4642 b05 1333333113 335323324 3533333233 3333331131 3322333  
5616 p15 3244234333 4342313124 1533334535 3533442134 3432333  
9564 p15 3322422333 2333343334 2233332232 3233333232 3232333  
8352 p15 3333222333 3343333335 1533334434 3433341333 3432333  
0323 p15 3233233333 4322223223 1233343333 4334222313 3323333  
6445 p15 3423422333 3353343444 2533333335 3433453342 3333333  
9423 p15 3333332333 3343333334 2333334334 3433443332 3332333  
5852 p15 4322422243 4353333343 2232244224 2424453252 3233224  
--- p15 3323521332 1312253444 4533312122 5343332242 3233333  
1234 p15 3333432333 4343343244 2424332332 3433443234 3434333  
4093 p15 4333231432 5342424222 2432445435 5424451152 3332334  
5681 p15 3333432333 2344343444 2433332232 3433423232 3234333  
9687 p15 3522411345 4344333454 3433332132 3223224433 3344333  
7449 p15 3433432344 2334343444 3432333222 3233334332 3234333  
0142 p15 4333432333 3333343434 3333332232 3233233332 3333333  
7540 p15 4421512443 4344343434 3432241123 4433444442 3234333  
6291 p15 3333243432 4342223224 2433342334 3433432243 3332333  
0104 p15 3515554333 5325353443 3233332232 3233223333 3333333

Appendix C

9141 p15 1155333433 3353333335 3533343334 3533353333 3333333  
 3074 p15 3333432332 2334343344 2433342232 3233233232 3234333  
 ---- p15 3243232323 3332323233 1433333244 3433432254 3432333  
 1576 p15 3422421332 1333343332 3433232122 3233233342 3234334  
 6825 p15 3332445332 2334333234 2533332333 4433343343 3433333  
 3209 p15 2223422433 2334343443 4431442122 2424354344 4244325  
 6408 p15 2522123443 4331313121 1532345424 3433531154 3431333  
 6284 p15 3234344333 3344343443 4333322323 3333344323 3333333  
 ---- p15 3322422333 2344344444 3442432222 3224244432 3244335  
 ---- p15 3244242333 2343343334 3433333233 3333443342 3233333  
 ---- p15 3233322333 3333333344 2233332233 3233232232 3233333  
 ---- p15 3114342233 3342223213 2534334435 5533541153 2331333  
 ---- p15 4422422442 2444243444 3442442232 2224244343 4224334  
 6875 p15 3243131442 3341213112 1432324433 4433521144 3431333  
 ---- p15 3433534441 2455252552 4433314333 3333333333 3333333  
 ---- p15 2324242323 4251424242 5142444343 4333534535 3434345  
 ---- p15 3423243233 3433433334 2533332434 3433452132 3332423  
 5065 p15 4422444442 2333244323 2432443322 2224443234 3432323  
 0579 p15 4333422433 3334343334 3433332232 2223243343 3334332  
 6648 b15 3332422333 2324243444 2223332232 3223224442 3234334  
 3859 b15 3333432443 2333333343 2433442223 4224233441 3133334  
 7769 b15 3333342342 3333323224 1232333314 4434323134 3333333  
 1990 b15 4333432343 2344443445 3343331222 2134344432 3334334  
 7535 b15 3333422333 2344343444 3433334234 3433433133 3432333  
 9883 b15 3322343443 3343333334 2433333334 3433443243 3333333

Appendix C

1783 b15 3442421333 2344343444 3432431132 3224214332 3224324  
4783 b15 2343144555 4341312112 1514235534 4433451154 3431333  
5336 b15 2242253221 4151111134 1514244444 4442451152 1421244  
7644 b15 3322421543 2234344324 2442442245 3342532342 3244423  
9254 b15 3333232333 2332333234 2333333333 3433432233 3332333  
1884 b15 3223244223 2342323214 3513133433 3243251151 2231333  
7876 b15 2243143321 2321413114 1432224535 4523521132 2231333  
4642 b15 3333432321 1232211111 1211224555 5451451134 1511333  
9549 b15 4333442344 2332333324 3323333234 3324434254 3334333  
3441 b15 3422422324 2344243244 2433442222 4222244242 3233244  
5731 b15 3244452342 2344444444 3442442222 2224224442 4244424  
5863 b15 4421432243 1344344444 2522331111 3233223332 3234234  
2327 b15 4333433442 3244343344 2433332234 3424433244 3334333  
5510 b15 3333353443 3343333334 3433333333 4333433343 3333334  
7133 b15 4422422443 2234244444 2342442222 3224224442 4244324  
6050 b15 3244233343 2323434444 3243241231 4223223452 3234234  
2242 b15 3322242342 2252223224 1414244234 4433442152 2432242  
5812 b15 3332432443 2354344444 3542452122 2224254432 4244324  
3638 b15 3323422342 2344343445 2423332222 4233234242 3234334  
8046 b15 2332452212 2343233434 1524152122 5433322453 2223224  
2861 b15 3344452443 1343443444 3442342222 3322243353 3234334  
5493 b15 3422434342 2344243444 2444342222 2423423152 4233424  
8740 b15 4333531442 2355354554 3543441111 2224225551 4135335  
5028 b15 3333531113 1545354554 3133331131 3133155431 3135333  
7317 b15 3333331331 2343333335 3534331243 4233253451 3233333

Appendix C

3534 b15 4344242445 3353434234 3542443424 2414453344 4343424  
2014 b15 5322532555 2235555555 2351551111 2115115521 5155515  
4530 b15 4513531443 1325455555 5531451121 2115125531 5145224  
8245 b15 3322521315 4344444422 5333331134 4233244222 4234334  
8992 b15 4333533433 3335353455 4343331212 3233324532 3234334  
2533 b15 4411521355 2315353554 3131441112 2315225522 5255335  
1792 b15 4333432444 3324444444 4242422222 2224224332 4244334  
1371 b15 3423432443 2234444334 3442342222 2223223242 4233343  
---- b15 4333422443 1344344442 3542432212 2423433442 3244334  
---- b15 3332452332 2344343344 2433432232 3433243232 3234333  
---- b15 3243531542 1325355554 3432251111 4124135542 3235334

**Appendix C**

----- June 5 -----										----- June 15 -----										
---- FL/I ----					---- Basic ----					---- FL/I ----					---- Basic ----					
1 +	0	1	37	1	0	2	0	39	2	1	1	3	25	7	0	0	4	25	12	1
2 +	10	4	24	10	1	2	4	33	4	1	2	8	15	8	3	0	6	28	7	1
3 -	1	8	25	5	0	0	8	33	3	0	2	14	15	4	1	2	12	19	9	0
4 -	0	7	28	4	0	0	8	33	3	0	1	9	19	5	2	2	14	21	4	0
5 +	3	5	11	19	1	1	7	14	12	10	2	9	5	16	4	2	6	4	22	8
6 -	0	4	33	2	0	2	0	37	5	0	2	12	13	8	1	0	10	18	8	6
7 -	4	10	20	5	0	3	9	31	1	0	5	20	5	5	1	9	23	7	3	0
8 +	0	3	23	9	4	1	2	23	16	2	0	3	22	11	0	1	4	19	14	4
9 +	0	3	29	5	2	3	1	27	12	1	0	2	25	9	0	3	5	7	24	3
10 +	3	3	27	5	1	4	2	31	4	3	1	9	24	1	1	4	10	20	3	5
11 -	2	6	24	6	1	3	13	22	4	2	2	11	12	9	2	7	25	7	3	0
12 -	1	2	35	0	1	1	0	41	0	2	0	1	32	3	0	1	7	33	0	1
13 -	0	4	10	21	4	2	5	15	15	7	1	2	13	15	5	1	6	10	20	5
14 +	0	5	25	7	2	1	5	25	9	4	3	7	13	11	2	3	5	9	19	7
15 +	1	1	32	2	3	0	4	33	5	2	0	8	25	3	0	1	7	23	9	1
16 +	3	2	26	7	1	1	6	25	8	4	3	6	8	16	3	4	3	9	19	7
17 +	0	2	28	8	1	0	2	34	7	1	0	1	31	4	0	2	1	23	13	3
18 +	1	7	26	5	0	1	8	26	5	4	3	8	12	12	1	4	6	8	17	6
19 +	0	5	22	10	2	0	6	25	7	6	2	6	12	14	2	4	5	8	18	7
20 +	2	4	4	26	3	1	4	0	33	6	1	5	8	20	2	1	3	1	31	6
21 +	2	9	26	1	1	3	13	25	3	0	6	14	11	4	1	7	13	18	2	2
22 -	0	2	6	21	10	2	3	5	19	15	1	5	3	18	9	2	5	07	17	11
23 +	0	0	32	5	2	0	2	35	4	3	0	1	32	3	0	5	5	17	14	1

