You'd be Surprised at How Much Public Domain SoftwareYou Can Adapt to ESL and Language Learning

Vance Stevens, Hawaii Preparatory Academy

It is widely understood that the scope of commercial software currently available for learning and/or teaching languages is presently both overpriced and inadequate. It is less well known that there exist in the public domain numerous programs suitable for language learning/teaching which may be obtained free or for a nominal fee (see, for example, Biggie 1984). This article discusses how such programs may be obtained and how they can be altered for use in language learning/teaching situations.

It should be understood that few of the programs mentioned here are meant to teach discrete points of grammar, which has been the focus of the great majority of language learning software produced in the past decade. The latest trend in computer-assisted language learning (CALL) has been, alternatively, to provide students with an environment which facilitates their use of the target language in problem solving and in interaction, not only with the computer, but with each other (e.g. Higgins 1983; Higgins and Johns 1984). The purpose of most of these public domain programs, then, is to provide a stimulus for communicative activity in the target language.

Obtaining Public Domain Programs

The following are two sources for public domain programs which I have found useful:

Computer-Using Educators (C.U.E.)
333 Main St.
Redwood City, CA 94063

Honolulu Apple Users Society (HAUS)
Box 91
Honolulu, HI 96810

Two other sources which I have used more casually are noted elsewhere in this article. Still more potential resources can be found in Healey (1984). Finally, anyone wishing a copy of the programs mentioned here could obtain them by sending the author a 5 1/4" floppy disk in a stamped, self addressed disk-mailing envelope. The programs all operate on the Apple II family of computers.

All You Need to Know About BASIC in Order to Understand this Discussion

Merrill (1982), in an article about the relative merits of programming vs. authoring languages, notes that the major difficulty for educators in working with a programming language such as BASIC is that programming languages must usually be learned in ways not specific to language learning, and this makes them appear to be overly complex. More recently, attempts have been made to relate programming languages specifically to language learning (e.g. Kenning and Kenning, 1984). Here, I will further attempt to distill what must be known about BASIC into the few elements that usually have to be altered before a public domain program can be integrated into a specific language learning application.

The most useful type of CALL-adaptable public domain program is one in which the

target language material is stored in DATA statements, which are easily found and modified by other users. Items of data and /or other material are stored in string variables and operated on in ways that need not be understood by those adapting the programs. However, in altering such a program, you need to be aware of how the DATA statements and/or string variables function in the program, and of how these may be accessed in order to vary lesson presentation. The following is a very brief characterization of a few principles of BASIC which generally must be understood in order to successfully adapt public domain software to language learning:

(1) VARIABLES

Variables are physical locations in the computer where information is stored. There are two types of variables:

NUMERIC VARIABLES look like this:

A or AB or B1.

Numeric variables can be used in arithmetic operations. For example, if the value of the numeric variable that keeps count is greater than the number of questions in your exercise, then you jump to the end of the program (where END is at line 2510), like so:

```
1000 LET R = R + 1
1005 IF R > 15 then 2510
```

STRING VARIABLES look like this:

A$ or AB$.

String variables hold strings, or text of some kind. The "value" of a string variable is considered to be whatever its contents are. Contents of the variable are set off by quotation marks. For example:

```
340 INPUT "Again?  (Y/N) "; YN$
350 IF YN$ = "N" then 2510
```

ALL YOU NEED TO KNOW is how to recognize the two kinds of variables.

(2) The PRINT Statement

The computer prints things on the screen according to instructions given it in PRINT statements. Typical syntax for PRINT statements is:

```
10 PRINT "Print this:  ";X$
```

In this example, 10 is a line number, PRINT is a command, the text to be printed is placed in quotation marks, and X$ is the value of a string variable, which will also be printed. ALL YOU NEED TO KNOW is that you can change what gets printed on the screen just by changing the text between the quotation marks or stored in the variable.

(3) The DATA Statement

The easiest programs to change are ones which utilize DATA statements. These statements contain information which is fed into the program in response to a READ statement. A DATA statement might look like this:

```
25 DATA Zebras, Stripes, Giraffes, Spots
```

In this example, 25 is a line number, DATA is the command, and separate items of data are separated by commas.

The computer has to know one of two things about data in DATA statements. It has to know either how many such instances

of data there are (especially if the data are accessed randomly), or where the last item of data is (if the data are accessed sequentially).

