DOCUMENT RESUME

ED 259 192 CE 041 920

AUTHOR Sayers, Dennis
TITLE Computer-Assisted Language Learning in Bilingual Vocational Education.
INSTITUTION Connecticut State Dept. of Education, Hartford.
PUB DATE 84
NOTE 64p.; For related documents, see CE 041 919-922.
PUB TYPE Guides - Non-Classroom Use (055) -- Computer Programs (101)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS Adult Basic Education; *Bilingual Education; Comparative Analysis; *Computer Assisted Instruction; Computer Software; *Courseware; Educational Benefits; Glossaries; Job Training; Language Skills; Limited English Speaking; *Microcomputers; Program Descriptions; Programing; Secondary Education; Second Language Instruction; Spanish Speaking; Team Teaching; Teamwork; Vocabulary Development; *Vocational Education; Word Processing
IDENTIFIERS *Computer Assisted Language Learning; Connecticut; Vocational English as a Second Language

ABSTRACT

This manual is designed to provide bilingual educators with information concerning the realistic potential of microcomputers in vocational education programs. Discussed first are the benefits, limitations, and hardware configurations of computer-assisted language learning (CALL). The next chapter deals with courseware and instructional management software for use in CALL and includes a sample rhyming dictionary and reviews of four commercially available language arts-related computer programs. The relative advantages of CALL programming in BASIC and LOGO are compared. Concluding the manual is a chapter on using a teamwork approach in developing programs. Appendixes to the handbook include a glossary and computer programs for a vocational math guessing game, one-key English and Spanish word processors, and a guessing game based on shop vocabulary. (MN)

COMPUTER-ASSISTED LANGUAGE LEARNING IN BILINGUAL VOCATIONAL EDUCATION

This manual on computer-assisted language learning in bilingual vocational education programs was prepared by Dennis Sayers and published by the Connecticut State Department of Education.

State of Connecticut

William A. O'Neill, Governor

Board of Education

June K. Goodman, Chairwoman

James J. Szerejko, Vice-Chairman

A. Walter Esdaile

Warren J. Foley

Roberto Fuentes

Abraham Glassman

Rita L. Hendel

John F. Mannix

Julia S. Rankin

Norma Foreman Glasgow (ex officio)
Commissioner of Higher Education

Gerald N. Tirozzi
Commissioner of Education

Frank A. Altieri
Deputy Commissioner
Finance and Operations

Lorraine M. Aronson
Deputy Commissioner
Program and Support Services

Contents

Preface

The Division of Vocational-Technical Schools (DVTS) of the Connecticut State Department of Education (CSDE) through its Regional Vocational-Technical Schools provides bilingual educational services for Limited English Proficient (LEP) students in grades 9 through 12 and bilingual training programs for LEP adults.

In order to furnish technical assistance to staff and to take advantage of the educational opportunities afforded by computers, the DVTS obtained financial support from the Bilingual Education Service Center of the University of Hartford to develop this manual. Because of his expertise in this area, Dennis Sayers was commissioned to produce it.

Dr. Saul Sibirsky, the DVTS's Bilingual Vocational Education Consultant, provided technical assistance to the project.

Appreciation is extended to Dr. Sarah Melendez, former Coordinator of the Bilingual Education Service Center, for her important collaboration. A debt of thanks is owed to David Wyatt of Specialized Curriculum Design, a pioneer in computer-assisted language learning, who offered invaluable orientation at the outset of the project.

Angelo J. Tedesco
Associate Commissioner,
Director
Division of Vocational-Technical Schools

v

CHAPTER ONE -- Introductory Concerns

This manual has two essential objectives. The first is to offer accurate information to bilingual educators concerning the realistic potential of microcomputers in vocational education programs. A practical overview of computer-assisted language learning is provided, and the two major avenues to implementing this instructional strategy -- purchasing available software and developing original programs -- are thoroughly considered.

The second is to demystify a new educational technology which typically engenders either hostility or fear -- and often both -- in the average teacher.

The information presented here should interest

> bilingual teachers at every level,

> language educators, and

> administrators, vocational instructors, job-specific English as a second language teachers, and counselors working in bilingual vocational education or training programs.

The first chapter begins with definitions of key terms, and continues with a discussion of microcomputer "hardware" or machinery. Chapter Two considers types of available programs or "software", and concludes with a review of a sample program for each major type. The third chapter focuses on the skill of programming original material for bilingual vocational education, and the last chapter outlines a model computer-based lesson or "courseware" development process, concluding with several useful programs. A glossary of terms is appended.

Bilingual vocational education implies learning a new language -- that of a particular occupation -- while learning skills for employment. In many ways, the bilingual educator faces a similar learning task when she or he decides to confront the new language and skills of educational computing. Microcomputers will continue to assume an increasing role in every learning context. As we learn how to use computers in bilingual vocational education, we too are learning employable skills for our own jobs.

In any case, we as bilingual teachers are faced with a clear choice: either come to terms with the new technology and learn to adapt it to our own needs, or accept commercially developed materials when -- or if -- they appear.

The manual itself was written with the Bank Street Writer, a simple word-processing program designed for students in elementary grades and beyond. A diskette with the programs presented in this manual is available on a loan basis from the Connecticut State Department of Education, and the writer would be most pleased to respond to any queries.

BASIC TERMS

Certain basic definitions are provided at the outset which will clarify key concepts used throughout the manual.

LEP (Limited English Proficient) - referring to individuals who have not attained sufficient English skills and are therefore denied the opportunity for meaningful learning experiences where English mediates instruction, unless their first language is also utilized for instructional delivery. The term LEP should not be construed to imply or deny competency in another language other than English, and can be distinguished from non-English proficiency (NEP).

Job-Specific ESL (English as a Second Language) or Job-Specific ESOL (English for Speakers of Other Languages) -- instruction for LEP persons in the English vocabularies and syntax specific to successful performance in a particular occupation or job.

BVE (Bilingual Vocational Education) or BVT (Bilingual Vocational Training) -- programmatic skills development for the LEP individual which seeks to provide education and training in an employable skill, while developing vocation-specific English language skills, in a manner which respects and incorporates into instructional delivery the students' native language and cultural identity, so that successful participants may find, get, and keep a job in an English-dominant workplace.

CALL in Bilingual Vocational Education

Bilingual vocational students are LEP individuals from a particular language and cultural background whose limited English ability FOR A SPECIFIC VOCATION has been identified, this determination being influenced by two factors: 1) the length of the vocational skills program, and 2) the language demands of a particular occupation.

CAI (Computer-assisted Instruction) -- interactive instruction in which the computer is used as an aid to the presentation, reinforcement and assessment of material to be learned. Sometimes termed CBI (Computer-based Instruction) or, in Great Britain, CAL (Computer-assisted Learning), CAI is typically associated with microcomputers.

CALL (Computer-assisted Language Learning) -- The interactive use of computers to encourage the acquisition of language concepts and production skills.

Microcomputers -- Sometimes called a desk-top or home computer, the "micro" is a commercially available computer with a memory capability that is larger than the original UNIVAC computers of the 50's, and which is appearing in more and more school settings. A micro will have anywhere from 8K (8,000 pieces of information) of memory to around 64K at the time this manual was printed, while a main-frame is a full-scale computer typically found in large business applications.

A S S E T S
of CALL in Bilingual Vocational Education

1) Interactive with immediate feedback

2) Individualized and self-pacing

3) Relatively easy to update, if not "locked up" by the publisher of commercial materials

4) Potential for open-ended programming (games, simulations, problem-solving)

5) Special needs (such as those of the limited English proficient and all categories of exceptional

learners including the handicapped, the talented and the learning disabled) can be addressed through innovative, "customized" programming.)

Perhaps the most important feature of CALL programs is their interactive nature. "Interactive" refers to the capability of the computer to tailor its functioning according to the responses of the learner, skipping easy material or reinforcing more difficult concepts with extra work. Indeed, it is precisely this feature that some teachers "fear" the most, since the teacher provides the only other major form of instruction that is interactive, and some educators feel their job may be on the line.

Of course, the circumscribed nature of the immediate feedback provided by a computer is, even in the best designed programs, a poor substitute for human intervention in the learning process. However, it is far superior to most other forms of non-human instruction in its ability to combine presentation, assessment, and positive and negative reinforcement of a dynamic learning process.

Because CALL as a rule involves a single student at a computer station, there exists a potential for the computer to provide instruction keyed to the individual's learning style and learning rate. Alternately branching instructional sequences may be programmed which permit more rapid instruction or more intensive review of the material being presented. Moreover, the ease with which the individual learner's progress may be monitored encourages timely intervention by an instructor in the CALL process.

Since a CALL program ultimately consists of a sequence of electronically stored information, it is relatively easy for a trained programmer to modify, customize or update existing programs. In BVE this is a critical attribute, since instructional materials usually undergo extensive and on-going revision. A well-designed CALL program can be further modified to address special learning needs, which is rarely possible in print or other A-V media, which must aim at the most typical reader or audience.

Finally, the interactive and real-time processing capabilities of the computer permit the design of open-ended learning experiences in which instruction is a "by-product" of an otherwise motivating involvement with the computer: a game, simulation or problem-solving program designed to encourage language learning.

LIMITATIONS
of CALL for Bilingual Vocational Education

1) Not for large groups

2) Susceptible to abuse typical of new learning technologies

3) Constrains students' responses

4) Focus on reading and writing skills and a single learning mode

5) Expense of hardware, software and maintenance

6) Expensive planning and slow production of original materials

7) Unavailability of commercial materials with direct relevance to ESL, BVE or CALL

As a rule, CALL is an alienating experience if an instructor does not develop strategies to offset the atomization of learning promoted when individuals are paired with computers. The advantages of individualization and tracking of student progress are lost if the instructor cannot modify group instruction in response to whatever may be taking place in the "computer room."

CALL is particularly susceptible to abuses associated with new learning technologies. Perhaps the most relevant example is provided by the checkered history of the language laboratory. Early -- and misguided -- enthusiasts made spectacular claims which language teachers often accepted without question.

The advent -- one might say, the explosion -- of microcomputers into the business, the consumer and finally the educational markets has resulted in a similar situation, counterbalanced by a very real fear, and consequent rejection, on the part of many educators of all forms of Computer-assisted

Instruction. Perhaps more than any educational technology in recent history, the use of micros in learning is such a rapidly changing field that even the "experts" -- never mind the lay person -- have trouble staying on top of it all.

Language learning on microcomputers is severely restrained both by the complexity of natural language and the limitation of the computer's "languages." The majority of computer languages, such as BASIC or PASCAL, which are forced to treat human languages as a sequence of numerical codes, simply cannot cope with the complexities of spelling, syntax, semantics and the extensive lexicon of a human communication system, not to mention the special language of an occupation, or bilingual learning needs across two human languages.

