COMPUTERS AND LANGUAGE LEARNING

John Moore

Have you ever dreamed of a new kind of textbook? A book which you can change, for example, whenever you feel like it by putting in a new page, a new line or just a different word. Or a book which has moving pictures – or better still, pictures which you can make move in any direction and at any speed you want. What about a book which can make sounds? Or a book that can check your answers, keep a record of your progress and give you encouragement? A book that can answer your questions, respond to what you tell it, create or analyze texts for you? A computer can give you all this – and lots more – but it is, in effect, a kind of superior book, not a miracle. I have no doubt that, thanks to the availability of microcomputers in the home and office, computer-assisted learning is about to gain a new lease of life and, probably a radical change of direction. It is particularly important that in this formative stage, teachers should be aware of what computers can and can't do so that the computer can become a real aid, making use of sound pedagogical principles, rather than a plaything in the hands of educational salesmen. The sophisticated materials which computers are able to handle, are, in fact, based on quite simple principles and an understanding of these will quickly dispel the idea of the computer as some kind of mysterious machine – and which conceals its secrets to all but the few initiated followers. It is these simple principles that enable computers to do all the wonderful things they are capable of – because there is nothing already built into computers which enables them to, say, answer your questions and solve your problems. The computer can only do what it is told to do and in order to know what to tell it to do, you have to know what it can understand and what rules it can operate. Tim Johns (1) expresses this very well when he says: 'The most powerful myth of the computer is its all-knowingness. It is not an accident that one of the most famous of all computer programmes, Weizenbaum's Eliza, associated this image of the computer with that other twentieth-century stereotype of omniscience, the psychiatrist: nor is it accidental that it performed its side of the dialogue by means of trickery rather than as a result of real understanding. This paradox – the contrast between the appearance of total competence and the reality of the means by which that appearance is achieved – is characteristic of much computer-assisted instruction.'

So let's first look at what the computer can do, what it can offer the language teacher. The first thing the computer can do is present material – it can use graphics, moving pictures or text, to display information, convey meaning or just make material look attractive. For example, a commerically-available computer programme for learning Chinese shows pictures of mountains, moving horses and so on and then the characters that are associated with them.

Secondly, the computer can provide practice – it can ask you questions, give you problems to solve or engage you in conversation. One of its main advantages in providing practice is the quantity of examples, questions, etc. it can provide. A teacher on the spur of the moment may not be able to think of enough examples but a computer can make a selection from an enormous bank of data. It can even make this selection randomly so that the lesson can be different every time it is used. Simple programmes can be written so that the computer initiates a conversation, for example by asking a question, and then responds to what you say. Of course the way it responds is pre-determined but with a little ingenuity the programmer can provide the machine with sufficient flexibility to create realistic dialogue with the student, the user. The practice a computer can provide or stimulate can also take the form of a quiz or game. The quiz offers interesting possibilities because the computer can either ask you the questions or you can ask the computer questions – there are even programmes in which rather than the computer teaching you, you are given the opportunity to learn by teaching the computer (2). An enormous number of games are now available for computers – rather like the games of spaceships, monsters and dragons etc. that you can play in the video arcades – some of which can be easily used to stimulate language use by providing opportunities for genuine problem-solving communication (3).

Thirdly, the computer can monitor your practice – it can keep track of your errors, correct them or provide remedial practice, give you advice and encouragement or select an appropriate difficulty level for you. Here again, the computer offers an important potential advantage over the teacher – it has infinite patience and not only has no difficulty in identifying incorrect answers (4) but will give whatever encouragement or admonitions have been planned even in the most tiresome circumstances. This is not to suggest, however, that this dogged execution of a plan can be a substitute for the spontaneity and responsiveness of the teacher – the computer, for example, can't tell when its leg is being pulled. The potential advantage which the computer can offer is rather that of relieving the teacher of much monotonous practice and at the same time providing the student with immediate feedback.

Lastly, the computer can be used as a testing aid – in other words, it can not only monitor the student's performance for the benefit of the student himself, but can present items, evaluate the responses and keep the score for the benefit of an assessor.

