Approaches to the design of language teaching software

Martin K. Phillips

It is argued that there is a need in the field of computer-assisted language learning to develop a principled framework which will enable current activity to be related to an appropriate theoretical perspective and to suggest avenues of future development. A comparison of two different types of CALL programme leads to a discussion of some of the possible elements of such a framework.

Daar word 'n argument voor uitgemaak dat 'n verantwoorde raamwerk ontwikkel moet word op die gebied van rekenaarondersteunde-taalonderrig. So 'n raamwerk sal daarvoor sorg dat teenswoordige aktiwiteite verband hou met 'n toepaslike teoretiese perspektief en dit sal moontlikhede open vir toekomstige ontwikkeling. 'n Vergelyking van twee verskillende rekenaarprogramme vir taalaanleer lei tot 'n diskussie oor enkele moontlikhede vir die ontwerp van so 'n raamwerk.

Computer-assisted language learning is in a state of ferment. After many years of relatively unexciting experimentation using mainframe and minicomputers, which gave rise to such developments as the TICCIT (Time Shared Interactive Computer-Controlled Information Television) and PLATO (Programmed Logic for Automated Teaching Operations) systems and the British National Development Programme for Computer-Assisted Learning, a fresh impetus has been given to the field through the recent widespread availability of relatively cheap, high-quality microcomputers. Teachers are becoming involved in the exploration as never before. A recent publication encourages them to take up programming in BASIC (Beginners All-Purpose Symbolic Instruction Code) (Kenning & Kenning 1983); another suggests a variety of innovative ways in which the computer can be exploited in the classroom (Higgins & Johns 1984); in the British Council we have recently launched a project to introduce CALL (Computer-Assisted Language Learning) into our overseas teaching operations. Creativity is rife and the prospects are exciting.

There is, however, a price to be paid for this ferment. This is a certain fragmentation of development as individuals explore their ideas in the relative isolation that the microcomputer permits. I do not oppose this development. Indeed, I believe it is profoundly liberating and an essential prerequisite for the healthy growth of the field. But there is also a need to stand back from the immediacy of programme design in order to arrive at a more comprehensive view of CALL and to determine the principles upon which coherent development can take place. If we do not look at CALL in a theoretical perspective, there is a danger that the field will be discredited solely on the basis of shortcomings in particular implementations. Many programmes exist which are individually interesting; but the one-off success may not be sufficient to guarantee the viability of the field. It is therefore necessary to reflect upon the experience of successful programme design to see what generative principles for the development of CALL can be extracted. Only in this way will the potential of the field be translated into accepted practice.

Thus a framework is needed for thinking about the development of CALL. This framework should provide criteria so that decisions can be made about priorities and emphases. It should

offer a coherent overview so that individual developments can be related clearly to the curriculum context in which they must exist and to each other. It should suggest the lines upon which standards can be elaborated. This framework does not yet exist. What I propose is to take a preliminary look at what might be involved in arriving at such a framework by examining two CALL programmes in some detail for the insights they may offer into developmental principles.

These two programmes have been produced on an experimental basis under the auspices of the British Council. They are intended as an English language learning resource, though of course the principles involved are likely to apply to a large number of languages. The first is a programme called *Letter-shoot* and the second is called *Finder*. Both programmes are designed to run on the BBC Model B microcomputer with disc drives. This is the computer which has been adopted as the standard machine for the first phase of our project.

Letter-shoot takes the form of an arcade game for one player. The student controls the lateral movement of a cursor. The screen displays a random selection of letters in random locations. The display scrolls down the screen at a speed governed by the level of difficulty chosen. The objective is to compose words by manoeuvring the cursor to "hit" letters as they scroll past. The student can choose to make words in a number of different conceptual categories, for instance, "Animals", "Colours" and so forth. The computer keeps a running check of the state of each word being composed and credits the student as soon as a word for which there is an entry in its lexicon has been completed. An incorrect "hit", that is, hitting a letter which is not a possible continuation, aborts the current run of the programme. A further option offered by the programme, which is particularly useful for students whose native language does not use the Latin script, is the facility to hit letters in alphabetical order or to hit upper and lower case pairs. Thus *Letter-shoot* aims to harness the motivation and excitement of the *Space-Invaders* type of game to the tasks of letter recognition, spelling and vocabulary building.

Finder adopts a quite different approach. This programme presents the student with a problem situation. It is assumed that the student has lost his three suitcases which he can only recover if he succeeds in drawing them. This he does by giving the computer instructions to produce the required graphics. The student can control the colour, height, length and width of the suitcases displayed on the screen. The computer randomly generates the target configuration but this is not displayed. At any time the student has a number of options for comparing his efforts against this virtual target configuration. Using the clues provided by the computer each time he asks for a comparison, it is possible for the student by a process of elimination, to narrow down the range of variation for each of the suitcases he is drawing and finally to arrive at the correct configuration. In the course of doing so, the student will have practised intensively a number of comparative forms and have both given commands to the computer and interrogated it for relevant information.