Generally speaking, the computer requests a single-key response (T or F; A,B,C or D) to a highly-constrained but more easily "processed" student response. Most educators would doubt the effectiveness of any language teaching methodology which depended almost exclusively on multiple-choice question formats.

Moreover, the present state-of-the-art in CALL relies very heavily on reading and writing skills, with unacceptable quality in speech-generation or spoken language recognition devices for the computer.

Finally, there are remarkably few existing materials available for BVE or ESL or CALL. The educational publishing establishment has shown little willingness to provide appropriate materials of marginal profitability. The technology will probably change before materials are available for the special concerns of language learning in a bilingual vocational setting.

Therefore, a serious limiting factor must be viewed as essentially financial. Coupled with the expense of securing and maintaining CALL hardware and courseware is the even greater cost of developing courseware appropriate to BVE. This cost is very difficult to estimate. BVE programs involve an on-going and productive collaboration between vocational and ESL instructors. Very rarely does either have programming expertise. Equally unusual is the programmer who is skilled in translating the desires for useful instructional materials of both the job-specific ESL and the vocational instructors.

Ultimately, the lack of instructional material resources must be viewed as a human resource development challenge.

CALL Hardware Configurations

Hardware is the physical machinery which makes up a computer system. It is distinguished from software, which are computer programs that make the hardware do something, for example, balance your checkbook, update your inventory, or do the payroll. A subclass of software is courseware, which may consist of a program that presents a CALL lesson, or one which keeps track of students' scores. CALL software and courseware will be discussed in the next chapter.

Any computer hardware setup consists of input devices (the keyboard, the disk drive, the cassette tape recorder), the computer itself, and output devices (the video monitor, the printer, and again, the disk drive and cassette recorder). A typical "run" of a computer system usually involves inputting some information (data) and a program (a list of computer language commands) into the computer; execution or "running" of the program and manipulation of the information by the computer; and outputting the results of the program's execution to the user.

Administrators and instructors working with scarce financial resources are justifiably concerned with the costs of computer hardware. We will attempt to outline a minimal CALL hardware set-up, keeping in mind that a cheaper system that will not do the job is always more expensive than any set-up that gets the job done. C.A.L.L. imposes special hardware requirements which are outlined below.

INPUT DEVICES - There are two major input devices which are relevant to CALL: the keyboard and the disk drive.

The KEYBOARD is typically attached to the computer itself, and is the primary means by which the learner communicates with the system. At this point in time, voice-discrimination devices are expensive and limited in their capabilities; therefore, typing at the keyboard becomes an important skill (and a significant barrier) for "talking" to the computer in CALL. The keyboard should be full-sized, with movable keys like an electric typewriter; touch pads are much slower and lead to many more errors in typing.

The DISK DRIVE accepts a square plastic envelope with a round plastic disk inside, and then spins the disk looking for programs and other information stored on it. When the disk drive "finds" what it is looking for, it "reads" this information into the computer's memory. This is a critical job in any computer run, since it would take forever to type in programs every time you needed to run one. Therefore, the primary purpose of the disk drive is to quickly give the computer a lot of information (like a program). The typical purpose of the keyboard is to interact with the computer as it runs the program it "got" off the disk from the disk drive.

While there are other ways to store programs, such as cassette tapes, the disk drive outperforms all the competition. The cassette recorder is very slow and requires considerable expertise on the part of the learner. The tape must be loaded and rewound and the counter set to zero; the location of the program must be looked up and found by pushing "fast forward" and "rewind" buttons; only then can the computer be given the program.

The complexities of good CALL requires the rapid and repeated access to programs and other information that only a disk drive can provide. A program dealing with human language will often be very long, and other "support" materials, like bilingual dictionaries, can be stored on a disk and accessed in the middle of a program; with a cassette recorder, none of these possibilities are feasible.

THE COMPUTER -- There are hundreds of books available on just what a computer is, but for our purposes, it is perhaps more helpful to understand what a computer must do in CALL in order to describe its minimal characteristics.

It will help to conceive of the computer as an empty electronic box. To get it to do anything, you have to 1) plug it in, 2) turn it on, and 3) fill it up. If you ever turn it off or unplug it or the lights go out due to an electrical storm, you are definitely out of luck, since you have lost everything once the electricity stops flowing through the circuitry.

The important concept here is that of "live memory"; a computer is "good" if it has a big enough memory to hold the information it needs to do the job you need it to do, and then only so long as it is turned on. This leads to two major considerations: how big a memory is big enough (memory capacity)? and how can I save my work before I turn off the computer (mass storage)?

Computer memory capacity is measured in a unit called "Ks". A minimal CALL memory capacity is 48K. Most computer memory capacities can be upgraded by purchasing green circuit-board cards, but virtually any significant CALL program needs at least 48K of memory to run; 64K is ideal for most applications. The "cost" of memory for computers is coming way down, so that 128K and 256K are sure to be available by 1986 on microcomputers; this will indeed be a boon for CALL programming.

The second consideration -- what to do before you turn the computer off -- underscores the critical importance of the disk drive, once again, in any C.A.L.L. set-up, and brings us to the final hardware devices, the machines that handle output.

OUTPUT DEVICES -- The major and essential output devices are the video monitor, the printer and the disk drive.

The VIDEO MONITOR constitutes the principal means by which the computer "interacts" with the learner in C.A.L.L. Just as the voice-recognition input devices are very expensive and none too effective, voice-synthesis output devices to produce speech-like sounds are costly and less than satisfactory in quality.

While a tape recorder can be hooked up to a computer, this is not a feasible alternative for "talking" to the user. Because of its uni-directional nature and the slowness with which recorded speech can be accessed, the use of a tape recorder eliminates one of the computer's principal assets: the ability to branch to alternate learning paths according to the student's learning needs. Therefore, most CALL programs will communicate with the learner on a TV-like screen, and must assume some requisite reading skills in order to be even minimally effective. A monochrome monitor (not a TV set) is preferred for clarity and to avoid learner fatigue.

A PRINTER is also a critical component in a CALL setup. Producing "hardcopy" or a printed version of a student's work that she or he can carry away greatly increases the effectiveness of the CALL session. Furthermore, if teachers are to be involved in the development of CALL courseware, the printer is doubly essential. Paper copy is a must for the tedious process of working out the "bugs" or errors in teacher-developed courseware.

Every learner needn't be hooked up to a printer; several computers can be tied into a single printer and access to the printer can be regulated with other green circuit-board cards called "firmware". But no serious teacher-based materials development can take place without at least one printer. The printer should produce plain paper (not thermal paper) copy, should have a graphics printing capability, and should have local service support available. The printer will break down before anything else.

Just as the DISK DRIVE is the primary input device for "mass" or large amounts of information such as programs, it is also the number one machine to handle large amounts of output from the computer. It is therefore sometimes referred to as an "input/output" device. Thus, the disk drive greatly expands the computer's capabilities.

Even though 64K of memory, meaning about 64,000 pieces of information, might seem like more than enough for any job, the fact is that a really complete program dealing with language learning takes up a lot of memory. The disk drive -- as an input and output machine -- can act as a "traffic cop" to manage the flow of information into and out of the computer's memory space.

Let's consider an extended example. If you write a CALL unit comprised of several lessons that's bigger than the machine's memory capacity, you can output the program one lesson at a time to the disk drive, which can "save" the longer program as a series of smaller programs linked together.

When you want to run the CALL unit, you can use the disk drive as an input device to the computer; the disk drive can "read" the smaller lessons one at a time as needed. Without the disk drive, the computer, as powerful a tool as it is, is incapable of "remembering" all that it needs to know to deal effectively with something as complex as a human language, let alone language learning, not to mention bilingual language learning with a vocational focus.

One acceptable CALL computer hardware configuration, that used by the author, is offered below.

INPUT
-----
1) Apple Disk][disk drive (with card)
2) attached keyboard for Apple][e

COMPUTER
----
1) Apple][e, 64K, with 80-column language card

OUTPUT
------
1) 9" monitor (black-and-amber)
2) Pro-writer printer, with Tackler interface card which permits graphics printing
3) Apple Disk][disk drive (see INPUT above)

CHAPTER TWO -- What about Software?

Computer - assisted Language Learning, like all forms of CAI, takes certain basic forms, although, as will be seen, there are major exceptions. CALL implies two major categories: non-instructional and instructional CALL.

Non-Instructional CALL

At first, the term "non-instructional CALL" may appear to be a contradiction in terms. How can there be computer-assisted language learning that is not instructional by definition? The fact is that the computer is a remarkably powerful tool capable of manipulating information tirelessly, and as any English teacher will tell you, preparation of language learning material frequently involves the most monotonous drudgery on the teacher's part. Non-instructional CALL helps the teacher accomplish this drudgery.

A very important example of the use of this power of the computer is found in Improving Techniques in Teaching English for the Job [sic -- the Bank Street Writer can't underline] (Interamerica Research Associates, 1982). This document details the results of a pioneering effort to utilize large, mainframe computers in the analysis of vocational textbook materials.

Using an advanced text-processing package written in PL/1 called TXTPRO, instructional materials were analyzed and all kinds of useful reports were generated: alphabetized and reverse alphabetized lists of vocational words, frequency lists (alphabetized both backwards -- showing key suffixes and inflections -- and the more traditional forwards), 2- and 3-word combinations, and even concordances (where a word is shown in its context) !

Any language teacher in a bilingual vocational program would recognize these listings as a goldmine, both for mediating dialogue with the vocational instructor about the specialized vocabulary and structures of the trade, as well as more directly as a primary source for lesson planning.

Even though these listings were generated on a mainframe computer, educators with micros should not despair. While the field of computerized textual analysis is a specialty area in computer science, there almost certainly will appear packages that can generate similar information on micros, if not for whole texts, then for chapters entered one at a time. Meanwhile, there are many word processing software packages which can perform some of these rudimentary listing chores, such as alphabetization and frequency studies.

Another use of non-instructional software for bilingual educational ends is the use of what are termed database management programs, which is computer jargon for computerized filing and fetching systems. In a database system, the teacher can create a file of tools used in a particular trade, for example, machining. The file may stipulate the tool's primary and secondary functions, any identifying or distinguishing characteristics, what machines it is typically used with, how it is calibrated, related tools, etc. Once the file is created, different tools may be entered into the system. Of course, the file may be updated or changed at any time.

The teacher may use the database system to conduct searches of the tool database for specific characteristics and "sorts" of cross-referenced attributes. This information can be used in a number of ways in a bilingual vocational program. For one, the process of developing the file and getting the Vocational and Job-specific ESL teachers to complete the database can serve as a focus for productive discussion of the trade and clarity of its linguistic "structure". Secondly, the ESL or Vocational Instructor can utilize the database to isolate vocational language "points" for study in the classroom or the shop. Finally, learners may access this information and multiply their basic knowledge by generating reports of vocational language.