Appendix D

24 - 2 3 32 2 0 3 12 26 3 0 1 10 23 2 0 4 14 18 6 0  
 25 + 0 1 28 10 1 0 2 31 9 2 0 3 27 6 0 2 7 18 14 1  
 26 + 0 3 27 9 0 0 2 29 11 2 2 2 20 12 0 0 3 20 14 5  
 27 - 2 11 17 9 0 8 15 12 7 2 1 18 7 8 2 13 16 7 5 1  
 28 - 5 14 17 3 0 9 19 11 3 2 5 13 11 6 1 11 21 4 3 3  
 29 - 1 5 30 3 0 2 8 33 0 1 0 10 24 2 0 8 14 16 3 1  
 30 - 2 7 23 7 0 5 10 18 10 1 0 15 9 8 4 7 17 5 10 3  
 31 - 0 6 30 3 0 1 4 38 1 0 0 5 23 5 2 0 12 15 13 2  
 32 - 1 9 10 16 3 6 16 9 11 2 0 12 6 15 3 5 17 05 14 1  
 33 - 0 4 33 2 0 1 9 34 0 0 0 8 27 1 0 4 18 16 3 1  
 34 + 0 1 32 6 0 1 1 39 2 1 0 0 29 7 0 1 4 21 13 3  
 35 - 2 9 9 16 3 5 17 5 15 2 0 10 9 13 4 4 19 4 13 2  
 36 + 3 10 12 9 5 4 16 11 9 4 0 5 12 13 6 2 15 10 7 8  
 37 + 2 7 25 4 1 1 7 28 4 4 5 7 16 8 0 5 4 15 12 6  
 38 + 3 9 18 9 0 8 7 15 11 3 6 10 16 3 1 9 9 7 11 6  
 39 - 1 0 22 16 0 0 2 30 10 2 1 1 18 11 5 0 3 14 14 11  
 40 - 5 18 13 3 0 8 24 7 5 0 0 16 12 7 1 7 23 5 6 0  
 41 + 2 2 34 1 0 0 0 40 3 1 0 1 33 2 0 2 4 24 9 3  
 42 - 0 8 25 6 0 4 10 26 4 0 0 13 14 9 0 5 24 8 4 1  
 43 + 0 4 31 3 1 0 1 38 3 1 0 2 31 3 0 1 3 28 8 2  
 44 + 1 5 27 5 1 1 6 27 6 4 3 9 12 12 0 5 3 10 18 6  
 45 + 0 3 36 0 0 0 0 42 1 1 0 1 34 1 0 0 7 30 3 1  
 46 - 1 1 34 3 0 1 0 43 0 0 0 4 31 1 0 1 9 29 4 0  
 47 + 0 0 35 3 10 0 0 34 8 2 0 1 28 4 3 0 1 14 25 3

Appendix D

- - - - - J u n e 5 - - - - -

class	number of					total	var	+	z-	95%ci + or -		
	respondants									mean	-	value
	1	2	3	4	5							
1 pl/1	0	1	37	1	0	39	3.00	0.05	+	0.00	0.00	0.18
basic	2	0	39	2	1	44	3.00	0.33				
2 pl/1	10	4	24	10	1	49	2.76	1.15	+	1.08	-0.20	0.36
basic	2	4	33	4	1	44	2.95	0.46				
3 pl/1	1	8	25	5	0	39	2.87	0.43	-	0.11	-0.01	0.25
basic	0	8	33	3	0	44	2.89	0.24				
4 pl/1	0	7	28	4	0	39	2.92	0.28	-	-0.32	0.04	0.22
basic	0	8	33	3	0	44	2.89	0.24				
5 pl/1	3	5	11	19	1	39	3.26	0.99	+	1.17	-0.27	0.45
basic	1	7	14	12	10	44	3.52	1.19				
6 pl/1	0	4	33	2	0	39	2.95	0.16	-	0.71	-0.07	0.20
basic	2	0	37	5	0	44	3.02	0.30				
7 pl/1	4	10	20	5	0	39	2.67	0.70	-	0.09	-0.02	0.32
basic	3	9	31	1	0	44	2.68	0.41				
8 pl/1	0	3	23	9	4	39	3.36	0.60	+	0.03	-0.00	0.33

Appendix E

page 148



	basic	1	2	23	16	2	44	3.36	0.56			
9	pl/1	0	3	29	5	2	39	3.15	0.40	+	0.03	-0.01 0.31
	basic	3	1	27	12	1	44	3.16	0.65			
10	pl/1	3	3	27	5	1	39	2.95	0.63	+	0.28	-0.05 0.36
	basic	4	2	31	4	3	44	3.00	0.79			
11	pl/1	2	6	24	6	1	39	2.95	0.63	-	-1.07	0.20 0.36
	basic	3	13	22	4	2	44	2.75	0.80			
12	pl/1	1	2	35	0	1	39	2.95	0.26	-	0.85	-0.10 0.22
	basic	1	0	41	0	2	44	3.05	0.28			
13	pl/1	0	4	10	21	4	39	3.64	0.66	-	-0.91	0.19 0.40
	basic	2	5	15	15	7	44	3.45	1.09			
14	pl/1	0	5	25	7	2	39	3.15	0.50	+	0.43	-0.07 0.34
	basic	1	5	25	9	4	44	3.23	0.74			
15	pl/1	1	1	32	2	3	39	3.13	0.48	+	-0.10	0.01 0.28
	basic	0	4	33	5	2	44	3.11	0.38			
16	pl/1	3	2	26	7	1	39	3.03	0.66	+	0.85	-0.16 0.36
	basic	1	6	25	8	4	44	3.18	0.76			
17	pl/1	0	2	28	8	1	39	3.21	0.33	+	-0.38	0.05 0.24
	basic	0	2	34	7	1	44	3.16	0.28			
18	pl/1	1	7	26	5	0	39	2.90	0.41	+	1.02	-0.17 0.33

