Title	Choosing call software
Author(s)	Chew, Phyllis Ghim Lian
Source	*Teaching and Learning, 7(2)22-26*
Published by	Institute of Education (Singapore)

CHOOSING *CALL* SOFTWARE

PHYLLIS CHEW

In 1960, a flurry of excitement surrounded the advent of the language laboratories as a means towards more innovative and successful language teaching. Today, the latest technological device which has made itself available for language teaching and learning is the microcomputer. Although its real onset can be dated to only as recently as 1980, CALL (Computer-assisted language learning) is quickly establishing itself as a new and potent weapon in the language teacher's armoury.

Like the Language Laboratory, the computer is neither good nor bad in itself. It is only as good or bad as the stuff that is fed into it. Unfortunately, the language laboratory was associated with software which were mainly behavioristic drills and quickly fell into disfavour in the sense that it took a very long time before teachers learnt to use it imaginatively. For a long time, users of the language laboratory were fussing about the advantages of the three-phase drill over the four-phase drill-and vice versa, not realising that there were much more they could do to make laboratory work more meaningful, interesting and stimulating. It would take close to twenty years before teachers began to experiment with more interesting and imaginative exercises in for example, listening comprehension, for use in the language laboratory. In modern language laboratory materials, there is a consciousness by producers to keep dreary drills to a minimum and to at least put them into context. Now that the microcomputers are here, one may wonder whether it will go the same way as the language laboratory and whether it will take us that long to become creative users? I hope not, but the situation is worrying. In order not to repeat the mistake made by the language laboratory, language teachers have to be conscious of some kind of guiding principles in the choice of software for use by their students. This is especially important, bearing in mind future expansion of computer-assisted learning in Singapore schools.[1]

Before reviewing these principles, we should be aware that there are presently two camps in terms of CALL methodology: the "tutorial" camp and the "non-tutorial" camp. The first sees

the computer as essentially a tutor helping students acquire linguistic competence through drill and practice of discrete points in morphology, syntax and semantics. Some examples of such programs are "English Achievement" (I–III), "The Antonym Game" and "Wizard of Words".[2] Based on the belief that linguistic competence precedes communicative competence and must be attained before real communication is possible, tutorial programs are basically grammar-based learning.

On the other hand, the non-tutorial camp feels that the only valid approach to language learning is the communicative one. It advocates programs which are essentially restoration of mutilated text such as "Storyboard", and language games such as "Word Snap", simulations and problem solving activities such as "Lemonade" and dialogue type games such as "Eliza" and "Chatterbox".[3] The keywords are communication and integration of skills. There is emphasis on programs with whole language and whole text. The computer becomes the focus of group discussion and the machine becomes the means whereby a group of students seated round it can discuss, hypothesize or argue around a given situation depicted in a program.

Although some people may argue that we should only choose non-tutorial software, I believe this is too absolute a stand to take. Tutorial software can serve a purpose and has proven to be useful to some students. Much as I admire the European-based communicative approach, I believe too that a wide variety of supportive materials and tools should be made available to teachers. If we believe that language teaching is an art, dependent on the skills and talents of the practitioner, it would not matter whether we choose software from the tutorial or non-tutorial camps. The thing to bear in mind is that there are "good" and "bad" programs from both camps. Choice of software should depend on other more essential factors, not just whether the program in question is tutorial or non-tutorial. What then should be the criteria or guiding principles in our choice of language software?

Perhaps the most important thing to remember about effective language-oriented CALL is that a computer is not a book. Many early CALL programs were essentially "page

turners", that is, the student would read the question, type in a response and the computer would indicate whether it was right or wrong, with no further information. The student then turns the "page" to the next question. This kind of program is essentially an exercise book in disguise. Any workbook with answers printed in the back can do the same thing less expensively. By using such programs, we are therefore under-estimating the role of computers since they are capable of more imaginative, challenging and powerful procedures, such as in adventure stories when, for example, the "player" becomes a protagonist and the computer responds to his commands.

The next important point to look for is to see that the error checking procedures of a particular program is comprehensive and its comments to the students non-threatening. This is especially important for tutorial-type programs. The program should be able to distinguish between involuntary errors (that is, hitting the wrong key, forgetting or adding space and periods) and an actual error. After all, programs should teach language, not typing skills. Meaningful errors should elicit appropriate comments which guide the student to correct his own answers and to learn from the mistake. One common but serious objection to most programs is when there is more than one correct answer and this is not indicated by the computer. In the following, for example, students were asked to continue the clause by subordination:

Peter became worse. He did not take his medicine. (because)

The only correct answer appeared as:

Peter became worse because he did not take his medicine.

However, we know that there are a number of other possible answers:

Because he did not take his medicine, Peter became worse.

Because Peter did not take his medicine, he became worse.

Peter did not take his medicine because he became worse.

In this respect, the ideal program is one which has made use of an experienced teacher to design the error-checking routine, as only such a person can foresee the typical anticipated error.

Next, a good program is one which respects individual learning strategies to the greatest extent possible. We must remember that computers are ideal for individualized learning tasks and allow students to pace themselves in a way which is not possible in any other medium. Options must thus be given to students to proceed at their own speed, to request for remedial help and to permit them to skip ahead or to repeat a section of the program with ease. Here, programs should exploit the computer "menus" in such a way as to enable the user to easily choose what he wants to do. An additional individualised tool is the built-in dictionary for words in the exercise, if the teacher judges that to be useful.

An additional point to bear in mind is that, since the best computer games use a system of rewards (high scores, brilliant graphics, etc.) to motivate, CALL programs should also follow suit. Brilliant graphics should be taken advantage of. However, where scores are concerned, one must remember that some students are put off by a score being kept, even if the teacher has no access to it. For one thing, it removes one of the main positive features of computers: the privacy to make as many mistakes and learn from them without the rest of the class or the teacher sharing in the embarassment. On the other hand, some students like to know how well or badly they have fared and to compare their present scores with the previous one. One solution here would be for a program to give the student the choice of seeing their scores. Whether scores are good or bad, it is obvious there should be a feeling of satisfaction for the student from having completed the activity. Some kind of congratulatory message or tune from the program in question would act as a good motivator.

Last but not least, since our students are likely to be inexperienced users of the device, one should see that programs make conventions for formulating answers to the computer as clear as possible. They should take advantage of flexible matching procedures for checking answers. Generally, programs should avoid lengthy typing with student answers so as to decrease the likelihood of error. They should be easy to start and allow the student the choice about when to exit from program.

To conclude, it is important for us to be constantly aware of the main guiding principles in the choice of software; otherwise, we might just repeat the mistake made with the language laboratory. This consciousness on our part would significantly encourage higher quality from software writers and producers, and contribute collectively towards making CALL a more enjoyable and more profitable experience for teachers and learners.

1. The move in 1980 to introduce computer education focussed initially on two areas: Computer Science as an 'A' level subject and computer appreciation courses conducted through Computer Appreciation Clubs (CACs) at the secondary level. There are indications at present that the new objectives for computer education in schools are likely to include not just the above but computer literacy courses for all, as well as computer-assisted learning.

2. These programs run on the Apple machines and are available from the Computer Science Resource Center (CSRC), Curriculum Development Institute of Singapore.

3. These well-known programs run on the BBC and Spectrum TS2000 machines. "Lemonade" and "Eliza" are also available on the Apple system.