This author is implementing a database system for spelling and pronunciation, which is intended for general as well as content-specific and vocational learning. The most frequent words in general language use have been filed both by their natural language spelling, by whether or not they are single- or multi-syllabled, by the letter-sound correspondences of their initial and final consonant clusters, etc. Of course, any vocational word may be entered into this general file as well. The resulting "sorts" of cross-indexed information may yield rich insights into spelling and pronunciation for both teacher and learner. Two pages from this computer-generated bilingual rhyming dictionary are included at the end of this chapter.

A final "non-instructional" example of CALL is the use of word processing software in bilingual vocational education. While the usefulness of these packages has been hinted at in previous paragraphs on rudimentary vocational text analysis conducted on micros, the major use of word processors in education is so significant that it will be treated in considerable detail in a review of the Bank Street® Writer later in this chapter. At this point, it is sufficient to indicate that in this writer's estimation word processing ranks with the LOGO programming language as the most important development to date in CALL for bilingual vocational education.

Instructional CALL

As later reviews of software will demonstrate, the distinction between instructional and non-instructional software can get blurry at times. Nevertheless, most of the literature on CAI speaks of three categories of courseware with such frequency that educators should be familiar with the terminology, even while we recognize the need to keep an open mind and be alert to courseware that transcends these categories.

A first category of courseware is referred to typically as DRILL & PRACTICE. It is relatively easy to produce, and is often similar to materials available in other (less expensive) media, such as print. Drill & Practice courseware concerns itself with the repetitive types of practice that are critical to language learning: vocabulary development (flashcard programs) or verb conjugations are likely candidates for this type of instruction on a micro. Often these programs are sequenced by order of imagined difficulty, but very rarely is a Drill & Practice program expected to TEACH anything. It is a reinforcement activity.

A second category is the TUTORIAL, which is further sub-divided into linear and branching. A tutorial attempts to teach something, and often includes some sort of pre-test, an introduction to motivate interest, a presentation and finally an evaluation. In short, it is a lesson, and more specifically, it is a form of programmed learning.

A branching tutorial permits a number of alternate learning paths, either based on an assessment of the student's learning style, or owing to his or her difficulty in comprehending the material "taught" by the computer. While linear tutorials can be easy to produce and are often implemented as readily through other less expensive media, branching tutorials are much more difficult and expensive to create, yet are far superior to anything available in other media (certainly, those "go to page 105" programmed learning books which many of us have used are a good example of a "bad fit" of branching tutorials in the print media).

A third category is often called SIMULATIONS, but I prefer the term PROBLEM-SOLVING courseware, with simulations as a sub-set of this type of instructional software. Problem-solving courseware are designed to encourage creative thinking with a "by-product" of content-specific learning. A simulation, for example, might be used in a bilingual program to teach how the circulatory system functions in the human body, a perfect application since no experiments can be allowed! If the learner is allowed to manipulate certain variables, such as food or liquid intake, the computer can keep track of the effects these decisions WOULD HAVE on a human body, and report these to the student, who then can draw conclusions and refine learning by manipulating variables more effectively.

Similarly, games can be designed which have as an additional outcome the learning of certain information. In a later chapter, a guessing game is designed which the learner must develop a strategy to solve, namely, halving; yet the game is set up in such a way that the resulting halved numbers provide practice with those numbers used in decimal and fractional equivalencies, a crucial competency in many occupations.

Indeed, in the case of courseware written in LOGO, it is possible to use certain principles of

Artificial Intelligence research to simulate human-like language behavior on the computer's part. Before we take exception to the term Artificial Intelligence, imagine how ludicrous it would be to maintain that no one should fly since airplanes engage in Artificial Flight! In fact, many important lessons are being learned in the field of linguistics from what computers can -- and can't -- be programmed to do with language. A later chapter includes a program which appears to "learn" from its mistakes, and is very effective in a bilingual vocational setting.

The following reviews sample available software (both non-instructional and instructional) in terms of their relevance for bilingual vocational learning:

Review: The Bank Street Writer

A word processor is a powerful tool being used more and more by professionals of all specializations. It would be difficult to imagine anyone who works with words not having been exposed to word processing.

Basically, a word processor is a machine that composes text without the use of paper. All the inevitable errors of typing and the unavoidable agonies of revision are more easily corrected on the word processor. Rather than retyping a page with a single typo, the error can be removed before printing; text can be moved about and insertions made electronically, rather than with scissors and rubber cement.

Therefore, one major educational application of microcomputers is the use of word processing programs by teachers, curriculum designers and administrators. In this sense, we are no different than any other profession in accepting the efficiencies of this remarkable tool: so much of our work is based on effective and timely use of the written -- and the re-written -- word.

As bilingual educators and vocational educators, however, word processing takes on a further significance. More and more occupations, even at the entry level, require what were previously considered sophisticated skills with the written word. In many instances, word processing skills themselves are considered elementary requisites in a growing number of jobs. Familiarity with a "simple" yet powerful word processor like the Bank Street Writer can facilitate the development of writing skills and a working knowledge of word processing that can be transferred to the text editing machines found on many jobs today.

Effective writing is a skill -- or more accurately, an integrated array of related skills -- which is learned only with much practice and considerable difficulty. Most of us can recall how we learned to write and to compose written material. Intense effort went into producing the only copy the teacher would accept, a "perfect" copy.

And how would we get back the results of our labor? Very rarely without being "damaged" by the teacher, with instructions to re-write another perfect copy, which in turn would be subjected to a similar fate, and so on. It would be hard to imagine a more de-motivating experience than learning to compose the way most of us have experienced.

The Bank Street Writer promises to change all that. A learner may write an assignment, correcting and revising as she or he finds necessary. A "draft" -- which any of us who are writers accept as a necessary intermediate stage of anything we write -- is printed for the teacher, who then provides suggestions for revision.

The Bank Street Writer can then be utilized to IMPROVE the original work. When all revisions and corrections have been made, a "perfect" final copy -- or a hundred -- can be made on the printer. It would be hard to imagine a more motivating way to focus the practice and incorporate the criticism that is so necessary to learning to write.

Bilingual educators, as well, would do well to look closely at the educational benefits of word processing.

Language learning involves the four skills of listening, speaking, reading and writing. In order to master these highly complex linguistic competencies, our LEP students will have to make thousands and thousands of errors, and will need to learn from their "mistakes" as much as possible. In the related areas of spelling, reading and writing, the word processor offers one productive channel through which the process of learning from linguistic errors can be harnessed for language acquisition purposes.

The Bank Street Writer is the word processor that was used to prepare this manual.

It comes with three modes: WRITE, EDIT and TRANSFER. WRITE is the mode you use when you enter text from the keyboard. EDIT is used to make all kinds of changes (like ERASE/UNERASE to remove and put back text, MOVE/MOVEBACK to change the position of written material, and FIND/REPLACE which looks for errors and substitutes corrections). Finally, the TRANSFER mode takes care of your "files", by getting them, saving them, renaming them or printing them in draft or final form.

Because all the instructions to run the program are on the screen at any time, the Bank Street Writer is said to be "menu-driven". There are many other error-handling features, with beeps and warning messages which occur whenever any irreversible steps are about to be taken, allowing you "one last chance" to change your mind.

Thus, the Bank Street Writer is extremely easy-to-use without being a toy word processor, and can be an effective catalyst to writing skills development as well as an introduction to word processing in general. These attributes only generally suggest the many creative applications which can be made with the Bank Street Writer in bilingual vocational education. Specific suggestions of uses of this simple word processor follow.

Of course, the development and production of all manuals, texts, worksheets and other written support materials can be greatly facilitated with any word processor. This is a particularly relevant trait in bilingual vocational work since materials necessarily undergo extensive continuing revisions.

Yet the ease with which a learner may use the Bank Street Writer presents other potential applications. For example, one important part of any vocational skill is the mastery of sequenced steps to accomplish a particular occupational task. To produce a cylindrical piece in a metal-cutting lathe, for example, the following sequence must be followed:

1. Cut off the stock, allowing for waste.
2. Insert the workpiece into the chuck.
3. Face the end of the workpiece.
4. Center drill the end of the workpiece.
5. Reverse the workpiece.
6. Face the other end to the final length.
7. Center drill the other end.
8. Place the workpiece between centers.
9. Turn the diameter of the workpiece to the figure on the blueprint.
10. Give the workpiece to the instructor for inspection.

These steps comprise an elementary "task analysis" of a basic occupational skill. Either the Job-specific ESL or the Vocational Instructor could easily create a valuable learning experience by removing the numbers and mixing up the steps, asking the learner to use the Bank Street Writer to re-order the steps and printout the correct sequence, which then could become part of the student's notes.

This example requires no sophisticated programming skills, yet achieves similar goals as more expensive programmed courseware, with the additional advantage of being more easily integrated into the existing curriculum, since it is teacher-made.

The Bank Street Writer opens up a new avenue of microcomputer applications for bilingual vocational education, one that is particularly relevant since the critical element of CONTROL rests with the instructors and the LEP learners.

Review: Language in Contrast

This CALL package consists of 8 diskettes containing a total of 12 lessons and a Teacher's

Manual. The materials are available from The Center for Educational Experimentation, Development and Evaluation of the University of Iowa.

Language in Contrast is a typical example of what is available for CALL with a bilingual focus: grammar lessons which present "grammar points" illustrated in the learner's first language, and which limit the student's "interaction" to hitting a number or a letter key of a series of possible multiple choice responses.

The "Teacher's Manual" is a slim document indeed. The main body of the text is a page and a half long, followed by a list of never more than 5 objectives for each of the 12 lessons. According to the Manual, the lessons "were developed for LEP students. The purpose of the lessons is threefold:

> 1. to increase the students' awareness of the linguistic similarities and differences between the home language (Spanish) and the target language (English) in order to facilitate transition;
>
> 2. to reinforce the students' understanding of word meanings, usage, and grammar of the home language (Spanish) in order to facilitate vocabulary acquisition and language usage in the target language (English);
>
> 3. to provide students with the analytical skills to comprehend similarities and differences between English and Spanish grammar."

While these are the stated goals of the programs, the structure of every lesson in fact serves to reinforce very different skills, and more importantly, attitudes about the "home" and "target" languages.

Each lesson gives grammar information in English for at least 2 or 3 full screen "pages," then asking if the material was understood. If the answer to the question is "no" (posed in English!), then the SAME material is presented in Spanish. While the intention may be to "force" learners to try to comprehend the lesson in English, the hidden message is to look at the home language as a to-be-discarded crutch.

The "language learning lesson" therefore quickly becomes an exercise in how well one can memorize the Spanish content and respond to the English questions, and get on with it.