Appendix E

	basic	1	8	26	5	4	44	3.07	0.76				
19	pl/1	0	5	22	10	2	39	3.23	0.55 +	0.36	-0.06	0.35	
	basic	0	6	25	7	6	44	3.30	0.77				
20	pl/1	2	4	4	26	3	39	3.62	0.93 +	1.36	-0.27	0.39	
	basic	1	4	0	33	6	44	3.89	0.71				
21	pl/1	2	9	26	1	1	39	2.74	0.51 +	-0.68	0.11	0.31	
	basic	3	13	25	3	0	44	2.64	0.52				
22	pl/1	0	2	6	21	10	39	4.00	0.63 -	-0.22	0.05	0.40	
	basic	2	3	5	19	15	44	3.95	1.16				
23	pl/1	0	0	32	5	2	39	3.23	0.29 +	-0.39	0.05	0.25	
	basic	0	2	35	4	3	44	3.18	0.38				
24	pl/1	2	3	32	2	0	39	2.87	0.33 -	-1.51	0.21	0.28	
	basic	3	12	26	3	0	44	2.66	0.51				
25	pl/1	0	1	28	10	1	40	3.27	0.31 +	-0.20	0.02	0.25	
	basic	0	2	31	9	2	44	3.25	0.38				
26	pl/1	0	3	27	9	0	39	3.15	0.29 +	1.10	-0.14	0.25	
	basic	0	2	29	11	2	44	3.30	0.40				
27	pl/1	2	11	17	9	0	39	2.85	0.71 -	-1.40	0.30	0.42	
	basic	8	15	12	7	2	44	2.55	1.23				
28	pl/1	5	14	17	3	0	39	2.46	0.68 -	-0.70	0.14	0.40	

Appendix E

	basic	9	19	11	3	2	44	2.32	1.06				
29	pl/1	1	5	30	3	0	39	2.90	0.30	-	-0.95	0.12	0.26
	basic	2	8	33	0	1	44	2.77	0.41				
30	pl/1	2	7	23	7	0	39	2.90	0.57	-	-0.41	0.08	0.38
	basic	5	10	18	10	1	44	2.82	0.99				
31	pl/1	0	6	30	3	0	39	2.92	0.23	-	-0.36	0.04	0.20
	basic	1	4	38	1	0	44	2.89	0.20				
32	pl/1	1	9	10	16	3	39	3.28	1.00	-	-2.47	0.58	0.46
	basic	6	16	9	11	2	44	2.70	1.28				
33	pl/1	0	4	33	2	0	39	2.95	0.16	-	-2.05	0.20	0.19
	basic	1	9	34	0	0	44	2.75	0.24				
34	pl/1	0	1	32	6	0	39	3.13	0.17	+	-1.05	0.11	0.20
	basic	1	1	39	2	1	44	3.02	0.26				
35	pl/1	2	9	9	16	3	39	3.23	1.13	-	-1.69	0.41	0.48
	basic	5	17	5	15	2	44	2.82	1.36				
36	pl/1	3	10	12	9	5	39	3.08	1.34	+	-0.93	0.24	0.50
	basic	4	16	11	9	4	44	2.84	1.30				
37	pl/1	2	7	25	4	1	39	2.87	0.59	+	1.11	-0.20	0.35
	basic	1	7	28	4	4	44	3.07	0.72				
38	pl/1	3	9	18	9	0	39	2.85	0.77	+	0.08	-0.02	0.45

Appendix E

	basic	8	7	15	11	3	44	2.86	1.42				
39	pl/1	1	0	22	16	0	39	3.36	0.39 -	-0.63	0.09	0.27	
	basic	0	2	30	10	2	44	3.27	0.39				
40	pl/1	5	18	13	3	0	39	2.36	0.66 -	-0.83	0.15	0.36	
	basic	8	24	7	5	0	44	2.20	0.77				
41	pl/1	2	2	34	1	0	39	2.87	0.27 +	2.37	-0.24	0.20	
	basic	0	0	40	3	1	44	3.11	0.15				
42	pl/1	0	8	25	6	0	39	2.95	0.37 -	-1.76	0.27	0.30	
	basic	4	10	26	4	0	44	2.68	0.59				
43	pl/1	0	4	31	3	1	39	3.03	0.29 +	0.62	-0.07	0.21	
	basic	0	1	38	3	1	43	3.09	0.18				
44	pl/1	1	5	27	5	1	39	3.00	0.47 +	0.81	-0.14	0.33	
	basic	1	6	27	6	4	44	3.14	0.73				
45	pl/1	0	3	36	0	0	39	2.92	0.07 +	2.19	-0.15	0.13	
	basic	0	0	42	1	1	44	3.07	0.11				
46	pl/1	1	1	34	3	0	39	3.00	0.21 -	-0.53	0.05	0.17	
	basic	1	0	43	0	0	44	2.95	0.09				
47	pl/1	0	0	35	3	10	48	3.48	0.68 +	-1.43	0.21	0.28	
	basic	0	0	34	8	2	44	3.27	0.30				

Appendix E

class	number of					total	var	+	z-	95%ci + or -		
	respondants									mean	-	value
	1	2	3	4	5							
1 pl/1	1	3	25	7	0	36	3.06	0.40	+	1.25	-0.18	0.29
basic	0	4	25	12	1	42	3.24	0.43				
2 pl/1	2	8	15	8	3	36	3.06	1.03	+	0.08	-0.02	0.38
basic	0	6	28	7	1	42	3.07	0.41				
3 pl/1	2	14	15	4	1	36	2.67	0.74	-	0.87	-0.17	0.38
basic	2	12	19	9	0	42	2.83	0.68				
4 pl/1	1	9	19	5	2	36	2.94	0.74	-	-1.56	0.29	0.36
basic	2	14	21	4	0	41	2.66	0.53				
5 pl/1	2	9	5	16	4	36	3.31	1.30	+	1.42	-0.36	0.50
basic	2	6	4	22	8	42	3.67	1.20				
6 pl/1	2	12	13	8	1	36	2.83	0.89	-	1.86	-0.40	0.43
basic	0	10	18	8	6	42	3.24	0.97				
7 pl/1	5	20	5	5	1	36	2.36	0.98	-	-1.28	0.27	0.41
basic	9	23	7	3	0	42	2.10	0.67				

Appendix E

8	pl/1	0	3	22	11	0	36	3.22	0.35 +	0.94	-0.16	0.33
	basic	1	4	19	14	4	42	3.38	0.78			
9	pl/1	0	2	25	9	0	36	3.19	0.28 +	1.41	-0.26	0.36
	basic	3	5	7	24	3	42	3.45	1.08			
10	pl/1	1	9	24	1	1	36	2.78	0.46 +	0.51	-0.10	0.40
	basic	4	10	20	3	5	42	2.88	1.18			
11	pl/1	2	11	12	9	2	36	2.94	1.03 -	-3.86	0.80	0.41
	basic	7	25	7	3	0	42	2.14	0.61			
12	pl/1	0	1	32	3	0	36	3.06	0.11 -	-2.11	0.22	0.21
	basic	1	7	33	0	1	42	2.83	0.34			
13	pl/1	1	2	13	15	5	36	3.58	0.82 -	-0.28	0.06	0.42
	basic	1	6	10	20	5	42	3.52	0.94			
14	pl/1	3	7	13	11	2	36	3.06	1.08 +	1.87	-0.46	0.48
	basic	3	5	9	19	7	43	3.51	1.26			
15	pl/1	0	8	25	3	0	36	2.86	0.29 +	1.24	-0.19	0.30
	basic	1	7	23	9	1	41	3.05	0.60			
16	pl/1	3	6	8	16	3	36	3.28	1.23 +	0.96	-0.25	0.50
	basic	4	3	9	19	7	42	3.52	1.33			
17	pl/1	0	1	31	4	0	36	3.08	0.14 +	1.73	-0.25	0.28
	basic	2	1	23	13	3	42	3.33	0.72			