So, how is it possible for the computer to do such wonderful things? Very simply, the computer is a device for processing information – it accepts information and instructions, stores and/or modifies the information it is given, and displays the result. The kind of processes the computer can perform include storing information, searching for and locating information, adding to, subtracting from and changing text as well as a range of mathematical operations which can be used indirectly in language operations. (5)

A basic micro-computer system consists of a keyboard, a central processing unit (often part of the same piece of equipment), a screen like a television monitor on which information is displayed and a disk drive (or ordinary cassette recorder) to feed in and store programmes. An ordinary television set can be used in place of a special monitor.

For the teacher, the most crucial thing to know about is the programme – for it is the quality of the programme on which the quality of the learning depends (just as is the case with language laboratories, for example.) A computer programme is a set of instructions to the computer, telling it what to do, when and how to do it, and also what to say. Programmes can be non-interactive (that is, not allow the user to intervene), student-paced (the user controls the speed), student-directed (the user controls which part of the programme will be used) or interactive (that is, the computer will respond to what the user types in).

Before looking at how programmes are constructed, let us look at two very different types of language teaching programmes – different in terms not only of the construction of the programme but, more fundamentally, in terms of the assumptions about learning and the role of the computer on which they rest.

The first type we will call instructional, or tutorial. The second type is non-instructional, or non-tutorial. Instructional computer assisted learning is what held sway in the early days of the computer and has, in fact, remained firmly entrenched in the majority of institutions using computers. Non-tutorial uses of computers have developed as a reaction to this and can most easily be approached in the writings of Tim Johns of Birmingham University (6).

The instructional mode uses the computer as teacher, with overt teaching covering presentation, reinforcement and assessment. It is used by students mainly for individual instruction and materials are based on question-answer sequences with possibilities of branching, but ressembling very much conventional stimulus-response learning.

In non-tutorial uses, the computer is seen as partner, the learning tends to be covert and its main use is for reinforcement of skills. It can be used by groups of students with the teacher and material rests on cognitive or discovery learning. Materials include games and conversation simulations in which the student can not only learn from the machine but teach it, pit his wits against it and try to catch it out. Instructional uses are generally machine-controlled, non-instructional uses are often student – or teacher-controlled. The basic routine in instructional programmes is: present question, student inputs answer, check question, provide feedback and determine the next step to be followed. The basic routine in a non-instructional programme is: elicit, student inputs his contribution, the machine responds accordingly.

SOME SIMPLE PROGRAMMING TECHNIQUES

We will now look at some of the simple but powerful techniques, using the BASIC language, on which even the most sophisticated programmes are built and which will enable you to write simple but effective language learning programmes.

1 Telling the computer what to say

A programme consists of a set of statements telling the computer what to do and display. In its simplest form, each statement goes on a separate line. Each line must be numbered so that the computer knows in what order to execute your instructions. The computer will only display something if your programme tells it to and the command you use for this is PRINT. The words that you want printed are put in quotation marks. Example 1 below is a short programme which will get the computer to print HELLO GOODBYE every time it is run.

Example 1:

```
10    PRINT "HELLO"
20    PRINT "GOODBYE"
30    END
```

2 Interacting with the computer

A simple routine for interacting with the computer is as follows: the computer elicits something from you, you type it in and the computer responds to you. The first command you need is one to tell the computer to accept whatever you type in at this point of the programme.

The command that does this is INPUT. You must tell the computer to accept anything that you type in, so you will need some symbol to stand for whatever you type. We use a letter of the alphabet followed by a $ sign – for example A$. Look at Example 2. When you run this programme, the computer will display:

```
      WHAT'S YOUR NAME?
```

You type in your name and the computer will reply:

```
      HELLO followed by your name.
```

Example 2:

```
20    PRINT "WHAT'S YOUR NAME?"
30    INPUT A$
40    PRINT "HELLO"
50    PRINT A$
60    END
```

3 Getting the computer to respond

We often want to tell the computer to vary its response according to what has been typed in (i.e. what is in the INPUT). We do this by using a construction familiar to any language teacher – the conditional sentence. In other words, we tell the computer "If the input is this, then do that." The command is thus: "IF A$ =" "THEN ..." You will see an example of this in Example 3. If the user types in "FINE" the computer will reply "I'M GLAD". If the user types in "NOT VERY WELL" the computer will reply "SORRY TO HEAR THAT". With this simple set of commands you can write quite a lot of conversation simulation programmes.