And while each lesson begins with a very flashy set of the Center's initials swirling at you from the distance, the fact is that the process of going through the material is BORING. It was all this writer could do to work his way through it.

Yet Language in Contrast is practically unique among tutorial materials which attempt to teach English utilizing bilingual approaches! Of course, all the examples are "neutered," seemingly without any contextual content, as many language learning materials unfortunately are; therefore, there is absolutely NO vocational focus to these materials or any others this writer could discover.

The reality of the paucity and superficiality of the available materials for bilingual and/or vocational language learning leads us quite properly to the next review, which treats software that permits the instructor to create courseware.

Review: STORYBOARD and CLOZEMASTER

STORYBOARD and CLOZEMASTER are offered as examples of "generative" CALL software, "templates" which an instructor may use to produce software that is directly keyed into the curriculum. Like other programs available from Wida Software, CLOZEMASTER and STORYBOARD are based on an elegantly simple concept that harnesses the computer's "workhorse" capacity to manipulate and keep track of sequenced data, in this case, language.

The STORYBOARD package consists of a simple manual and a single diskette containing a menu-driven program to write "lessons", create and correct student diskettes, and perform other utility functions.

An effective vocational lesson can be quickly generated by entering a series of steps to perform a basic occupational task, much like the example offered in the review of the Bank Street Writer above. When the student inserts a diskette, an opportunity is provided to see instructions, and the option is offered to read the text for 30 seconds, 5 seconds, or not to see the text at all. Then the text is presented, with every character represented by an asterisk. Thus,

3. Face the end of the workpiece.

becomes

*. **** *** *** ** *** *********.

All punctuation is maintained, thus giving some minimal clues to meaning. The learner is given a chance to guess a word. Many students immediately develop an effective strategy of entering common structure words such as "the", "a", "of", etc. If a word is correctly guessed, it is then inserted at every point it occurs in a passage.

Thus, structure words alone generate many contextual clues as more content-oriented words are attempted. If the student gets stuck, there are a number of options -- "HHH" gives help in the form of the first letter of the first unguessed word, "?" shows the whole word, "SEE" shows the whole text till RETURN is pressed, and "EEE" exits the program.

CLOZEMASTER is a similar template program which allows the instructor to present text in a Cloze test format, with every n-th word deleted, which the learner then fills in and has judged by the computer.

Once again, this is a job for which a computer is eminently suited, since a Cloze exercise is very mechanical and arithmetical in nature. Yet a great deal of linguistic skills development may occur in this type of exercise, and the CLOZEMASTER permits instructors to determine the vocational content and accuracy of the material to be practiced.

Additionally, students can determine which n-th word they want deleted, thus permitting learners to decide the level of complexity of the material, an important element indeed, and one that is missing in many more expensive courseware.

Similar help messages are provided as in the STORYBOARD programs, but unlike STORYBOARD, CLOZEMASTER will offer a score, each time offering the message that on a Cloze exercise, 70% is considered excellent and 100% is rarely attained.

Indeed, the unabashedly non-competitive focus of the Wida materials is refreshing, and stresses the learning - aid intent of their template courseware development software, a welcome relief from many a noisy arcade look-alike CAI package mascarading as educational courseware.

Both STORYBOARD and CLOZEMASTER can be used in tandem to practice similar skills in different ways, and the complementary nature of the resulting courseware can lend as much consistency as variety to teacher - generated materials.

As simple as these programs may appear, they can be enormously effective in reinforcing both linguistic skills and content-area knowledge, and STORYBOARD and CLOZEMASTER are therefore highly recommended for easily - generated bilingual vocational materials.

Following are two pages from a bilingual rhyming dictionary created with a database management program (described on pages 13 and 14 of this manual). The words presented are from the first thousand most frequent words in written English.

THE n WORD FAMILY

ă	ĕ	ĭ	ŏ	ŭ	ŏŏ
an	men	been	gone	done	
can	ten	in	on	fun	
man	then	thin	``upon	one	
ran	when	[skin		run	
than	`any	`finished		son	
[plan	`anyone	`interest		sun	
`animal	`anything	`interested		won	
`animals	`center	`interesting		`countries	
`answer	`century	`into		`country	
`answered	`energy	`minutes		`hundred	
`answers	`general	`window		`money	
``began	`many	`winter		`running	
	`sentence	``begin		`under	
	`sentences	``beginning		`wonder	
	``again	``within			
	``attention				

an	un/una [antes de una vocal]
been	estado/sido
can	poder [verbo]/lata
done	hecho
fun	divertido
gone	ido
in	en/dentro de
man	hombre
men	hombres
on	sobre/en/en uso u operación
one	uno
ran	corrió
run	correr
son	hijo
sun	sol
ten	diez
than	que [de comparacion]
then	entonces/luego
thin	delgado
when	cuando
won	ganó/venció
[plan	plan/planear
[skin	piel
`animal	animal
`animals	animales
`answer	respuesta/contestación
`answered	contestó/contestado
`answers	respuestas
`any	cualquier
`anyone	cualquier persona
`anything	cualquier cosa
`center	centro
`century	siglo
`countries	países
`country	país
`energy	energía
`finished	terminado/terminó
`general	general
`hundred	cien
`interest	interés
`interested	interesado/interesó
`interesting	interesante
`into	adentro/en
`many	muchos
`minutes	minutos
`money	dinero
`running	corriendo
`sentence	oracion/sentencia
`sentences	oraciones
`under	debajo de
`window	ventana
`winter	invierno
`wonder	preguntarse/maravilla
``again	otra vez
``attention	atención
``began	empezó/comenzó
``begin	empezar/comenzar
``beginning	principio/empezando/comenzando
``upon	sobre
``within	dentro

Visual Spelling and Pronunciation through Rhyme, Bilingual Co-op Press, 1983.

$\bar{a}$	$\bar{e}$	$\bar{i}$	$\bar{o}$	$\bar{u}$ / $\overline{oo}$	$\hat{o}$	ou	oi
main	mean	fine	known	moon	gone	down	
rain	seen	line	own	soon	on	town	
[plane	[clean	sign	shown	```afternoon	``upon	[brown	
[train	[green		[grown			`mountains	
``contain	`meaning		[stone				
``explain	``between		`only				
	``machine		``alone				

down	abajo
fine	bien/fino
gone	ido
known	conocido
line	línea
main	principal
mean	significar/querer decir/cruel
moon	luna
on	sobre/en/en uso u operación
own	ser dueño de/poseer/propio
rain	lluvia
seen	visto
shown	mostrado
sign	cartel/indicador/señal
soon	pronto
town	pueblo
[brown	marrón/castaño
[clean	limpiar/limpio
[green	verde
[grown	crecido
[plane	avión/plano
[stone	piedra
[train	tren/adiestrar
`meaning	significado
`mountains	montañas
`only	solamente
``alone	solo
``between	entre
``contain	contener
``explain	explicar
``machine	máquina
``upon	sobre
``afternoon	tarde

26

CHAPTER THREE -- CALL Programming

A Vocational Instructor or a Job-specific ESL Instructor might very well despair at the lack of directly applicable "computer lessons", throwing up their hands altogether at the unlikely possibility of ever finding CALL courseware in their students' language of the home.

What is "out there" is clearly for the mainstream student; even these materials which are well-designed utilize a level of English well beyond the skills of most LEP students.

One obvious response is to arrange to program your own materials. On the one hand, this is probably the only realistic alternative in courseware development for CALL in Bilingual Vocational Education since it is very unlikely that a profit will ever be realized by commercial publishers, thus providing little incentive for the significant investment of time and expensive programming expertise.

Indeed, it is perhaps not even desirable to depend on publishing houses to supply the courseware needs of Bilingual Vocational Education, which relies so heavily on the productive collaboration of all members of the education or training program staff, particularly the Vocational and Job-specific ESL instructors. Any CALL materials should flow directly out of -- and be subject to constant revision based on -- the on-going dialogue between those who know the language of the occupation and those who facilitate learning it.

However, the obstacles which present themselves are overwhelming. In fact, it is not easy to program really significant courseware; it will be the rare Vocational Instructor or Job-specific ESL teacher who can also produce CALL lessons in the quantity and of the quality necessary to warrant the tremendous investment of time required. What will often happen is that a third person is introduced into an already complicated dialogue: the educational programmer must somehow mediate the often vaguely articulated needs of the occupational skills and vocational language instructors.

The courseware development process under these circumstances is remarkably long. The need for explicitly stated objectives in instructional materials is nowhere felt more strongly than in CALL courseware. Yet the Job-specific ESL Instructor's lack of familiarity with the vocabulary and structures of the occupation and the Vocational Instructor's need to develop new strategies to encourage occupational language acquisition combine to create an extremely "volatile" hard-to-pin-down situation.

Yet if there are ever to exist materials which are relevant to the needs of LEP learners in Bilingual Vocational programs, we as educators must begin to confront the challenges of CALL programming. In my estimation, there are three alternative courses of action which present themselves:

1. Learning a general purpose programming language such as BASIC

2. Learning a special purpose educational programming language such as EnBASIC or PILOT

3. Learning LOGO, a general purpose programming language tuned for educational application.

Programming in BASIC

BASIC was originally developed to teach elementary programming concepts to engineers at Dartmouth. It has often been described as easy to learn, with a limited number of commands, and very hard to do anything of significance with.

While it is rarely used in business, BASIC is definitely the "coin of the realm" in computer magazines, where many of the programs that are listed are offered in BASIC. It is the language which is offered as "built-in" on many computers, and quite a bit of educational courseware is written in BASIC since programs in this widely used language are more "portable," or useable on a variety of machines with minor adaptations.

BASIC is definitely a good language to know, but it is often erroneously regarded as easy-to-learn.

In fact, learning BASIC or any programming language is much like learning a spoken language as a second tongue: minimal fluency requires at least 100 hours of instruction and many more hours of guided practice. Even then, significant programming results require twice this commitment. It is important to beware of the tendency of those who act as though the acronym meant "Be A Self-appointed Instant Computer-expert." The name BASIC belies the complexity of the language.

The nature of this complexity lies in the very versatility of BASIC: it is designed to do anything, that is, it is an all-purpose programming language, equally suitable for balancing the company's books and writing a computerized lesson. Street corner philosophy will tell you that anyone or anything that gets spread so thin loses something in the process.

From the point of view of CAI, BASIC is clumsy; simple processes like storing questions and their answers with hints or scorekeeping require quite a bit of juggling in BASIC's arrays, and heaven help the one who mixes numbered with lettered variables or can't unlearn high school algebra so that BASIC's use of " A = A + 1 " makes sense.

These shortcomings are magnified in any application that has to do with language learning. Since BASIC views words as "strings" of coded characters, and cannot recognize even the most basic hierarchy of a space separating words into sentence parts, the processing of natural language brings immense complexities.