Appendix E

18	pl/1	3	8	12	12	1	36	3.00	1.03 +	1.45	-0.37	0.49
	basic	4	6	8	17	6	41	3.37	1.44			
19	pl/1	2	6	12	14	2	36	3.22	0.98 +	0.93	-0.23	0.48
	basic	4	5	8	18	7	42	3.45	1.42			
20	pl/1	1	5	8	20	2	36	3.47	0.83 +	2.19	-0.43	0.39
	basic	1	3	1	31	6	42	3.90	0.67			
21	pl/1	6	14	11	4	1	36	2.44	1.00 +	0.25	-0.06	0.44
	basic	7	13	18	2	2	42	2.50	0.99			
22	pl/1	1	5	3	18	9	36	3.81	1.13 -	-0.37	0.09	0.49
	basic	2	5	7	17	11	42	3.71	1.28			
23	pl/1	0	1	32	3	0	36	3.06	0.11 +	-0.19	0.03	0.33
	basic	5	5	17	14	1	42	3.02	1.05			
24	pl/1	1	10	23	2	0	36	2.72	0.38 -	-0.62	0.10	0.33
	basic	4	14	18	6	0	42	2.62	0.73			
25	pl/1	0	3	27	6	0	36	3.08	0.25 +	0.22	-0.04	0.31
	basic	2	7	18	14	1	42	3.12	0.79			
26	pl/1	2	2	20	12	0	36	3.17	0.60 +	1.86	-0.33	0.35
	basic	0	3	20	14	5	42	3.50	0.65			
27	pl/1	1	18	7	8	2	36	2.78	1.03 -	-2.57	0.61	0.47
	basic	13	16	7	5	1	42	2.17	1.17			

Appendix E

28	pl/1	5	13	11	6	1	36	2.58	1.05	-	-1.61	0.39	0.48
	basic	11	21	4	3	3	42	2.19	1.28				
29	pl/1	0	10	24	2	0	36	2.78	0.29	-	-2.14	0.37	0.34
	basic	8	14	16	3	1	42	2.40	0.93				
30	pl/1	0	15	9	8	4	36	3.03	1.11	-	-1.49	0.38	0.51
	basic	7	17	5	10	3	42	2.64	1.50				
31	pl/1	0	5	23	5	2	35	3.11	0.52	-	0.03	-0.00	0.36
	basic	0	12	15	13	2	42	3.12	0.79				
32	pl/1	0	12	6	15	3	36	3.25	1.05	-	-2.10	0.51	0.48
	basic	5	17	5	14	1	42	2.74	1.27				
33	pl/1	0	8	27	1	0	36	2.81	0.22	-	-1.98	0.31	0.30
	basic	4	18	16	3	1	42	2.50	0.74				
34	pl/1	0	0	29	7	0	36	3.19	0.16	+	0.79	-0.12	0.29
	basic	1	4	21	13	3	42	3.31	0.71				
35	pl/1	0	10	9	13	4	36	3.31	1.02	-	-2.23	0.54	0.48
	basic	4	19	4	13	2	42	2.76	1.31				
36	pl/1	0	5	12	13	6	36	3.56	0.88	+	-1.87	0.46	0.48
	basic	2	15	10	7	8	42	3.10	1.50				
37	pl/1	5	7	16	8	0	36	2.75	0.94	+	2.00	-0.49	0.48
	basic	5	4	15	12	6	42	3.24	1.41				

Appendix E



38	pl/1	6	10	16	3	1	36	2.53	0.94 +	1.40	-0.38	0.53
	basic	9	9	7	11	6	42	2.90	1.94			
39	pl/1	1	1	18	11	5	36	3.50	0.77 -	1.40	-0.29	0.40
	basic	0	3	14	14	11	42	3.79	0.86			
40	pl/1	0	16	12	7	1	36	2.81	0.73 -	-2.71	0.54	0.39
	basic	7	23	6	6	0	42	2.26	0.83			
41	pl/1	0	1	33	2	0	36	3.03	0.08 +	0.96	-0.14	0.28
	basic	2	4	24	9	3	42	3.17	0.78			
42	pl/1	0	13	14	9	0	36	2.89	0.62 -	-2.91	0.56	0.37
	basic	5	24	8	4	1	42	2.33	0.81			
43	pl/1	0	2	31	3	0	36	3.03	0.14 +	1.08	-0.14	0.25
	basic	1	3	28	8	2	42	3.17	0.53			
44	pl/1	3	9	12	12	0	36	2.92	0.94 +	2.00	-0.49	0.48
	basic	5	3	10	18	6	42	3.40	1.42			
45	pl/1	0	1	34	1	0	36	3.00	0.06 +	-0.49	0.05	0.20
	basic	0	7	30	3	1	41	2.95	0.35			
46	pl/1	0	4	31	1	0	36	2.92	0.14 -	-0.71	0.08	0.22
	basic	1	9	29	4	0	43	2.84	0.38			
47	pl/1	0	1	28	4	3	36	3.25	0.42 +	3.08	-0.45	0.29
	basic	0	1	14	25	3	43	3.70	0.41			

Appendix E



3218 b 5 33222335332332332435333322323332532323333333  
3441 b 5 32332433432343333224243333223432334322333432333  
3441 b 15 3422422324234424324424334422224222244242323244  
3534 b 5 32432433333343333334243343443434234433343333333  
3534 b 15 43442424453353434234354244342424144533444343424  
3638 b 5 33334333333344333345243333223232332323234333  
3638 b 15 33234223422344343445242333222242332342423234334  
3859 b 5 3223333443233333334353445112132232433423233334  
3859 b 15 33334324432333333343243344222342242334413133334  
4246 b 5 34224324452344444452444144222232333333333333333  
4530 b 5 34225114431325354555355155111131331255313135333  
4530 b 15 4513531443132545555553145112121151255315145224  
4642 b 5 1333333113335323332435333332333333311313322333  
4642 b 15 33334323211232211111121122455554514511341511333  
4783 b 15 23431445554341312112151423553444334511543431333  
4785 b 5 4244143434415242222253233443434334421443432333  
5028 b 5 53335332131535353554313342111121311553313135333  
5028 b 15 33335311131545354554313333113131331554313135333  
5336 b 5 3333233331535332313435343355353535133323333333  
5336 b 15 22422532214151111134151424444444424511521421244  
5493 b 5 34234334332343333444343424323233332232433334334  
5493 b 15 34224343422344243444244434222224234231524233424  
5510 b 5 3333433333333333334343333333333333333333333333  
5510 b 15 3333353443334333333434333333343334333433333334  
5731 b 5 31223434412334342444443244222232242244424244334

Appendix F

5731 b 15 3244452342234444444434424422222242244424244424  
5812 b 5 33335324352334343444354243212232342244323334334  
5812 b 15 33324324432354344444354245212222242544324244324  
5863 b 5 34224132431344244454253222113221331244523134435  
5863 b 15 44214322431344344444252233111132332233323234234  
6050 b 5 33325334433343343334343334123232232233323333333  
6050 b 15 32442333432323434444324324123142232234523234234  
6648 b 5 33334323333323333334222333223232332234323233333  
6648 b 15 33324223332324243444222333223232232244423234334  
7133 b 5 3332432444334424434424323422222232234424243334  
7133 b 15 44224224432234244444234244222232242244424244324  
7229 b 5 3333433443335333333353334223334332434423333333  
7317 b 5 333333333133333333353333333333331331234333333333  
7317 b 15 33333313312343333335353433124342332534513233333  
7535 b 5 33335344333333333234353333423333332433343333333  
7535 b 15 33334223332344343444343333423434334331333432333  
7624 b 5 33335324422343244332143233222334234432323343334  
7644 b 5 3333333334333333333433333333333333533333333333  
7644 b 15 33224215432234344324244244224533425323423244423  
7769 b 5 3333233334322323224243333433335334321343332333  
7769 b 15 33333423423333323224123233331444343231343333333  
7876 b 5 33332333313321313225153233553434334221313332333  
7876 b 15 22431433212321413114143222453545235211322231333  
7882 b 5 31443424444343424234144244433424234421423441334  
8046 b 5 33333333323343333334343234313332334233223333333