Example 3:

```
20    PRINT "HOW ARE YOU?"
30    INPUT A$
40    IF A$ = "FINE" THEN PRINT "I'M
      GLAD"
50    IF A$ = "NOT VERY WELL" THEN
      PRINT "SORRY TO HEAR THAT"
60    END
```

4. Telling the computer how and when to do things

You may not want the computer merely to follow a straight line from each line of the programme to the next. You may want the computer to jump forwards or backwards or repeat the programme or part of it. To get the computer to go to another part of the programme we use the command GOTO followed by the line number. In particular, we may want the line the computer goes to to depend on what has been typed in by the user. You may want one input to send the computer back to the beginning of the programme and another input to send it to the end. For example, if the student's answer is correct you want to direct him to the next question; if the student's answer is incorrect you may want him to do the programme again and you return him to a point earlier in the programme. We can do this by combining a GOTO command with an IF .... THEN command as in Example 4. This shows a simple number guessing game. The computer stores a number (LET N = 45) and the user inputs his guesses. If the guess is correct the user is directed to the end of the programme; if the guess is incorrect, the user is sent back to the line in the programme which accepts the guess (line 40) and this guess is then evaluated. So we can get the programme to loop once, until a particular input is given or a specified number of times.

Example 4:

```
5     LET N = 45
10    PRINT "THIS IS A NUMBER GUESSING
      GAME"
20    PRINT "THE NUMBER IS BETWEEN 1
      AND 100"
30    PRINT "YOUR GUESS?"
40    INPUT G
50    IF G = N THEN GOTO 100
60    IF G > N THEN PRINT "TOO HIGH"
70    IF G < N THEN PRINT "TOO LOW"
80    PRINT "GUESS AGAIN"
90    GOTO 40
100   PRINT "YOU'VE GOT IT!"
110   END
```

APPLYING THE TECHNIQUES

The routine in Example 4 is in fact all that is required for a simple instructional programme. The question is presented with a PRINT command and the student's answer accepted with an INPUT command. The computer checks the answer by using IF ... THEN statements and gives feedback using PRINT commands. Finally, the student is directed on to the next question, back to the same question, or sideways to a branch by combining IF ... THEN with a GOTO command. A printout of a simple interaction between student and computer using an instructional programme, is given in Printout 1.

Printout 1:

(Note: On all printouts, S marks the student response.)

```
THIS IS A SIMPLE
MULTIPLE CHOICE TYPE PROGRAMME.
TYPE Y TO CONTINUE
S: ?Y
YOU WILL ANSWER A QUESTION ON A TEXT.
TYPE X FOR TEXT
S: ?X

IF DRIED IN THE SUN, THE BEANS MUST BE
TURNED BY HAND SEVERAL TIMES FOR
EVEN DRYING.

WHAT DOES EVEN MEAN IN THIS CONTEXT?
1. DIVISIBLE BY TWO.
2. FLAT
3. EQUAL
4. CALM

TYPE IN THE NUMBER OF YOUR ANSWER.
S: ?1

NO, DRYING IS NOT SOMETHING WE CAN
COUNT. TRY AGAIN.

WHAT DOES EVEN MEAN IN THIS CONTEXT?
1. DIVISIBLE BY TWO.
2. FLAT
3. EQUAL
4. CALM

TYPE IN THE NUMBER OF YOUR ANSWER.
S: ?3

VERY GOOD.
THE BEANS MUST BE DRIED EQUALLY.
GO TO THE NEXT QUESTION.
```

A conversation simulation programme uses an even simpler routine. A PRINT command tells the computer to elicit something from the student and the response is accepted with an INPUT command. A series of IF ... THEN commands can be used to get the computer to respond according to what response the student typed in. An example of student-machine interaction in a conversation simulation is given in Printout 2.