The problems are particularly evident in what is termed "answer-processing". Language is such a free-form medium that there is typically tremendous variation in correct student responses, and it is often undesirable to punish answers which are only slightly wrong, since much may be learned from the error. BASIC is merciless; if a single jot or tiddle is off; then the student is totally wrong.

All these factors combine into a typical BASIC language lesson: even though a language teacher would like full utterances, the shorter response, often the simple multiple-choice letter, is isolated, as is the highly focussed grammatical point, since BASIC requires very strict parameters to function properly.

Educational Programming Languages

Computer languages arise out of specific applications. COBOL responds to business needs, FORTRAN to scientific programming task. In fact, BASIC's original impetus was the need for a language to teach elementary programming concepts. Educational applications are a major computer market, and specialized programming languages have arisen which respond to educational needs. The major example of this type of computer language is PILOT, which is a totally distinct programming language that includes specialized commands, which facilitate the design of lessons by teachers.

While any teacher must be relieved to realize that whole computer languages have been designed with our needs in mind, educational programming languages have major drawbacks. These are similar to the disadvantages for computerized instructional systems; most educational languages for the computer produce information-based lessons described by most observers as "programmed instruction," or electronic page-turning.

The potential of the computer to produce more "generative," open-ended and user-defined learning experiences is rarely tapped.

Moreover, learning a computer language implies exactly that -- investing a significant amount of time, at least 100 hours, to achieve minimum competency, and educational programming languages are certainly no exception. Many educators, even the most devoted, hesitate to invest that amount of time in learning an entire language which can only be used to write lessons.

A very interesting response to these drawbacks is the EnBASIC Authoring System. This system is an add-on to BASIC, so that this most common of popular programming languages can be built on, rather than scrapped altogether when it's time to write lessons. EnBASIC is definitely designed with the teacher in mind, and the language teacher in particular.

EnBASIC has its own character set editor, which means you can make foreign language letter sets yourself. Other display features are double sized letters, super- and sub-scripts, and all kinds of rewrite, erase, inverse and exclusive-or modes, which in English means there's a lot of potential for animation. Graphics and text can be mixed together.

But the real promise of EnBASIC lies in its ability to respond to natural language from the student. As a rule, if a single character is "off" in BASIC, the answer is considered completely wrong. EnBASIC is much more flexible. The teacher can specify that a student's answer may contain synonyms, that capitalization is unimportant, that only certain words should be judged as answers and the rest ignored, even that students' spelling be "marked-up".

All these features add up to advanced "answer-processing" capabilities that increase the amount of feedback which the student may receive in a computerized language lesson. The advantages of this approach would be completely lost if the student had no way to correct errors; EnBASIC very nicely provides a miniature line-editor which permits simple corrections after feedback and hints have been offered, and the answer can then be rejudged.

Bilingual Language Learning in a LOGO-mediated Computer Environment

Most computer programming languages, while attempting to bridge the gap between the machine's code -- nothing but endless lines of zeros and ones -- and the natural language of humans, seem less human and more mechanical, as though they were designed to accomodate the machine and only incidentally its user. As concerns students, it seems the "language needs" of the computer have received much more attention than the language needs of learners.

By way of contrast, let's examine a key passage from Papert's landmark book MINDSTORMS, which details the evolution since the late '60s of the LOGO programming language. During a pilot implementation of LOGO in the Brookline Public School System, a group of seventh graders developed programs to write what they called "computer poetry." They "taught" the computer to generate sentences by randomly selecting from lists of words grouped according to syntactic function, i.e. nouns, verbs, adverbs, adjectives, etc.

> One of the students, a 13-year-old named Jenny, had deeply touched the program staff by asking on the first day of her computer work, 'Why were we chosen for this? We're not the brains.'...One day Jenny came in very excited. She had made a discovery. 'Now I understand why we have nouns and verbs,' she said...[As] she tried to get the computer to generate poetry, something remarkable [had] happened. She found herself classifying words into categories, not because she had been told she had to but because she needed to. In order to 'teach' her computer to make strings of words that would look like English, she had to teach it to choose words of an appropriate class. What she learned about grammar from this experience with a machine was anything but mechanical or routine. She understood the general idea that words (like things) can be placed in different groups or sets, and that doing so could work for her. She not only 'understood' grammar, she changed her relationship to it. It was 'hers,' and during her year with the computer, incidents like this helped Jenny change her image of herself. Her performance changed too; her previously low to average grades became 'straight A's' for her remaining years in school. She learned she could be a 'brain' after all.

It's important to understand the learning dynamic of this particular project. A group of students wanted to get the computer to write poetry using their own words. They found that the only way to accomplish this task was to categorize words by their function in sentences, and looked to their teachers as a resource for this activity.

The teachers and students ALL knew LOGO, and mediated their communication about human language through the LOGO language. The result was a much more active and profound learning experience than turning the page of an electronic workbook.

A few words are in order concerning the origins and characteristics of LOGO. Its theoretical foundation is in the work of Piaget, whose view of the learner as an active agent in a dynamic process of assimilation and accomodation between what is already known and what is being learned is reflected in the key LOGO concept of de-bugging. For Piaget, and in LOGO, a "mistake" leads to the question "Why that particular wrong answer?" or "What is the logic behind the thinking that led to this error?"

As a language for programming, LOGO has virtually no threshold and certainly no ceiling -- it has been used to produce non-trivial programs by 4-year-olds through graduate level physics students, by "average" students like Jenny, by learners from such disparate groupings as the autistic, the gifted, the learning disabled, the handicapped, the developmentally disabled, indeed every class of special needs student and exceptional learner.

The typical path into the LOGO programming environment is through turtle graphics, by which sophisticated programming techniques are introduced through a computer "sketchpad" microworld. For primary school children, there is the additional aid of the turtle robot, which anthropomorphizes programming by running around the floor drawing pictures like the ones the students have programmed on the monitor.

LOGO and the turtle have found their way into thousands of schools since 1969, and this fact itself has created both enormous potential and significant obstacles for the successful use of LOGO in bilingual vocational education and language learning.

The real advantage of LOGO use is that it is widely available on a number of computers and a lot of teachers and kids have used it. Many are familiar with LOGO and are enthusiastic about its potential. A major disadvantage is that few are aware of its capabilities beyond working with very young children or teaching elementary programming -- sometimes called computer literacy -- through turtle programming. LOGO thus has assumed an aura of "childishness," and is frequently pooh-poohed when talk of serious educational programming begins.

However, the real potential of LOGO for language acquisition lies in certain unique properties of its design. In computer jargon, LOGO is procedural and extensible (meaning that it uses certain primitive commands which you can then combine into new procedures which become an integral part of the LOGO language, and can be as English-like or Haitian-Creole like as you wish); LOGO utilizes list processing (read, that words and sentences can be treated as whole entities rather than as chains of numerical characters); and LOGO relies on recursion as its principal programming strategy (which means that procedures can actually contain themselves, as though the computer gave you 3 wishes and you used the third to get 3 more, and so on, an extremely powerful property for manipulating language structures).

LOGO occupies a unique position among educational software. It is not a lesson, or an authoring system, or a word-processor, or a standard programming language, although it can be used to give lessons, author programs or process text, and is a powerful general-purpose programming language. It may best be described as a computer learning environment tuned for interesting educational applications.

LOGO's promise for bilingual education and vocational language learning lie in the level of integration it brings to the total learning context. With even rudimentary programming training in LOGO, teachers can translate their insights into vocational language to students. As Papert notes,

> In many schools today, the phrase 'computer-aided instruction' means making the computer teach the child. One might say the computer is being used to program the child. In my vision, the child programs the computer.

What Papert says of children is also true of the bilingual vocational instructor. All too often, CAI means watching students go off to the "computer room," leaving the instructor to wrestle with the logistics of incorporating into a group-oriented curriculum sequence whatever learning may have taken place there.

An alternate scenario is to place your curriculum into the hands of the computer "expert". In both schemes, learners and instructors are passive observers, not active participants.

LOGO offers other, more interesting

possibilities and many more challenges. By combining graphics and programming and word-processing and list-processing into a single package, and even more importantly, by providing access to all this "computer power" for teacher, learner and educational programmer alike, LOGO permits a more balanced development of computer-based skills for all involved in the learning environment.

For bilingual education and vocational training, this is particularly critical, since so much of the responsibility for curriculum development, computerized or otherwise, must rest within a school's or program's control.

A Comparison of LOGO and BASIC

The two programs which follow are offered to allow the reader to compare similarities and differences of LOGO and BASIC. The LOGO program accomplishes the same task as the BASIC program, using slightly different strategies. However, both programs are "Flashcard" lessons and get the same job done: drilling for vocabulary development.

It must be stressed that these programs are extremely elementary, and are not presented as examples of excellent CALL programming. However, each includes key features of most CALL lessons in rudimentary form: instructions, presentation of a question with judging of the learner's response, positive and negative feedback, and scorekeeping. And both programs may be used to generate an unlimited number of new flashcard lessons by changing a couple of lines. Indeed, true/false or multiple choice formats can be easily created from these same programs.

The LOGO program is written in MIT LOGO, while the BASIC program is in Applesoft BASIC. Minor changes will be required in other dialects of LOGO or BASIC.

```
10   READ N
20   DIM A$(N,2)
30   FOR K = 1 TO N
40   READ A$(K,1),A$(K,2)
50   NEXT K
60   LET R = 0
70   LET T = 0
80   LET CR = 0
100   HOME
110   PRINT "LET'S PRACTICE SOME W
      ORDS."
120   PRINT
130   PRINT "Translate, then press
       [Return]."
140   PRINT
200   IF R = N THEN 500
210 R = R + 1
220   PRINT : PRINT "==============
      ================================": PRINT

230   PRINT A$(R,1)
240   PRINT
250   INPUT "->";SA$
260   IF SA$ = A$(R,2) THEN  GOTO
      300
270   GOTO 400
280   PRINT
290   GOTO 200
300   PRINT
310   PRINT "GOOD! KEEP IT UP!"
320   LET CR = CR + 1
330   LET T = T + 1
340   GOTO 200
400   PRINT
410   PRINT "BETTER STUDY THAT ONE
       SOME MORE!"
420   GOTO 200
500   PRINT
510   PRINT "=========================
      ======================"

520   PRINT
530   PRINT "YOU GOT ";CR;" RIGHT"

540   PRINT
550   PRINT "OUT OF ";N;"."
560   END
1000   DATA  4
1010   DATA  "NUNCA", "NEVER", "SI
      EMPRE", "ALWAYS", "DE VEZ EN
       CUANDO", "CNCE IN A WHILE",
       "ANOCHE", "LAST NIGHT"
```

```
TO FLASH
 CLEARTEXT
 MAKE "RIGHT 0
 MAKE "TOTAL 0
 PRINT [Translate, then press <Return>.]
 WAIT 3
 QA [NUNCA] [NEVER]
 QA [SIEMPRE] [ALWAYS]
 QA [DE VEZ EN CUANDO] [ONCE IN A WHILE]
 QA [ANOCHE] [LAST NIGHT]
 SCORE
 FLASH
END

TO QA :QUESTION :ANSWER
 PRETTYLINE
 \
 PRINT1 ( SENTENCE :QUESTION [->] )
 TEST REQUEST = :ANSWER
 IFTRUE POSITIVE STOP
 IFFALSE NEGATIVE STOP
END

TO POSITIVE
 \
 PRINT [GOOD! KEEP IT UP!]
 \
 MAKE "RIGHT :RIGHT + 1
 MAKE "TOTAL :TOTAL + 1
 WAIT 3
END

TO NEGATIVE
 \
 PRINT [BETTER STUDY THAT ONE SOME MORE!]
 \
 MAKE "TOTAL :TOTAL + 1
 WAIT 3
END

TO SCORE
 \
 PRINT ( SENTENCE [YOU GOT] :RIGHT )
 \
 PRINT ( SENTENCE [OUT OF] :TOTAL [.] )
 \
 PRETTYLINE
 WAIT 5
END

TO PRETTYLINE
 PRINT [=======================================]
END

TO WAIT :SECONDS
 REPEAT :SECONDS * 1000 []
END
```

CHAPTER FOUR -- A Teamwork Approach to Developing Programs

Let us imagine for a moment the process of developing CALL courseware for a bilingual vocational program.