Appendix F

8046 b 15 23324522122343233434152415212254333224532223224  
8245 b 15 33225213154344444422533333113442332442224234334  
8740 b 5 333333333333433333343533333233333423332333333  
8740 b 15 43335314422355354554354344111122242255514135335  
8992 b 5 333343331333333333432333322333233213432333333  
8992 b 15 43335334333335353455434333121232333245323234334  
9254 b 5 3333233433333232323424333343343433432232333333  
9254 b 15 3333232333233233323423333333334334322333332333  
9549 b 5 33334333332343333334343333223332332431313334333  
9549 b 15 43334423442332333324332333323433244342543334333  
9601 b 5 35345313335534533332255243122332322134514233333  
9883 b 5 33333333333353333334353333333434334533323333333  
9883 b 15 3322343443334333333424333333343433443243333333  
0104 p 5 3532432333334333333324333322333535131323233333  
0104 p 15 3515554333532535344332333322323233223333333333  
0123 p 5 34431434344342314224253343434323334521413222243  
0142 p 5 333343233333333333334333333223232332233423333333  
0142 p 15 43334323333333334343433333322323233233323333333  
0163 p 5 33334325443344334322244444434443344324342444345  
0323 p 5 33343332334243332233243333433234333222323332333  
0323 p 15 3233233333432223223123334333343342223133323333  
0579 p 5 33334344333344343344343343322334332444333234333  
0579 p 15 43334224333334343334343333223222232433433334332  
0671 p 5 322223333234333334243333223232332433323233333  
1234 p 5 33334333333344333244343234442334324232423333333

Appendix F

1234 p 15 33334323334343343244242433233234334432343434333  
1576 p 5 33334313332333333342343333213332332333323333333  
1576 p 15 34224213321333343332343323212232332333423234334  
2391 p 5 333333333333333333434333333333334433333333333  
3074 p 5 3333433333333333334343333223332333333333333333  
3074 p 15 333343233223343433442433342232323323323234333  
3075 p 5 33324243332344344334234323233432444433324422323  
3209 p 5 32432313432344343444344244122224233444423344334  
3209 p 15 22234224332334343443443144212224243543444244325  
4093 p 5 44223325324343535241153233333444334322423423333  
4093 p 15 4333231432534242422243244543554244511523332334  
5065 p 15 44224444422333244323243244332222244432343432323  
5310 p 5 3422334334435333333425333333343433443132343333  
5616 p 5 33442343223342224224223332443434334222433332333  
5616 p 15 32442343334342313124153333453535334421343432333  
5681 p 5 33334333333343333334343333323334334233323333333  
5681 p 15 33334323332344343444243333223234334232323234333  
5720 p 5 34334323233324444444343334323224334234423333334  
5842 p 5 24244322211545352454343342213121331154423235333  
5852 p 5 342243224432434433323433443234344432433433333  
5852 p 15 43224222434353333343223224422424244532523233224  
6284 p 5 33333323332343333344343333223332234433433333333  
6284 p 15 3234344333334434344343333223233333443233333333  
6291 p 5 33333334323343333334343333443334333332333333333  
6291 p 15 33332434324342223224243334233434334322433332333

Appendix F

6408 p 5 33331325534342513121154143422335245212443331333  
6408 p 15 25221234434331313121153234542434335311543431333  
6445 p 5 3423233333335333333435333332343433553333333333  
6445 p 15 34234223333353343444253333333534334533423333333  
6617 p 5 3322331433232333344243333313334334332331333333  
6825 p 15 33324453322334333234253333233344333433433433333  
6875 p 5 3333333533333333333435333333333333334433333333  
6875 p 15 32431314423341213112143232443344335211443431333  
7449 p 5 333343243333333333343433333333333433332333333  
7449 p 15 34334323442334343444343233322232333343323234333  
7506 p 5 3333333333332333333233333333333332233323233333  
7540 p 5 33334334333333333334343334333333332334323333333  
7540 p 15 44215124434344343434343224112344334444423234333  
8352 p 5 3333233333342323235353333433334333224333433333  
8352 p 15 33332223333343333335153333443434333413333432333  
8515 p 5 34145243551143534455555154111125241513421354213  
8703 p 5 32334334333324343344343333223234332234323433333  
9141 p 5 33333333333353333335353332333333334533313333333  
9141 p 15 11553334333353333335353334333435333533333333333  
9361 p 5 32434314413353134333354443333434334332443333333  
9423 p 5 3333433333333333333433333333333433333313333333  
9423 p 15 33333323333343333334233333433434334433323332333  
9456 p 5 34234233415335343444333344212232342443433234334  
9564 p 15 332242233323333433342233332232323333323232333  
9568 p 5 33334333333333333334333333223332332333313333333

Appendix F

9687 p 5 33333333333433333343433333333233313332333333  
9687 p 15 3522411345434433345434333213232232244333344333

Appendix F

page 164



----- pl1 class -----

question	plus	minus	total	z
01 +	4	3	7	0.000
02 +	5	7	12	-0.866
03 -	6	3	9	0.667
04 -	5	4	9	-0.000
05 +	5	5	10	-0.316
06 -	9	3	12	1.443
07 -	11	4	15	1.549
08 +	3	4	7	-0.756
09 +	5	2	7	0.756
10 +	3	4	7	-0.756
11 -	4	9	13	-1.664
12 -	0	2	2	-2.121
13 -	5	4	9	-0.000
14 +	8	6	14	0.267
15 +	1	6	7	-2.268
16 +	11	6	17	0.970
17 +	1	2	3	-1.155
18 +	9	3	12	1.443
19 +	7	6	13	-0.000
20 +	2	3	5	-0.894

Appendix G

21 +	4	11	15	-2.066 *
22 -	6	1	7	1.512
23 +	0	3	3	-2.309 *
24 -	5	2	7	0.756
25 +	1	5	6	-2.041 *
26 +	7	3	10	0.949
27 -	7	6	13	-0.000
28 -	6	7	13	-0.555
29 -	5	2	7	0.756
30 -	8	9	17	-0.485
31 -	2	4	6	-1.225
32 -	5	5	10	-0.316
33 -	4	2	6	0.408
34 +	4	1	5	0.894
35 -	8	4	12	0.866
36 +	12	1	13	2.774 *
37 +	4	5	9	-0.667
38 +	3	9	12	-2.021
39 -	5	8	13	-1.109
40 -	3	9	12	-2.021 *
41 +	1	0	1	0.000
42 -	8	6	14	0.267
43 +	2	1	3	0.000
44 +	8	5	13	0.555
45 +	0	1	1	-2.000 *