ALL YOU NEED TO KNOW is that you can usually delete existing DATA statements and type in your own. Usually, it doesn't matter what the line numbers are or how many items of data are in a line, as long as you pay attention either to how many items you have all tolled, or to what the last data item is.

(4) ARRAYS

Sometimes, it is convenient to consider values in a program to be grouped together logically in some way. You store such values in array variables. For example, suppose you want to randomly select distractors for a multiple choice problem from certain parallel items in your data base. You might have the computer generate a random value for R. The computer would go to a different place in your DATA base and randomly select distractors for you, depending on the value of R.

ALL YOU NEED TO KNOW is that the number in parentheses following a variable often bears some relation to some variable in your program, such as the number of items in your data base, or the number of questions you are planning to have.

Some Public Domain Programs Useful to Language Learning

The following are some of the programs which I have found most useful in my own ESL program at Hawaii Preparatory Academy, and which I assume would be useful in ESL and foreign language programs elsewhere. The source, along with the disk identification decipherable by that source, is given in parentheses after the name of the program on that disk.

Easily Modified Programs:

SCRAMBLED WORD (C.U.E. #2) is a program that takes words in its DATA statements and presents them scrambled on the screen. The students' job is to figure out what the ergbaga (that's garbage) on the screen is. You can easily alter this program just by deleting or changing the DATA line. Within certain limits, it doesn't matter how many lines there are, or how many items there are, or how many items there are in a line, or even what the line numbers are. Four to six letter words work best.

Also, you should be aware of one characteristic of C.U.E. software. When the program runs, it ends by running a catalog program. If your disk does not contain the catalog program (called MENU), then change the appropriate line to read END, or alter it to issue another DOS command. (Here, the appropriate line is 715; END is safest, because if your DOS command runs another program, and you haven't saved the current version of this program, you will lose it during practice runs.)

HANGMAN FOR ONE (C.U.E. #1): This is a low-resolution hangman game. When you get the word right, it says "You live . . . for now." The program has acceptable graphics, and you can change DATA statements to insert your own words. There also exists on this disk a version in French. As with SCRAMBLED WORD, you can delete or change any or all data lines, but here, the first piece of data has to be the number of items in the data base.

STATES AND CAPITALS (C.U.E. #1): Although this program could be used as is in a geography class, you can play with it

to create your own programs. This is worth doing, because the program takes two associated words or concepts (A and B, or in this case, a state and its capital) and gives the learner the choice between doing a multiple choice or fill in the blank exercise. It then allows the learner the choice of having concept A either in the questions or in the answers. So, from one data base of paired items, you derive four exercises in one program.

To ALTER this program, you need to decide how many questions (associated pairs) you will have, and then find and set all the arrays accordingly. (A better way to do this is to change all the array values to a variable, say QB for "question base"; then each time you change the program, you have only to set QB in one statement to the appropriate value.) You will then want to change the various print and remark statements to match your lesson.

Finally, you will want to alter the DATA statements and, in the C.U.E. version, modify the exit subroutine as noted previously.

CRYPTOGRAM (C.U.E. #2) is an excellent program for language learning or for getting students to regurgitate concepts in any subject. It takes strings of any length, up to about half a screen, and substitutes a random letter for each letter in the original string. You get a cryptogram on your screen, and you have to use your ability to predict in order to elucidate the original message. The program feeds off DATA statements within the program, and there is also the option of inputing a string for someone else to unscramble while the program is running.

The program works by generating a random number that directs program execution to statements which are multiples of 10 in the 1010-1290 range. (The program also allows students to pre-empt the generation of a random number with one of their choice.) This is where it gets cryptograms 1 to 29 (middle two digits in the lines numbered 1010 to 1290; get it?). You can change these DATA statements to hold your own strings. If your string runs over a line, simply add line numbers between the first in that series and the next multiple of ten (for example 1120 to 1122, 1124 . . . up to 1129).

LUCY (HAUS #25) is a clone of Weisenbaum's famous ELIZA program. Both ELIZA and LUCY allow you to talk to the computer. The computer simulates conversation by responding to key words in your input; for example, the presence of a question word like "what" or "where" in your input might elicit the response, "Why do you ask?" from the computer.

I use this program by letting the students play with it for one session, having them try to figure out what the key words are the next, and finally having them write down the rules they have discovered that the program follows. Finally, I prepare a compilation of these rules and have the students test their validity.

All the computer's 330 responses are kept in a text file which you can read and alter. I have written a program called LDATA CHANGER% which allows you to alter the text files, and which I would be happy to share.