Printout 2:

```
THIS IS A PROGRAMME TO SHOW HOW I
CAN RESPOND TO WHAT YOU SAY.
PRESS X TO CONTINUE
S: ?X
HOW ARE YOU?
S: ?I'M FINE THANKS
THAT'S GOOD
HOW OLD ARE YOU?
```

```
S: ?21
NO, TELL THE TRUTH. WHAT'S YOUR REAL
AGE?
S: ?40
THAT'S PRETTY ANCIENT ISN'T IT?
HAVE YOU ANY CHILDREN?
S: ?YES
HOW MANY?
S: ?3
3? THAT'S A LOT!
DO YOU LIKE ME?
S: ?NO
YOU'VE HURT MY FEELINGS
I MUST BE GOING. BYE
```

The same set of routines can be used for games or quizzes like the one illustrated in Printout 3.

Printout 3:

```
QUIZ GAME
THINK OF A MEANS OF TRANSPORT AND I
WILL TRY AND GUESS WHAT IT IS
READY?
S: ?YES
DOES IT TRAVEL ON LAND (L) IN THE AIR
(A) OR ON WATER (W)?
S: ?L
SO IT'S A MEANS OF LAND TRANSPORT
HAS IT GOT AN ENGINE?
S: ?YES
HAS IT GOT WHEELS?
S: ?YES
SO IT'S GOT AN ENGINE AND WHEELS. DOES
IT RUN ON ROADS OR RAILS?
S: ?ROADS
HOW MANY WHEELS HAS IT GOT?
S: ?FOUR
IS IT USED FOR CARRYING PEOPLE OR
GOODS?
S: ?PEOPLE
CAN IT BE USED FOR PUBLIC TRANSPORT?
S: ?YES
IT'S A BUS
SO THAT'S IT. DO YOU THINK I DID WELL?
S: ?YES
THANK YOU VERY MUCH. DO YOU WANT
ANOTHER GO?
S: ?NO
```

Now, there is no denying that even simple programmes like these take a long time to construct and type out — a beginner would need an hour or two to write the programme in Printout 3. So it is worth thinking how all the effort put into constructing programmes can be made to go as far as possible. An instructional programme is relatively inflexible and, with the exception of optional remedial loops produces the same amount and kind of practice each time. A non-instructional programme, however, can be designed to be open-eneded so that as students inputs will vary the programme may turn out quite different each time it is run. In addition, they can be designed for use in groups and to generate language use rather than control it.

As an example of this rather less conventional type of language learning programme, I will mention one not intended for language practice but which can serve this purpose very well. The game is called Space Mines and the group of students playing it are rulers of an imaginary planet. They have to make a number of decisions on what to buy and sell. The computer tells them how the price of commodities varies and informs them of the outcome of their decisions. Depending on their luck and skill the players may survive their term of office to lead the country to prosperity or their rule may finish prematurely in bankruptcy or a *coup d'etat*. The important thing is that a series of decisions have to be taken and students are forced to predict the consequences of possible moves and analyze the consequences which the computer tells them after each move. This, I would suggest, has all the ingredients of natural communication and, with language preparation and follow-up from the teacher, can easily become the basis of a language learning unit. So, rather than expecting the computer to take over the functions of the teacher, the two are working together, each doing what they are best equipped to contribute to the learning. In other words, the teacher not only gains a valuable aid: she keeps her job.

(1) T.F. Johns EXPLORATORY CAL: AN ALTERNATIVE USE OF THE COMPUTER IN TEACHING FOREIGN LANGUAGES (mimeo)

(2) An example is the game ANIMALS included in the DOS 3.3 System master diskette.

(3) See, for example, the game SPACE MINES described at the end of this article.

(4) One of the difficulties in computer assisted learning, in fact, is that the computer does this too well – it will only accept as a correct response one which is identical in all respects to what it has been told to accept. While this is no problem in multiple-choice-type items it makes more open-ended learning items difficult to handle.

(5) For example, the ability to select random numbers, the recognition of numbers which are equal to, greater than or less than another number.

(6) See 'The uses of an analytic generator: the computer as teacher of English for Specific Purposes.' ELT Documents No. 112, 1981 pp. 96-105. Also, G. Davies 'Computers, language and language learning' CILT.

JUST SITTING AROUND?

GET INTO ACTION.

Join

MELTA