There are three critical participants in this process: the Vocational Instructor, the Job-specific ESL Instructor, and the CALL Programmer.

The Vocational Instructor must exercise considerable leadership in any curriculum development effort in bilingual vocational education. The particular challenge posed by bilingual vocational training is the requirement to specify IN WORDS what has become second nature to the vocational practitioner. The Job-specific ESL Instructor needs to know exactly the safety requirements of an occupation; what steps are required to complete a vocational task; the typical occupational vocabulary and structures used to communicate on the job, and much more. And the source of this information will be the Vocational Instructor.

Similarly, the Job-specific ESL Instructor must not only determine what the vocational language is that will be taught and how to teach it, but often must aid the Vocational Instructor in techniques to reinforce language learning at the same time as job skills are acquired.

The CALL Programmer mediates the curriculum development process in any bilingual training program where micros are used to advance instruction. Often, this may result in a recommendation that computers are inappropriate to teach or practice a particular concept. However, if she or he determines that courseware development is realistic, then the C.A.L.L. Programmer will play the key role in translating the teaching objectives of the Vocational and Job-specific ESL Instructors into reality.

What follows is a simulation of the process of developing courseware for a Machinist Bilingual Training Program.

Meeting One

The Vocational Instructor speaks of the primary communication needs of a machinist: the ability to understand and communicate measurements taken with precision instruments, such as the micrometer and the scale. This skill involves a number of sub-skills: namely, a working knowledge of fractional and decimal measuring systems, the ability to comprehend and communicate these measurements, and the skill of measuring itself.

While the measuring skill must be taught in a hands-on approach, the Vocational Instructor points out, there is plenty of potential for language practice in learning the fractional and decimal systems, particularly understanding and speaking the numbers common to each system. Indeed, one of the major deficiencies of most novice machinists is their inability to convert between fractional and decimal systems. Any competent machinist will know that three-eighths is three hundred seventy five thousandths, as well as the decimal equivalents for a host of other fractions.

The Job-specific ESL Instructor takes the initiative by asking the Vocational Instructor exactly what the most common equivalencies are, and under what conditions they are used on the job. The reply is that a machinist uses a scale to measure halves, quarters, eighths and sixteenths, but has to convert these into thirty-seconds or sixty-fourths such as appear on the scale. However, the machine is calibrated in decimal dimensions, specifically, thousandths of an inch, so that every cut on the machine must be converted into thousandsths.

The CALL Programmer intervenes to ask what all these systems have in common, and is told that they all are given in terms of an inch. When pressed for more commonalities, the Vocational Instructor stresses that each system is based on halving the inch progressively. Halves break the inch into two parts, quarters into four parts, eighths into eight, etc., until thousandths break the inch into one thousand parts. Thus, 1/2 = 2/4 = 4/8 = 8/16 = 16/32 = 32/64 = .500 .

This is clearly a lot to digest, and the meeting moves on to other points with the agreement to pick this discussion up when all have had a chance to give it some thought.

Meeting Two

The discussion resumes with all participants assessing what each might best accomplish in the teaching of fractional and decimal equivalencies. The Vocational Instructor is sure that the hands-on practice of measuring and calling out dimensions can be best achieved in the shop. The Job-specific ESL Instructor feels that a good lesson on the grammar structure "There is / there are" can be built around the question "How many halves are there in one inch?", "How many quarters...?", "How many thousandths...?", etc.

The CALL Programmer feels that a program that would actually teach all that is involved in this complex set of skills and sub-skills would take very long to develop, even longer to refine, and would probably not compete with an instructor for effectiveness. A game or a problem-solving program that would address one key sub-skill that required a great deal of practice would be much better-suited to the teamwork approach of achieving this learning goal. Thus, the task of giving the learners practice with the concept of halving seems best left to the endless patience of the computer.

An easily developed problem-solving strategy game that would provide considerable practice in halving is a guessing game. The CALL Programmer proposes a guessing game which would 1) ask the student to select a number like 8, 16, 32, 64 or 1,000 which forms the basis for a fractional or decimal system; 2) choose a secret number between zero and that number; 3) tell the student that he or she had a limited number of tries to guess the secret number. For example, if the learner selected 1,000, the computer would randomly select a number, say, 327. This number can always be guessed in nine tries if the correct strategy is used, namely, halving the original range.

A guess of 500 would yield a response "Too high, try a smaller number." A second guess of 250 gets "Too low, try a bigger number." The third try of 375 results in "Too high, try a smaller number", etc. As the student guesses, he or she is also using the most common numbers associated with the decimal/fractional system, namely 62, 125, 187, 250, 312, 375, 437, 500, 562, 625, 687, 750, 812, 875 and 937.

Similarly, if 64 was chosen, trying to guess the number in less than six tries would encourage a strategy that is based on halving and would require the use of the numbers typically used in the 64ths scale.

Further meetings

This process continues of outlining learning goals and determining how each instructional element of the bilingual training program can best contribute to the overall objective of encouraging skills and job-specific language development, and may result in interesting educational spin-offs.

What follows are examples of programs which may serve as suggestive models for CALL programming in bilingual vocational programs. They are written in LOGO, which is easier to follow than most high-level languages, and are available in diskette format on a loan basis from the Connecticut State Department of Education.

1) A guessing game program such as described above.

2) A "word-processor" that uses one-key commands, in English and in Spanish.

3) A program modeled on the famous "Animal" program, which appears to be learning from its mistakes. This program is very interesting, in that it can be used to "extract" from the Vocational Instructor a small microcosm of information which can then be saved and used with learners. I call it "Shop" and it can be created by typing in

MAKE "KNOWLEDGE "MICROMETER

or whatever other tool or instrument is associated with the occupation. A vocational instructor playing this game for half an hour will create hours worth of learning material in the form of a guessing game that can be used by the Job-specific ESL instructor.

Program: Vocational Math Guessing Game

```
TO GUESS
 MAKE "ORDINALS [FIRST SECOND THIRD FOURTH FIFTH SIXTH SEVENTH EIGHT
H NINTH LAST]
 INTRO
 CHOOSENUMBER GETUP
 GETGUESS
 GUESS
END

TO INTRO
 CLEARTEXT
 PR [In every vocation, measurement is very] \
 PR [important. We measure distances in] \
 PR [the machine shop, and we use fractions] \
 PR [to talk about these distances.]
 HITRETURN
 CLEARTEXT
 PR [But the machine doesn't understand] \
 PR [fractions. It can only understand] \
 PR [decimals, and only in THOUSANDTHS!] \
 PR [So when we work with machines,] \
 PR [we constantly "translate" from] \
 PR [fractions to thousandths and back!] \
 HITRETURN
 CLEARTEXT
 PR [Here is a game to practice the most] \
 PR [common numbers used in the decimal] \
 PR [and fraction systems...]
 HITRETURN
END

TO GETUP
 CLEARTEXT
 PR [Type a letter to indicate which kind] \
 PR [of numbers you want to practice...] \ \
 PR [A > Decimals (thousandths)] \
 PR [B > Fractions (sixty-fourths)] \
 PR [C > Fractions (thirty-seconds)] \
 PR [D > Fractions (sixteenths)] \
 PR [E > Fractions (eighths)] \ \
 PRINT1 [Your choice? ->]
 MAKE "UP FIRST REQUEST
 TEST MEMBER? :UP [A B C D E]
 IFFALSE \ PRINT [Please type a letter!] WAIT 3 CHOOSENUMBER
 IFTRUE OUTPUT :UP
END
```

```
TO CHOOSENUMBER :UP
 IF :UP = "A MAKE "SEED 999 MAKE "SHOTS 10
 IF :UP = "B MAKE "SEED 63 MAKE "SHOTS 6
 IF :UP = "C MAKE "SEED 31 MAKE "SHOTS 5
 IF :UP = "D MAKE "SEED 15 MAKE "SHOTS 4
 IF :UP = "E MAKE "SEED 7 MAKE "SHOTS 3
 MAKE "NUMBER 1 + RANDOM :SEED
 CLEARTEXT
 PRINT [I'm thinking of a number between] \
 PRINT ( SENTENCE [1 and] :SEED + 1 [.] ) \
 PRINT ( SENTENCE [You have] :SHOTS [tries to guess it!] ) \
 PRINT [READY?] \
 WAIT 3
 PRINT [Here goes. . .]
 WAIT 3
END

TO GETGUESS
 \
 TEST :SHOTS = 1
 IFTRUE PRINT1 LAST :ORDINALS PRINT1 ">
 IFFALSE PRINT1 FIRST :ORDINALS PRINT1 ">
 MAKE "ORDINALS BUTFIRST :ORDINALS
 MAKE "GUESS READNUMBER
 CHECKGUESS
END

TO CHECKGUESS
 IF :GUESS = :NUMBER PRINT [GOT IT!] WAIT 3 GUESS STOP
 MAKE "SHOTS :SHOTS - 1
 TEST :SHOTS = 0
 IFTRUE PRINT ( SENTENCE [The answer was] :NUMBER [.] ) WAIT 3 GUESS
 STOP
 IF :GUESS > :NUMBER PRINT [Try a smaller number.] GETGUESS STOP
 IF :GUESS < :NUMBER PRINT [Try a larger number.] GETGUESS STOP
END

TO READNUMBER
 MAKE "NUM1 REQUEST
 TEST :NUM1 = []
 IFTRUE PR [Please type a number.] OUTPUT READNUMBER
 TEST NOT NUMBER? FIRST :NUM1
 IFTRUE PR [Please type a number.] OUTPUT READNUMBER
 IFFALSE OUTPUT FIRST :NUM1
END

TO WAIT :SECONDS
 REPEAT :SECONDS * 1000 []
END

TO \
 PR []
END

TO HITRETURN
 CURSOR 4 23
 PRINT1 [Press <Return> to continue.
 MAKE "WAITING REQUEST
```