RANDOM DRILL TUTORIAL (HAUS #14) is a program which allows students to drill concepts which can be considered associated pairs (A associated with B). You alter the DATA statements in lines by replacing each line with your own

associated concepts, one pair per DATA statement. The program then presents the concepts in one of two modes. In its flashcard mode, the program prints concept A and then B after the student presses RETURN. In its multiple choice mode, the computer presents concept A and draws distractors from randomly selected B's. One problem with the program as it is available from HAUS is that it has no means of exit, or from going from one mode to the other.

SHELL is a program which lets an instructor set up a question base of about 25 blank-filler questions. The program then presents a preset number of those questions in random order, so that any two sessions at the computer are slightly different. The program allows the student two attempts to fill in the blank, and then presents the correct answer along with appropriate feedback. Feedback messages, both congratulatory and corrective, are randomly accompanied by pleasant or caustic noises, which can be turned off if students desire privacy. The program gives students the opportunity to rework problems they have missed, and awards 100% scores to students who do so successfully.

The program was presented by Laura Savely at the 1984 TESOL Convention in Houston, and the handout included a program listing and carefully detailed instructions for customizing the program. The program is available for $5.00 from:

Executive Offices
American Language Academy
11426 Rockville Pike, Suite 200
Rockville, MD 20850

"AS IS" Programs:

I find this next group of programs useful for ESL "as is." No modification is necessary. With adaptation, some may be useful for teaching/learning other languages also.

MAKING CHANGE (C.U.E. #8) shows coins and dollar bills in high-resolution to illustrate transactions using the words *dollar*, *dime*, *nickel*, *penny*, and *quarter*.

ARTILLERY (C.U.E. #1) is a game where two players each have a cannon, and each in turn tries to direct shells over a mountain at midscreen so as to obliterate the other. In order to win, you have to be able to more and more closely approximate the correct angle and force to be applied to the shell. Wind velocity is also a factor. This is realistic practice with trajectories, and the game generates a lot of interaction between students.

However, if given their choice, students gravitate toward this game--eventually at the expense of more directed language practice. So, although the game is useful to a point, I delete it from student disks after a few sessions.

ADD-LIBS (C.U.E. #1): You are prompted for adjectives, nouns, etc., and the computer plugs your choices into a story. This is great for ESL students, who could be introduced to the program innards and encouraged to make up their own stories.

In my brief experience with this program, I have found that students sometimes attempt to use words of sexual and scatological connotation. Other groups may have better success, but this is something for the teacher to be aware of.

MEET THE ROMANS (C.U.E. #1) drills Roman numerals using the device of a game. With two players competing, the

computer gives each in turn a Roman numeral, which the players must write in Arabic numerals, or an Arabic numeral, which the players must transpose into Roman.

One often-overlooked deficiency in the inventory of skills of foreign students is a working knowledge of Roman numerals. This program encourages students to learn Roman numerals in an interactive setting:

CHRISTMAS TREE SONG (C.U.E. #1) draws a tree on the screen and plays a verse of "Oh Christmas Tree, etc." The word corresponding to each note is displayed as the note is played.

DECK THE HALLS (C.U.E. #1) plays several verses of the familiar carol. The word corresponding to each note is displayed as the note is played.

It is often nice to introduce students to items of American culture appropriate to the season. The above two programs are useful for this purpose at Christmas time.

Programs with Potential:

I have not used the following programs yet, but I think they have potential for ESL:

STAR LANES (C.U.E. #1): This is a complex, but not indecipherable, game, whose rules were not all apparent to me at first. Basically, you are given options for placing your piece on a grid. The idea is to get pieces next to each other, forming trading entities or merging with existing entities, thereby driving up the value of stock in those entities. Thus, the language of the game is that of business and economics. It would probably stimulate interaction among young learners, as well as lead to discussions of the economic principles involved.

CINQUAIN (C.U.E. #2) helps you write a poem in cinquain form. It prompts you for each line according to specifications, and prints out the final result, which should be pleasing to the ear. Also, using the algorithm here, it should be possible to prompt students for poems in other formats.

CALIFORNIA DRIVING TEST (C.U.E. #2) asks 60 multiple choice questions very similar to those on a driving test. Instant feedback makes this an improvement over the test in the manual. With some small effort, this could be adapted to the driving tests for other states. I find ESL students highly motivated to study such material in order to get their driver's licenses. Last year, the Hawaii Driver's Manual was part of my course.