Appendix G

page 166

46 -	2	0	2	0.707
47 +	4	1	5	0.894

Appendix G

page 167

- - - - - basic class - - - - -

question	plus	minus	total	z
1 +	13	4	17	1.940 *
2 +	9	4	13	1.109
3 -	9	7	16	0.250
4 -	11	5	16	1.250
5 +	12	8	20	0.671
6 -	13	8	21	0.873
7 -	21	2	23	3.753 *
8 +	10	9	19	-0.000
9 +	18	5	23	2.502 *
10 +	6	10	16	-1.250
11 -	20	0	20	4.249 *
12 -	7	0	7	2.268 *
13 -	7	12	19	-1.376
14 +	12	5	17	1.455
15 +	9	6	15	0.516
16 +	14	5	19	1.835 *
17 +	10	3	13	1.564 *
18 +	12	4	16	1.750 *
19 +	10	7	17	0.485
20 +	3	3	6	-0.408

Appendix G

page 168

21 +	5	12	17	-1.940
22 -	12	6	18	1.179
23 +	11	9	20	0.224
24 -	8	3	11	1.206
25 +	8	9	17	-0.485
26 +	11	5	16	1.250
27 -	14	3	17	2.425 *
28 -	9	3	12	1.443
29 -	13	6	19	1.376
30 -	13	9	22	0.640
31 -	10	14	24	-1.021
32 -	7	9	16	-0.750
33 -	14	5	19	1.835 *
34 +	13	5	18	1.650 *
35 -	8	9	17	-0.485
36 +	11	6	17	0.970
37 +	10	6	16	0.750
38 +	9	8	17	-0.000
39 -	2	18	20	-3.801
40 -	9	7	16	0.250
41 +	8	6	14	0.267
42 -	13	4	17	1.940 *
43 +	6	4	10	0.316
44 +	13	7	20	1.118
45 +	4	7	11	-1.206

Appendix G

46 -	9	4	13	1.109
47 +	16	2	18	3.064 #

Appendix G

page 170

----- p 1 1 -----

student	plus	minus	total	% plus	z-value
!	!	!	!	!	!
!	!	!	!	!	!
0104	15	5	20	0.750	2.01 *
0142	5	0	5	1.000	1.79 *
0323	13	12	25	0.520	-0.00
0579	8	9	17	0.471	-0.49
1234	9	9	18	0.500	-0.24
1576	10	5	15	0.667	1.03
3074	11	3	14	0.786	1.87 *
3209	13	9	22	0.591	0.64
4093	12	19	31	0.387	-1.44
5616	6	13	19	0.316	-1.84
5681	10	2	12	0.833	2.02 *
5852	13	13	26	0.500	-0.20
6284	10	9	19	0.526	-0.00
6291	4	13	17	0.235	-2.43
6408	7	18	25	0.280	-2.40
6445	7	5	12	0.583	0.29
6875	4	25	29	0.138	-4.09
7449	17	1	18	0.944	3.54 *
7540	18	8	26	0.692	1.77 *
8352	5	7	12	0.417	-0.87

Appendix H

page 171

9141	3	7	10	0.300	-1.58
9423	2	8	10	0.200	-2.21
9687	20	3	23	0.870	3.30 *

Appendix H

page 172



- - - - - b a s i c - - - - -

student	plus	minus	total	% plus	z-value
1371	20	3	23	0.870	3.34 *
1783	22	6	28	0.786	2.83 *
1792	20	5	25	0.800	2.80 *
1884	8	16	24	0.333	-1.84 *
1990	24	1	25	0.960	4.40 *
2242	5	18	23	0.217	-2.92 *
2327	8	5	13	0.615	0.55
2533	21	3	24	0.875	3.47 *
2861	22	5	27	0.815	3.08 *
3441	21	8	29	0.724	2.23 *
3534	23	5	28	0.821	3.21 *
3638	9	5	14	0.643	0.80
3859	10	9	19	0.526	-0.00
4530	14	5	19	0.737	1.84 *
4642	11	25	36	0.306	-2.50 *
5028	5	6	11	0.455	-0.60
5336	12	25	37	0.324	-2.30 *
5493	15	10	25	0.600	0.80
5510	4	5	9	0.444	-0.67
5731	11	5	16	0.687	1.25

Appendix H

5812	11	3	14	0.786	1.87 *
5863	10	10	20	0.500	-0.22
6050	13	10	23	0.565	0.42
6648	10	3	13	0.769	1.66 *
7133	14	3	17	0.824	2.43 *
7317	7	8	15	0.467	-0.52
7535	8	11	19	0.421	-0.92
7644	23	11	34	0.676	1.89 *
7769	15	6	21	0.714	1.75 *
7876	7	18	25	0.280	-2.40
8046	15	17	32	0.469	-0.53
8740	30	3	33	0.909	4.53 *
8992	20	2	22	0.909	3.62 *
9254	6	3	9	0.667	0.67
9549	11	10	21	0.524	-0.00
9883	7	5	12	0.583	0.29

Appendix H



the subject is free to withdraw consent and participation at any time.

On what page(s) of the proposal are the informed consent procedure and/or forms described? Page 8. (If not a part of the proposal, the procedures and informed consent document must accompany this application.)

6. EMERGENCIES

A. Are there risks to the human subjects? No.

If yes, describe briefly or give the page of the proposal where these are described.

B. Describe procedures for dealing with emergencies, or give the page of the proposal on which these descriptions may be found.

Since risks are no greater than normally encountered in everyday activities, no special emergency provisions have been taken.

7. PRIVACY: On what page of the proposal do you discuss procedures for keeping research data private? this page. This should include procedures for maintaining anonymity of subjects. Supplemental information concerning privacy of data may be discussed below.

During the course of the experiment, the experimental co-worker (the student) will keep the data unavailable to others. At the conclusion, the data will be given to the principle investigator who will file it in his locked office or destroy it. No results on individual subjects will be released -- formally or informally -- unless disguised or with the individual's permission.

8. STATEMENT OF AGREEMENT. The below named individual certifies that he has read and is willing to conduct these activities in accordance with the Handbook for Research, Development, Demonstration, or Other Activities Involving Human Subjects. Further, the below named individual certifies that any changes in procedures from those outlined above or in the attached proposal will be cleared through Committee 8290, The Committee on Research Involving Human Subjects.

Signed \_\_\_\_\_  
(Applicant)

Date \_\_\_\_\_

Appendix I

page 176

## ATTACHMENT

Task. Determine the attitudes towards computers of a group of students using a specially designed software package, versus the attitudes of a control group.

Experimental design. An attitude questionnaire will be given to the two groups of students and the students will rate their responses on a scale of 1 to 5.

Subjects and recruitment procedures. The students will be selected (from all the CS-200 classes) by the experimenter. All of the students in one class will (or will not, for the control group) use the software package.

Data collection time. The estimated time [from a pilot study] is one hour per student, per learning level. Presently there are five learning levels.

Statistical analysis. A T-test comparing the means of the two groups will be used to determine whether the results are statistically significant.

Application for approval to use human subjects.

Appendix I

page 177

## QUESTION CLASSIFICATION

In the following table, the 47 questions have been classified as to as to the type of question asked of the student. A 'cs' under the column marked 'cs' indicates that the question was designed to reflect the student's attitude toward computer science. The column marked 'cai' refer to questions concerning Computer Aided Instruction. The items marked 'lf' under the column marked 'lf' refer to questions that indicate that the student believes he/she learns faster under the conditions described in the respective question. The column marked 'inst' refer to the question that determines the students attitude toward the instructor. The items marked 'cc' refer to cross-check questions. (Question 2 cross-checks with Question 4). These pairs of items were inserted to allow the experimenter the option of checking to see if the students were just checking random answers when filling out the questionnaire.