Program: One-Key English Word Processor: WRITER

```
TO WRITER
 NODRAW
 PRINT [INSTRUCTIONS [for HELP just txpe "?"]]
 PR [=============================================] CR
 PR [> To WRITE, type W] CR
 PR [> To SAVE what you just wrote, type S] CR
 PR [> To PRINT, type P] CR
 PR [> To ADD to something you wrote, A] CR
 PR [> To READ your work, type R] CR
 PR [=============================================]
 COMMANDS
END

TO COMMANDS
 MAKE "COM READKEY
 IF :COM = "W WRITEIT STOP
 IF :COM = "S SAVEIT STOP
 IF :COM = "P PRINTIT STOP
 IF :COM = "R READIT STOP
 IF :COM = "A ADDIT STOP
 IF :COM = "? WRITER STOP
 COMMANDS
END

TO WRITEIT
 READTEXT "BLANK
 EDIT ; you must type WRITER now!------------------------------------------
---------
END

TO SAVEIT
 CLEARTEXT
 PRINT [GIVE YOUR WORK A NAME.]
 CR
 PRINT [TYPE ONE WORD ONLY, THEN [Return] .]
 CR
 PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE SAVEIT STOP
 MAKE "FILE < FIRST :FILE >
 SAVETEXT :FILE
 WRITER
END
```

```
TO PRINTIT
 CLEARTEXT
 PRINT [GIVE THE NAME OF YOUR WORK.]
 CR
 PRINT [TYPE ONE WORD, THEN [Return] .]
 CR PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE PRINTIT STOP
 MAKE "FILE ( FIRST :FILE )
 PRINTFILE :FILE
 WRITER
END

TO ADDIT
 CLEARTEXT
 PRINT [WHAT DO YOU WANT TO MAKE CHANGES ON ?]
 CR
 PRINT [TYPE ONE WORD ONLY, THEN [Return] .]
 CR
 PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE ADDIT STOP
 MAKE "FILE ( FIRST :FILE )
 CR
 READTEXT :FILE
 EDIT ; be sure to type WRITER now !-------------------------------------------
-----------
END

TO READIT
 CLEARTEXT
 PRINT [WHAT DO YOU WANT TO READ ?]
 CR
 PRINT [TYPE ONE WORD ONLY, THEN [Return] .]
 CR PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE READIT STOP
 MAKE "FILE ( FIRST :FILE )
 SHOWFILE :FILE
 HITRETURN
 WRITER
END
```

46

```
TO READTEXT :FILE
 .DEPOSIT :NOINTP 255 ; DON'T ALLOW CTRL-G
 .DEPOSIT :SAVMOD 1
 READ :FILE
 .DEPOSIT :SAVMOD 0
 .DEPOSIT :NOINTP 0
END

TO SHOWTEXT
 PRINT.MEM 8192 ( .EXAMINE :ENDBUF ) + 256 * ( .EXAMINE :ENDBUF + 1
)
END

TO SHOWFILE :FILE
 READTEXT :FILE
 SHOWTEXT
END

TO PRINTTEXT
 OUTDEV :PRINTER
 SHOWTEXT
 OUTDEV 0
END

TO PRINTFILE :FILE
 OUTDEV :PRINTER
 SHOWFILE :FILE
 OUTDEV 0
END

TO SAVETEXT :FILE
 .DEPOSIT :NOINTP 255
 .DEPOSIT :SAVMOD 1
 SAVE :FILE
 .DEPOSIT :SAVMOD 0
 .DEPOSIT :NOINTP 0
END

TO PRINT.MEM :FROM :END
 IF :FROM = :END STOP
 PRINT1 CHAR .EXAMINE :FROM
 PRINT.MEM :FROM + 1 :END
END

TO HITRETURN
 CURSOR 0 23
 PRINT1 [PRESS [RETURN] TO CONTINUE...]
 MAKE "WAITING REQUEST
END

TO CR
 PR []
END
```

Program: One-Key Spanish Word Processor: AUTOR

```
TO AUTOR
 NODRAW
 PRINT [INSTRUCCIONES]
 PR [==========================================]
 PR [Si quieres ...] CR
 PR [> ESCRIBIR, toca E] CR
 PR [> ARCHIVAR lo que has escrito, toca A] CR
 PR [> IMPRIMIR, toca I] CR
 PR [> CAMBIAR algo que has escrito toca C] CR
 PR [> LEER tu trabajo, toca L] CR
 PR [> ayuda, toca "?"]
 PR [==========================================]
 COMMANDS
END

TO COMMANDS
 MAKE "COM READKEY
 IF :COM = "E WRITEIT STOP
 IF :COM = "A SAVEIT STOP
 IF :COM = "I PRINTIT STOP
 IF :COM = "L READIT STOP
 IF :COM = "C ADDIT STOP
 IF :COM = "? AUTOR STOP
 COMMANDS
END

TO WRITEIT
 READTEXT "BLANCO
 EDIT ; tienes que escribir AUTOR ahora !------------------------------
------------------
END

TO SAVEIT
 CLEARTEXT
 PRINT [Que nombre quieres poner a tu trabajo?]
 PRINT []
 PRINT [Escribe una palabra, y luego [Return.]]
 PRINT []
 PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE SAVEIT STOP
 MAKE "FILE ( FIRST :FILE )
 SAVETEXT :FILE
 AUTOR
```

```
TO PRINTIT
 CLEARTEXT
 PRINT [Cual de tus trabajos quieres imprimir?]
 PRINT []
 PRINT [Escribe una palabra, luego [Return] >]
 PRINT [] PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE PRINTIT STOP
 MAKE "FILE ( FIRST :FILE )
 PRINTFILE :FILE
 AUTOR
END

TO ADDIT
 CLEARTEXT
 PRINT [Cual de tus trabajos quieres cambiar ?]
 CR
 PRINT [Escribe una palabra, luego [Return] .]
 CR
 PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE ADDIT STOP
 MAKE "FILE ( FIRST :FILE )
 READTEXT :FILE
 EDIT ; escribe AUTOR ahora !----------------------------------------
----
END

TO READIT
 CLEARTEXT
 PRINT [Que quieres leer ?]
 PRINT []
 PRINT [Escribe una palabra, y luego [Return.]]
 PRINT [] PRINT1 ">
 MAKE "FILE REQUEST
 TEST :FILE = []
 IFTRUE READIT STOP
 MAKE "FILE ( FIRST :FILE )
 SHOWFILE :FILE
 HITRETURN
 AUTOR
END
```

```
TO READTEXT :FILE
 .DEPOSIT :NOINTP 255 ; DON'T ALLOW CTRL-G
 .DEPOSIT :SAVMOD 1
 READ :FILE
 .DEPOSIT :SAVMOD 0
 .DEPOSIT :NOINTP 0
END

TO SHOWTEXT
 PRINT.MEM 8192 ( .EXAMINE :ENDBUF ) + 256 * ( .EXAMINE :ENDBUF + 1
)
END

TO SHOWFILE :FILE
 READTEXT :FILE
 SHOWTEXT
END

TO PRINTTEXT
 OUTDEV :PRINTER
 SHOWTEXT
 OUTDEV 0
END

TO PRINTFILE :FILE
 OUTDEV :PRINTER
 SHOWFILE :FILE
 OUTDEV 0
END

TO SAVETEXT :FILE
 .DEPOSIT :NOINTP 255
 .DEPOSIT :SAVMOD 1
 SAVE :FILE
 .DEPOSIT :SAVMOD 0
 .DEPOSIT :NOINTP 0
END

TO PRINT.MEM :FROM :END
 IF :FROM = :END STOP
 PRINT1 CHAR .EXAMINE :FROM
 PRINT.MEM :FROM + 1 :END
END

TO HITRETURN
 CURSOR 0 23
 PRINT1 [TOCA [Return] CUANDO TERMINES...]
 MAKE "WAITING REQUEST
END

TO CR
 PR []
END
```

```
TO SHOPTALK
 ND
 PRINT [THINK OF AN SOMETHING IN THE SHOP.]
 PRINT []
 PRINT [I WILL TRY TO GUESS IT]
 PRINT []
 PRINT [BY ASKING YOU SOME QUESTIONS...]
 PRINT []
 PRINT [==========================================]
 WAIT 2000
 GUESS :KNOWLEDGE
 PRINT [LET'S TRY AGAIN]
 WAIT 2000
 SHOPTALK
END

TO WAIT :N
 REPEAT :N []
END

TO GUESS :CHOICES
 IF ( WORD? :CHOICES ) FINAL.GUESS :CHOICES STOP
 MAKE "RESPONSE ASK.YES.OR.NO ( QUESTION :CHOICES )
 IF :RESPONSE = [YES] GUESS YES.BRANCH :CHOICES STOP
 GUESS NO.BRANCH :CHOICES
END

TO FINAL.GUESS :CHOICE
 MAKE "FINAL.QUESTION ( SE [IS IT] ADD.A.OR.AN :CHOICE [?] )
 MAKE "RESPONSE ASK.YES.OR.NO :FINAL.QUESTION
 IF :RESPONSE = [YES] PRINT [LOOK HOW SMART I AM!] STOP
 GET.SMARTER :CHOICE
END

TO ASK.YES.OR.NO :QUESTION
 PRINT1 :QUESTION
 IF "? = LAST LAST :QUESTION PRINT [] ELSE PRINT "?
 MAKE "INPUT REQUEST
 IF :INPUT = [YES] OUTPUT [YES]
 IF :INPUT = [NO] OUTPUT [NO]
 PRINT [PLEASE TYPE "YES" OR "NO"]
 OUTPUT ASK.YES.OR.NO :QUESTION
END

TO QUESTION :TREE
 OUTPUT FIRST :TREE
END
```

```
TO NO.BRANCH :TREE
 OUTPUT LAST :TREE
END

TO YES.BRANCH :TREE
 OUTPUT FIRST BUTFIRST :TREE
END

TO GET.SMARTER :WRONG.ANSWER
 PRINT [OH WELL, I WAS WRONG. WHAT WAS IT?]
 MAKE "RIGHT.ANSWER ( LAST REQUEST )
 PRINT [PLEASE TYPE IN A QUESTION WHOSE ANSWER]
 PRINT ( SE [IS "YES" FOR] ADD.A.OR.AN :RIGHT.ANSWER [AND] )
 PRINT ( SE ["NO" FOR] ADD.A.OR.AN :WRONG.ANSWER )
 MAKE "QUESTION REQUEST
 EXPAND.KNOWLEDGE :QUESTION :RIGHT.ANSWER :WRONG.ANSWER
END

TO ADD.A.OR.AN :WORD
 TEST MEMBER ( FIRST :WORD ) [A E I O U]
 IFTRUE OUTPUT SENTENCE "AN :WORD
 IFFALSE OUTPUT SENTENCE "A :WORD
END

TO MEMBER :ITEM :LIST
 IF :LIST = [] OP "FALSE
 IF :ITEM = FIRST :LIST OP "TRUE
 OP MEMBER :ITEM BF :LIST
END

TO EXPAND.KNOWLEDGE :NEW.QUESTION :YES.ANSWER :NO.ANSWER
 MAKE "KNOWLEDGE REPLACE :KNOWLEDGE :WRONG.ANSWER ( LIST :NEW.QUESTI
ON :RIGHT.ANSWER :WRONG.ANSWER )
END

TO REPLACE :DATA :WORD :NEW.BRANCH
 IF :DATA = :WORD OP :NEW.BRANCH
 IF WORD? :DATA OP :DATA
 OP ( LIST QUESTION :DATA REPLACE YES.BRANCH :DATA :WORD :NEW.BRANCH
 REPLACE NO.BRANCH :DATA :WORD :NEW.BRANCH )
END
```