It will be instructive to compare these two programmes. If it proves possible to determine the critical dimensions along which the programmes vary, it may be feasible to see any given programme as generated by the combination of particular values of a number of abstract variables. This will then represent an important step towards establishing the elements of the framework which is needed to guide our thinking about CALL.

We must thus attempt to answer the question, in what significant ways do these two programmes differ? The first answer which suggests itself is that they generate different types of activity. *Letter-shoot* adopts a game format whilst *Finder* is what may be called a problem-solving programme. Thus activity type seems to be a useful discriminating category. Programmes which adopt a game format may, for example, be adaptations of the now familiar "arcade" type game, as is *Letter-shoot*, or of the more strategic type of board game. In both cases, however, the point is that the objective of the game as exploited in CALL is attained by the exercise of a linguistic skill rather than or in addition to a motor skill. Malone calls these "extrinsic fantasies" because the activity does not depend on the skill (Malone 1982). The same activity could equally well be used to practise a non-linguistic skill, for example, mathematical operations. Similarly, problem-solving activities, such as that exemplified by *Finder*, typically set the student a problem of a logical nature involving the use of language in reaching a solution.

It is possible to identify a number of other activity types. We are all familiar with the type of programme which manipulates text in one form or another. Some programmes generate a task by analysis of a text drawn from a pre-existing database of texts. The student's task is to recover the text which has been "degraded" in this way. I call these text-reconstruction programmes. *Storyline*, another programme developed under the auspices of the Council's project, is one example. Other programmes invite input from the student in order to construct discourse which may take the form of a simulated conversation or a question and answer session. In this case the computer is generating the task by synthesising it from initially discrete linguistic elements. These may be called text construction programmes.

Tim Johns first drew the distinction between what he called analytic and synthetic generation (Johns 1981). An example of a programme based on the principle of synthetic generation would be *Chatterbox*, an adaptation to language teaching of the Eliza programme.

Other activity types which suggest themselves are simulations, exploratory programmes and, of course, quizzes. Quizzes were perhaps the first kind of language learning activity implemented on computers and still have a role to play. More recently, however, attention has turned towards trying, in a limited way, to apply artificial intelligence techniques to the design of CALL programmes, where the computer is programmed with a limited knowledge of a restricted aspect of reality. Such simulations thereby involve the student in decision-making tasks using language. Malone has referred to this type of programme as "intrinsic fantasies", because the existence of the situation modelled by the computer depends on the linguistic skills practised. John Higgins' *John and Mary* represents a limited implementation of this approach.

Finally, exploratory programmes are those in which the computer is provided with a set of rules for a fragment of the linguistic system. The student has the task of devising suitable examples with which to explore the limits to the computer's "knowledge". A number of such programmes exist, mainly focusing on word morphology. It is likely that activity type is a productive category and that further experimentation will suggest other possibilities.

It is obvious that programmes can differ in the linguistic learning point embodied in the activity they generate. *Letter-shoot*, for example, is concerned largely with word lexis and spelling whilst *Finder* practises a number of syntactic structures and comparative forms of the adjective in particular. Thus different programmes have different linguistic foci; this feature may be called programme focus and clearly represents an important element in any generalised framework for CALL. But *Letter-shoot* and *Finder* also demonstrate that the programme focus may not correspond to the way in which the student perceives the programme. In *Finder*, for example, the task recognised by the student is not that of practising comparative forms of the adjective, but rather the more meaningful one of attempting to solve the logical problem posed by the computer. Similarly, the task in *Letter-shoot* is interpreted as the competitive one of maximising one's score. In other words, programme focus need not coincide with learner focus. Indeed, it can be argued that it should not and that one of the strengths of the computer is the possibilities it opens up for reducing what may be called the "inauthentic labour" of language learning whilst increasing the opportunity for "authentic labour", that is, work which has a significant non-linguistic outcome (Kemmis 1977). Learner focus thus appears to be another useful category for thinking about CALL.

If the programme focus can differ from one programme to the next, this implies that the difficulty of programmes can differ. Difficulty as such, however, is a rather undifferentiated and therefore unhelpful notion. Again, it can be explicated by reference to *Letter-shoot* and *Finder*. It is clear that the two programmes differ in terms of the intrinsic difficulty of the language used. *Letter-shoot* is principally a matter of vocabulary, which may or may not be known, but which does not present any intrinsic conceptual problems. *Finder*, on the other hand, involves appropriate use of a variety of comparative structures and two types of question form. It could be argued that linguistically *Finder* is inherently more difficult than *Letter-shoot*. But it should also be evident that this difficulty is adjustable precisely because it is a function of the linguistic content. Clearly, the difficulty of *Letter-shoot* depends to a certain extent on the particular choice of vocabulary incorporated in the game whilst *Finder* could include a number of more sophisticated turns of phrase than

those currently implemented (for example "Lengthen my red suitcase!" "Is my blue suitcase as long as yours?"). What emerges from this discussion is a notion of language difficulty for CALL.