Appendix J

page 178

	cs	cai	lf	cc	inst
1			lf		
2	cs			cc4	
3	cs				
4	cs			cc2	
5	cs				
6	cs				
7					inst
8		cai			
9		cai			
10		cai		cc33	
11			lf		
12			lf		
13		cai	lf		
14	cs				
15		cai			
16	cs				
17		cai			
18	cs			cc30	
19	cs			cc27	
20					
21	cs				
22		cai			

Appendix J

23	cai		
24	cai		
25	cai	lf	
26	cai		
27	cs		cc19
28	cai		
29	cai		
30	cs		cc18
31	cai		
32	cs		
33	cai		cc10
34	cai		
35	cs		
36	cai		
37	cs		
38	cs		
39	cai		
40	cai		
41	cai		
42	cs		cc44
43	cai		
44	cs		cc42
45	cai	lf	cc46
46	cai		cc45
47	cai		

Appendix J



## Bibliography

- Abel 82 Abelson, Harold. Logo for the Apple II. Byte/McGraw Hill,
- Abel 82b Abelson, Harold. "A Beginners Guide to Logo", Byte, 7,8 (August 1982), 88-112. The editors of Byte magazine, as is their custom, devote the major portion of any issue to a single topic. The August, 1982, issue of Byte centered on Logo, its implementation and applications. Much of the material in this paper was drawn from the Byte articles.
- Alde 78 Alderman, Donald L., Appel, Lola Rhea, and Murphy, Richard T. "PLATO and TICCAT: An Evaluation of CAI in the Community College", Educational Technology. (April 1978), p.40.
- Alle 69 Allen, Dwight W. "Team Research in Education", Educational Technology. (April 1969), 19.
- Ause 63 Ausebel. D.P. The Psychology of Meaningful Verbal Learning. Grune & Stratten, New York, N.Y. 1963. Peterborough, N.H. 1982.
- Balk 69 Blake, Howard E, & McPherson, Ann W. "Individualized Instruction -- Where Are We? A Guide for Teachers", Educational Technology. (December 1969), 63.

- Baym 83 Bayman, Piraye, and Mayer, Richard E. "A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements", Commun ACM. 25,9 (September 1983), 677-679.
- Benn 84 Bennet, Corwin E. In conversation with the author during the Spring and Summer terms of 1984 at Kansas State University, Manhattan, Kansas.
- Belt 76 Belt, Signey, and West, Sara F. "Implementing Individually Guided Education (IGE) with Computer Assistance", Educational Technology. 16,9(September 1976), 40.
- Bick 76 Bickerstaff, Douglas D. The Effect of Computer Assisted Instruction Drill and Practice Used to Obtain Homework Credit on Achievement and Attitudes of College Level Intermediate Algebra Students. PhD Dissertation. Kansas State University, Manhattan, Ks. 1976. This publication includes a list of questions that was used to determine students attitudes toward the use of CAI while learning (in this case) mathematics. The questions used by the author of this dissertation were drawn from Bickerstaff's list and many were reworded and/or modified so that the direction of questioning was more toward Computer Science.
- Cham 80 Chambers, J.A., Sprecher, J.W. "Computer Assisted Instruction: Current Trends and Critical Issues", Commun ACM. 23,6(June 80), 332-342.

- Crai 80 Crain, William C. Theories of Development : Concepts and Applications. Prentice-Hall, Englewood Cliffs, N.J., 1980.
- Davi 81 Davidson, Melvin. "ACM Forum", Commun ACM, 24,2 (February 81), 96-97.
- Denn 81a Denning, Peter J. "Eating Our Seed Corn", Commun ACM, 24,6 (June 81), 341-343.
- Denn 81b Denning, Peter J., Editor. "A Discipline in Crisis", Commun ACM. 24,6(June 81), 370-374.
- Devo 82 Devore, Jay L. Probability and Statistics for Engineering and the Sciences. Brooks/Cole Publishing Co. Monterey, Ca. p. 192.
- Dono 76 Donovan, John J. "Tools and Philosophy for Software Education", Commun ACM. 19,8(August 1976), 430.
- Dunc 81 Duncan, Karen A. "On Current Trends and Critical Issues in CAI", Commun ACM. 24,2 (February 1981), 95-96.
- Educ 69 ----- Educational Technology. (May 1969), 66.
- Educ 78 ----- Educational Technology. (April 1978), 7.
- Feur 79 Feurzeig, W., Papert, S., Bloom, M., Grant, R., and Solomon, C. "Programming Languages as a Conceptual Framework for Teaching Mathematics", Report 1889. Bolt, Beranek and Newman, Inc., November 1969. Initial development of LOGO was funded in part by an NSF research project conducted at Bolt, Beranek and Newman in Cam-

bridge, Mass. in 1968. This paper reports on that work.

- Feur 80 Feurzeig, W., Goldenberg, E.P., Lukas, G., Manis, V., Rubenstein, R., Stachel, R. The Logo-S Language and the Portable Logo System. Bolt, Beranek and Newman, Inc. 1980.
- Fish 82 Fisher, Donn. "Micro-Professor". Computist News (April 1982). Micro-Professor is sold by Multitech Electronics, Inc, 195 W. El Camino Real, Sunnyvale, Ca, 94086.
- Flak 75 Flake, Janice L. 'Interactive Computer Simulations for Teacher Education,' Educational Technology. (March 1975), 54.
- Fox 75 Fox, G. Thomas, Jr., De Vault, M. Vere, Golladay, Mary A. "A Descriptor for Individualized Instruction: Two Case Studies", Educational Technology. (May 1975), 25.
- Gagn 65 Gagne, Robert M. The Conditions of Learning. Hold, Rinehart and Winston. New York, 1965.
- Gall 77 Gallup, David Al. "Determining the Cost-Effectiveness of Instructional Technology", Educational Technology. (February 1977), 34.
- Gard 83 Gardner, David P., (Chair of The Presidents Commission of Education in Education). A Nation at Risk. Republished in Commun. ACM, 26,7 (July 1983), 467-478.
- Gibb 67 Gibbs, William E. 'The Teacher and Programmed Instruction,' Educational Technology. (June 15, 1967), 9.

Glin 84 Flinert, Ephraim P., Tanimoto, Steven L. "PICT: An Interactive Graphical Programming Environment", Computer, 17,11 (Nov 1984), 7-28.

Harr 81 Harris, Diana, and Nelson-Heern, Laurie, Eds. Proceedings of NECC81. The University of Iowa, Iowa City, Iowa, 1981. This Education/Computer conference was held on the campus of North Texas State University in Denton, Texas, June 17-19, 1981. A variety of papers covering a wide spectrum of computer assisted activities was given whose content includes the following topics:

- Computer simulation
- Interactive programs
- Transportability
- Micro Synergistic Computer's
- Video disc
- CAI
- Computer Aided Scheduling
- Computer Literacy
- Data Bases
- Management training
- Graphics
- Assembly Language Programming
- Computer Aids in the Humanities
- Social Science
- Math, & Science & Engineering

The papers covered projects that were implemented for use by students of various ages from pre-school to college level.

Hart 71 Hartman, Edward. "The Cost of Computer Aided Instruction", Educational Technology. (December 1971), 6.