52

G L O S S A R Y

The glossary which follows was created on the PFS:File software package, which is a database management system that is integrated with a word processor (PFS:Write), a report writing system (PFS:Report) and a graph making program (PFS:Graph). This family of software is particularly valuable in that they are designed to work together, are geared to the non-specialist, and have set the industry standard for good documentation.

ALGORITHM The solution to a particular computer problem. The part of the program which rounds off the percentage correct in a lesson is an algorithm.

ALPHANUMERIC Any character, a letter or a number, which a human can read and which can be represented to the computer.

ASCII AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE The code used to assign a number to each character of the keyboard, so that the computer can "crunch" letters like numbers. For example, A=65, B=66, etc.

APPLICATION PROGRAM A software package designed to do a certain job, balance the checkbook or teach a lesson.

A.I. ARTIFICIAL INTELLIGENCE Branch of cognitive science which studies systems that can perform "logical" analysis and "learning," improving every next performance based on experience; AI systems emulate human intellectual activity. LOGO is based on AI research.

AUTHORING PACKAGE Like an AUTHORING LANGUAGE, but usually MENU-DRIVEN, so that a special set of instructions don't have to be learned. Easier to learn to use, but somewhat monotonous in presentation.

AUTHORING LANGUAGE A programming language designed specifically for a task, for example, to make a medical information inquiry program, or to make a CALL lesson. Examples are PILOT, EnBASIC, EDUTEXT and TUTOR.

BOOT To start a computer up, often by turning it on and loading the Operating System.

BUG Something that keeps a program from working properly. A mistake in program design, rarely a malfunction in the computer. De-bugging is the process of correcting it, and learning to de-bug is the process of learning to program.

CHARACTER SET The set of characters & numbers which can be displayed on the monitor screen or printed on the printer. Foreign language character sets must be purchased as part of a printer package, or may be developed by the user with special software.

CODE What you type into a computer program; "30 PRINT "HI THERE!"; GOSUB 200" is a line of code in BASIC.

CALL COMPUTER ASSISTED LANGUAGE LEARNING Acronym gaining currency for foreign language and second language learning contexts; supposed to refer to a broad range of programming, but typically a grammar-based tutorial approach.

CAI COMPUTER ASSISTED INSTRUCTION Acronym common to USA.

CAL COMPUTER ASSISTED LEARNING Acronym common to Britain.

COMPUTERESE Slang for the jargon used by people who forget what they talked like before they used computers. Sounds much like a foreign language, the listener is required to stop the speaker and ask many questions.

CRASH When the program suddenly stops working and nothing you do to the keyboard results in any change, your program has crashed.

CURSOR The movable spot on the monitor which shows where the character that is about to be typed will appear.

DATA Information in a form which can be processed by a computer, and may be held within a written program, or input from the keyboard directly, or stored in a data file.

DATABASE MANAGEMENT SYSTEMS A fancy name for a computerized filing system; very helpful in storing and retrieving information. One application is storing student information, to be aggregated and manipulated for record keeping; another is the storing of key language and vocational information, to be analyzed for linguistic features, and taught in job-specific ESL lessons.

DISC or DISK or DISKETTE A mass storage device which allows you to store and get data rapidly. Can be floppy or hard, the floppy variety is used in educational applications and looks like a square envelope with a piece of brown plastic inside. Much preferred to other mass storage devices like cassette tapes.

DOCUMENTATION The manuals and other instructional materials which accompany a software package. Famous for their jargon and obtuse style. Good documentation generally reflects a clearly designed software package.

DRILL AND PRACTICE A repetitive process used to refine an emerging skill. Used in conjunction with another teaching technique.

EGRULE The technique of giving examples before the rule which explains them is provided. The opposite of RULEG.

FEEDBACK Information the learner receives from the program during execution regarding his/her performance. May be immediate or delayed, positive or negative, direct ("good!") or indirect (pleasant music).

FIRMWARE A little green circuit board which you can plug into your computer to allow it to perform new functions.

GIGO GARBAGE IN, GARBAGE OUT An adage which reflects the need to sit down and develop a program concept before switching on the computer.

GRAPHICS Any output on a monitor screen or printer which involves pictures, line drawings or graphs. Graphics may come in high resolution, low resolution or character graphics form, in descending order of quality.

HARDWARE The physical elements of a computer system, the machinery you can touch. Distinguised from FIRMWARE, and SOFTWARE.

HIGH LEVEL PROGRAMMING LANGUAGES Includes BASIC, COBOL, FORTRAN, PASCAL, LISP and LOGO; they are designed specifically for the convenience of human beings in communicating instructions to computers. These languages are converted by the computer into machine code, i.e. strings of numbers which are the only language the computer really understands.

INTERACTIVE Any form of instruction in which the response of the learner affects the response of the instructor, and vice versa.

LISTING The sequence of steps in a program, printed out in order so you can read them. Computer magazines are filled with listings, usually in BASIC, which you can type directly into your computer. Written programs.

MAINFRAME COMPUTER A very large computer which can be used by many people at remote terminals. Distinguished from Mini- and Micro-computers, medium-sized and desk-top sized computers, respectively.

MATCHING Very important in CAL and CALL, it is the process whereby the computer compares a learner's response with the "correct" expected response. Partial Matching is character by character comparison, sometimes with error signals offered, and is frequently called Answer Processing.

MEMORY Of two varieties, RAM and ROM. RAM is the working area where one writes and runs programs. ROM is an unchangeable area of memory which is usually supplied on chips. RAM is emptied every time the computer is turned off, or a fuse blows, or lightning strikes. ROM is there whenever you are using the computer.

MENU-DRIVEN When the user is presented a series of options, and can choose the next branch of a computer program, the program is menu-driven. These programs always return to a main menu, often leading to sub-menus.

MONITOR A video display that permits the computer to give information on a screen directly, as distinguished from a video receiver.

OPERATING SYSTEMS Software which supervises routine tasks in the computer, especially how the disk-drive reads information from the diskettes.

PARALLEL Allows computer information to be transmitted simultaneouly over one wire. Fastest form of transmission to a printer.

PRINTER Produces, or outputs, printed material or hardcopy. Dot-matrix printers are faster but lower in quality (this manual was produced on dot-matrix), while daisy wheel printers produce letter quality, carbon ribbon material, suitable for offset printing.

PROGRAM A sequence of instructions written in a programming language, and according to the requirements of a computer system, which the instructions will direct to solve a particular problem.

PROGRAMMING Process of writing a program in a programming language, reducing the solution of a problem to a program.

REAL TIME When a computer simulation takes place simultaneouly with the program execution, the result occurs in real time. This is not necessarily better; the simulation of geologic processes should be much faster than real time, for instance, but a chemical process should be slowed down considerably.

RULEG Giving the rule before the examples which illustrate it. Distinguished from EGRULE.

SERIAL Allows computer information to be sent only one piece at a time over a wire. Good for telephone transmission, but very slow.

SIMULATION Imitation of the functioning of a process by means of a program; the execution of the program is the simulation.

SOFTWARE Programs and their documentation which make the computer do its thing, like teach a lesson, once you turn it on.

STRING Denotes a cluster of concepts innportant for linguistic programming. A string is a sequence of characters or words (and numbers, possibly) which are handled differently than numeric values by the computer. They are manipulated by string functions, which look for the first letter, the last letter, the length of the total string, etc. String functions are to language what adding, subtracting, multiplying and dividing are to numbers for the computer. Often strings are loaded into string arrays, which are tables of non-numeric information. LOGO is unique among computer languages in treating letters and numbers alike through List-processing.

TEMPLATE PROGRAM A program which is easily varied by just changing key data. For example, a flashcard program which can be made into a new program by re-writing a single line is a template program.

TEXT EDITOR Program which edits (formats, manipulates, prepares, changes) any input; the editing part of a word processor. LOGO has its own text editor; other languages have editors which must be purchased separately and loaded with the language.

TUTORIAL A program designed to teach a concept, usually with sub-programs which branch if the learner needs further help in acquiring the behavior being taught.

USER-DEFINED FUNCTION Most programming languages allow you to set up numeric functions that you will use again and again. LOGO lets you define entire operations of both numeric and general use for a program.

UTILITY SOFTWARE General software for manipulating files (for example, editing, sorting, transferring computer files from one location to the next, or, in CAI, canned sub-routines which can be used from program to program).

VARIABLE Most readers can relate this concept to high-school algebra, where a variable stands for something which can change in value from problem to problem. Similarly, computers use variables to temporarily hole information which may change from program to program (like the user's name), or within a program (such as the number wrong so far).

WILD CARD CHARACTER A way of allowing more latitude in user response. A character is allowed to stand for anything in a match. If the program compares "cumputurs" with "c?mp?t?rs", the learner is given a correct score, but "Komputers" will be marked wrong.

WORD PROCESSOR An electronic device used to manipulate text. Probably the most powerful form of non-instructional CALL, a word processor allows more room for compositional creativity. A first draft may be easily revised, until a final version is acceptable to learner and teacher.

WRITE-PROTECTED A software program which cannot be changed or copied, as opposed to public-domain software.

Connecticut State Department of Education

Division of Vocational-Technical Schools

Angelo J. Tedesco, Associate Commissioner
and Division Director

Kenneth C. Gray, Assistant Division Director

Saúl Sibirsky, Consultant
Bilingual Vocational Education