IQ TEST (C.U.E. #7) displays text in various configurations on the screen. For example, it writes *man* and below that *board*, and you are supposed to think, "man overboard." As it is written, this program just displays several instances of such text. It could easily be altered so that you have to type in the answers. Altered thusly, it could be fun in an ESL class.

CLOUZOT (C.U.E. #7) is a take-off on the bumbling detective, Inspector Clouseau. You are given suspects A through F (the first letters of the suspects names). When you press one of these letters, CLOUZOT "interrogates" that suspect--actually, a bit of information is printed on your screen. This information is in the form of a characteristic either of one of the suspects or of the murderer. The guilty suspect may be lying. Taking all this into consideration, you are to figure out which of the suspects is the

murderer. When you think you have solved the mystery, you "accuse" one of the suspects, and CLOUZOT tells you if you are right or not.

JOTTO (C.U.E. #7) is a variation on a game often used in foreign language and ESL classes. The computer "thinks" of a five letter word in which no two letters are alike. You type in your own five letter words, and the computer compares them with its word and reports back how many letters coincide between the two. You have 20 chances to guess the word. The computer keeps track of all your guesses and the number of coincidental letters, and allows you after each guess to use the computer as a scratchpad by recording on the screen letters you know are coincidental, and deleting from an alphabet on the screen those letters you know are not.

I have several suggestions for altering this program. One problem with the existing program is that the computer assumes any five letter input is a word, and so students can type in "ABCDE" and systematically beat the computer without actually exercising their ability to recall words in the target language. However, the program could be changed so that the computer could search through its own DATA statements and test student input against what it knows to be legal words. The only problem here is that there would be words that are legal but that the computer doesn't "know." One solution to this might be that these words could be written to a text file on the disk, and the programmer could come along periodically, read the text file, and add those 5 letter inputs which were indeed legal words to the data base. (By the way, the program already tests for the length of the word, and to see if any letters in the word are the same). A final alteration I would make is to relegate the scratchpad function to a subroutine and ask users whether they want to invoke it or not.

PIZZA (C.U.E. #8): You take orders for pizza and tell the delivery boy where to deliver. The phone rings (via the Apple speaker). You learn that someone is calling to order a pizza. They tell you who they are, and you consult the map on your screen to see where they live. All the people happen to live at houses numbered by X, Y coordinates. The delivery boy then calls in to ask where he should take the pizza, and you give him the proper coordinates, If you get it right, the person calls back and thanks you for the pizza. If you get it wrong, the people whose house you sent the pizza to call in and say they didn't order a pizza. If you send the pizza to several wrong houses, then when you finally get it right, the person calls back to say the pizza is cold. Clever.

Unfortunately, bad input bombs the program. This should be easy to correct. Try at about line 850, where the program compares your input with the locations it knows. Debugged, this programed would be fun for ESL.

ZORK is a well known adventure program. A listing for a simple version of ZORK was published in *Softline*, March 1982. The significance of the version presented here is that it provides an algorithm for parsing simple imperative sentences in English. For example, the program published here allows the parsing of two-word 'sentences' such as "open box"; but using the algorithm provided, I was able to modify the program to check for the complete sentence, "Open the box." I have not myself gone beyond this modest beginning, but I recognize that much of the groundwork has been laid here for more elaborate adventure games for ESL. For a copy of the article, write:

Softline
11021 Magnolia Blvd.
North Hollywood, CA 91601

Conclusions

Although commercial publishers have, to date, been slow in responding to the demand for reasonably priced CALL software, this is no reason for language instructors to delay CALL implementations in their language classes. Alternative sources are readily available and can be adapted to specific language learning situations. It has been shown in this article where such programs can be obtained and how they can be modified using only the principles of BASIC programming detailed here.

References

Biggie, Louis. 1984. Public domain software. *TESOL Newsletter* 18 (3) [June]: 11.

Healey, Deborah. 1984. Free or cheap sources of information and material. *TESOL Newsletter* 18 (4) [August]:15.

Higgins, John, and Tim Johns. 1984. *Computers in language learning*. Reading, Mass: Addison-Wesley (Co-published with Collins, Ltd., London, 1983).

Kenning, M. J., and M. M. Kenning. 1983. *An introduction to computer-assisted language teaching*. London: Oxford University Press.

Merrill, Paul E. 1982. The case against Pilot: The pros and cons of computer-assisted instruction languages and authoring systems. *Creative Computing* 8 (7) [July]:70, 75-77.