Holt 77 Holt, R.C., Wortman, D.B., Barnard, D.T., Cordy, J.R. "SP/K: A

- System for Teaching Computer Programming", Commun ACM. 20,5 (May 1977), 301-309.
- Hoot 76 Hooten, David E., Zakia, Richard D. "Educational Technology and the Adult Learner", Educational Technology. (October 1976), 20.
- Kay 77 Kay, A. "Microelectronics and the Personal Computer". Scientific American (September 1977).
- Kear 82 Kearsley, Gred. "Authoring Systems in Computer Based Education", Commun ACM. 25,7 (July 1982), 429-437.
- Kell 68 Keller, Fred S. "Good-Bye Teacher...", Journal of Applied Behavioral Analysis, 1,1(1968),83.
- Krau 82 Krause, Kenneth L., Sampsell, Robert E., Grier, Samuel L. "Core Computer Science at the Air Force Academy", SIGCSE Bulletin. (February 1982), 144-146.
- Ledg 80 Ledgard, Henry, Whiteside, John A., Singer, A., Seymour W. "The Natural Language of Interactive Systems", Commun ACM. 23,10 (October 1980), 556-563.
- Lyon 78 Lyons, Norman R. "Systems Design Education: A Gaming Approach", Commun ACM. 21,11 (November 1978), 889-895.
- Mage 67 Mager, Robert F. "The Instructional Technologist", Educational Technology. (April 15, 1967), 15.

- Mage 81 Magel, Kenneth, et al. "Recommendations for Master's Level Programs in Computer Science - A Report of the ACM Curriculum Committee on Computer Science", *Commun ACM*. 24,3 (March 1981), 115-123.
- Maye 79 Mayer, Richard E. "The Psychology of Learning BASIC", *Commun ACM*. 22,11(November 1979), 589-593.
- Maye 81b Mayer, Richard E. "The Psychology of How Novices Learn Computer Programming", *ACM Computing Surveys*. 13,1 (March 1981), 121-141.
- Maye 81b Mayer, Richard E., Bayman, Pirayl. "Psychology of Calculator Languages: A Framework for Describing Differences in Users' Knowledge", *Commun ACM*. 24,8(August 81), 511-520.
- McLu 67 McLuhan, Marshall. *The Medium is the Message*. Bantam Books, New York. 1967.
- Morr 83 Morrison, Perry R. "A Survey of Attitudes Towards Computers", *Commun ACM*. 26,12 (December 1983), 1051-1057.
- Mosh 82 Moshell, Michael. *Computer Power : A First Course in Using the Computer*. McGraw-Hill, New York, 1982.
- Moze 82 Mozeico, Howard. "A Human/Computer Interface to Accommodate User Learning Stages", *Commun ACM*. 25,2 (February 1982), 100-104.
- Nels 81 Nelson, Harold. Ed. "Logo for Personal Computers", *Byte*, (June 81), 36-44. A bibliography of papers on Logo can be obtained by writing to Apple Logo, The Logo Project, 545 Technology Square,

Cambridge MA, 02139.

Ofie 68 Ofiesh, Gabriel D. 'Tomorrows Education Engineers,' Educational Technology. (July 15,1968), 5.

Patt 81 Pattis, Richard E. Karel the Robot : A Gentle Introduction to the Art of Programming. John Wiley & Sons, New York, 1981. Pattis describes the mechanics of Karel the Robot and outlines how the primitive commands of Karel can be combined to accomplish sophisticated programming tasks. The purpose is to teach programming concepts, and a large number of concepts are covered within the two and one-half weeks allocated for student exposure to Karel and his world.

Pape 80 Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York, 1980.

Rath 68 Rath, Gustave J. "Non-CAI Instruction Using Computers and Non-Instructional Uses of CAI Computers", Educational Technology. (Feb 15, 1968), 11-13

Ried 67 Riedesel, C Alan. 'An Appraisal of Computer Assisted Instruction,' Educational Technology. (May 30, 1967), 19.

The author lists eight advantages of CAI, particularly in comparison with other programmed devices.

Rosc 69 Roscoe, John T. Fundamental Research Statistics for the



Behavioral Sciences. Holt, Rinehart and Winston, Inc. New York.  
1969, p157.

- Seke 69 Sekerak, Robert, & McDonald, Bryon A. "Two Views of Educational Technology", Educational Technology. (August 1969), 47.
- Shei 81 Shiel, B.A. "The Psychological Study of Programming", ACM Computing Surveys. 13,1 (March 1981), 101-120.
- Shne 93 Shneiderman, Ben. "Direct Manipulation: A Step Beyond Programming Languages", Computer. 16,8 (August 1983), 57.
- Solo 74 Soloman, Lanny. "CAI: A Study of Efficiency and Effectiveness", Educational Technology, 14,10 (October 1974), 39-40.
- Spli 79 Splittgerber, Fred L. "Computer Based Instruction: A Revolution in the Making", Educational Technology, 19,1(January 1979), 20.
- Stol 68 Stolurrow, L.M. "What is Computer Assisted Instruction?", Educational Technology. (August 15, 1968), 10.
- Teit 81 Teitelbaum, Tim and Reys, Thomas. "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment", Commun ACM. 24,9 (September 1981), 563-573.
- Tuts 81 Tutsch, Jerry. "On the Crisis in Computer Science Manpower", Commun ACM. 24,9 (September 1981), 619.
- Uken 78 Ukena, Ann Seymour. Statistics Today. Harper & Row, Publishers,

New York, NY. 1978.

- Ullm 69 Ullmer, Eldon J. "Instructional Development in Higher Education: Basic Premises of a Learner Centered Approach", *Educational Technology* (April 69), 11
- vanH 76 van Hees, E J W M. 'Computer Managed Learning at the University Level in the Netherlands'. *Educational Technology*. (April 76), 28.
- Walk 76 Walker, Noojin. "What a Great Idea! Too Bad It Didn't Work", *Educational Technology*. (February 1976), 46.
- Wats 68 Watson, James D. *The Double Helix*. Atheneum Publishers, New York, NY. 1978.
- Watt 82 Watt, Dan. "Teaching Turtle: Logo as an Environment for Learning". *Popular Computing*. Vol 1, Number 9 (July 1982).
- Weiz 76 Weizenbaum, J. *Computer Power and Human Reason*. W.A. Freeman and Co., San Francisco, 1976.
- Wexe 81 Wexelblat, Richard L. "The Consequences of One's First Programming Language", *Software-Practice and Experience*. 11 (1981), 733-740.
- Wins 80 Winston, Patrick H. "Learning and Reasoning by Analogy", *Commun ACM*. 23,12 (December 1980), 689-703.

**A Model for Instructional Software Programming Concepts**

by

**JOHN WILLIAM MCDANIEL**

**B.S., East Central State University, 1972  
M.S., Oklahoma State University, 1975**

---

**AN ABSTRACT OF A DOCTOR'S DISSERTATION**

submitted in partial fulfillment of the

requirements for the degree

**DOCTOR OF PHILOSOPHY**

**College of Arts and Sciences**

**KANSAS STATE UNIVERSITY Manhattan, Kansas**

1985

## Abstract

A methodology for conveying Computer Science concepts via game-playing to non-computer science majors using reductionistic teaching techniques was developed. Three different games were developed following the guidelines of the model.

The model is divided into several phases. In the early phases, the user is given a problem to be solved using only a "mouse" (or, alternately, a selection of Function Keys) and successful completion of the problem results in one advancing to the next phase. The advanced phases generate code in a "Pascal-like" language to be replicated by the student.

Two of the games were implemented (one in Basic, one in Pascal) on micro-computers and the graphic capabilities of the hardware were utilized whenever possible. One of the games was used as the first homework assignment in two CS-200 (Introduction to Computer Science) classes the Summer of 1984. The students in the two classes (one experimental group, one control group) were given questionnaires that were used to determine their attitude towards (and, hence, effectiveness of) the software package. The results from a statistical analysis of the responses were positive.