The foregoing considerations also indicate that the activity generated by a programme can be conceptualised independently of any particular linguistic content. This suggests that there is another dimension of difficulty pertaining to the design of the programme as such rather than to language. Again, the comparison between *Letter-shoot* and *Finder* is instructive. *Letter-shoot* is designed so as not to require any linguistic input from the students. Words are composed by moving a cursor controlled by cursor keys on the computer keyboard. *Finder*, however, demands linguistic input: the student has to type in appropriate sentences and to this extent *Finder* is more exacting than *Letter-shoot*. Note that this type of difficulty is not a function of the linguistic content but of the design of the programme as a programme. It is entirely possible to conceive of the manipulation of letters in *Letter-shoot* being executed in response to typed commands from the student whilst *Finder* could be so arranged that the student selects input from a range of alternatives presented on the screen and thus never has to type more than a single letter or number. Neither alteration would affect the difficulty of the linguistic focus of the programme as such but clearly what may be referred to as the programme difficulty is changed.

There are other ways in which programme difficulty can be adjusted. *Finder*, for example, insists on accurate input from the student; there is strict matching of input against the language forms accepted by the computer. This constraint is not, however, a necessary one. It is quite possible to incorporate input routines involving fuzzy matching, as Johns' "intelligent spelling checker" demonstrates (Higgins & Johns 1984). In certain circumstances, the limitation of a restricted input set can be relaxed virtually altogether, either through keyword matching techniques of the sort used in Eliza-type programmes or through input parsing. Another obvious way in which programme difficulty can be controlled is where a timing element is introduced; obviously, where the student is working within a time constraint, the limit can be more or less liberal.

Another way in which programmes can differ is in the ways in which they can be exploited in the teaching situation. *Letter-shoot* for example, is clearly intended for use as an individual exercise, though it could conceivably be used as a competitive team activity. It would not be particularly effective as an individual activity since the element of strategy would thereby be lost. In contrast, *Finder*, whilst feasible as an individual activity, since its problem-solving nature is suited to solitary reflection, is perhaps best exploited as a group work or classroom exercise. It seems to be a characteristic of many programmes that they are exploitable in a number of different learning configurations with the principal distinction being between programmes usable by the individual and those which are not. Thus a further element in the framework for thinking about CALL is the category which I call classroom management.

Finally, it is possible to ask whether different types of programmes lead to qualitatively different styles of learning. In what way, for example, does the learning experience generated by *Letter-shoot*, differ from that produced by *Finder*? Different programmes induce different learning styles, which may be more or less complex. The principal dimension of difference among learning styles is the extent to which the learning which takes place is dependent on the context created by the programme. The simpler learning styles are context-dependent, that is, very much tied to the particular tasks in hand. The more complex styles involve learning which is context-independent, learning which is generalisable from the immediate programme task and therefore applicable in a wide variety of situations. The most basic learning style is simple recognition; the student is required only to recognise linguistic items as they are presented without any further creative involvement. It could be argued that this is the type of learning stimulated by *Letter-shoot*. *Finder* is an example of a programme fostering a more sophisticated learning style. Here the learner actively explores a field of linguistic behaviour and the programme presents a model of language use with which the learner is encouraged to experiment. This could be called "experiential learning". Other programmes will promote recall, comprehension or constructive understanding; in the latter, the learner draws upon his linguistic resources to create novel language events. It is clear that the computer can be used to promote learning in ways far

removed from the habit formation of programmed instruction associated with early attempts at computer assisted instruction and that learning style is an important category in any conceptual framework for CALL.

The preceding discussion has identified a number of useful analytical categories; activity type, programme focus, learner focus, language difficulty, programme difficulty, classroom management and learning style all seem to be necessary and helpful categories with which to approach the design of software. It may well be that other categories can be added to the analysis. One suitable goal for the development of CALL would be to explore what further categories might be needed. Nor have I considered the issues raised by the interactions among categories. It should be evident, however, that a given programme can be considered as a nexus of values taken in a particular instance for each of the categories considered here. *Letter-shoot* and *Finder* represent two different configurations arising from the interaction of the same set of categories but with different values in each case. This suggests that a fruitful approach to CALL software development would be to experiment with different combinations of values. One advantage of the conceptual framework which at the outset I suggest was needed would be its use as a heuristic device. Such a framework, incorporating the categories I have outlined here or similar ones, could encourage development by suggesting logical programme types. It is up to our imagination as teachers and our ingenuity as programmers to see how many of the logical possibilities can be translated into viable programmes and stimulating learning environments.

Bibliography

HIGGINS, J. & JOHNS, T. 1984. *Computers in language learning*. London: Collins.

JOHNS, T. 1981. The uses of an analytic generator: the computer as teacher of English for specific purposes. *ELT Documents* 112.

KEMMIS, S. 1977. *How do students learn?* Norwich: Centre for Research in Education.

KENNING, M. & KENNING, M.M. 1983. *An introduction to computer-assisted language teaching*. Oxford: Oxford University Press.

MALONE, T.W. 1982. What makes computer games fun? *Computers in Schools* 7.