Simplicity is best learning pattern

Dave Sayers relates how the ZX-81 has helped learning in his school.

THE POWER of a micro as a learning tool is often quoted in papers and magazines, yet most of the published programs are games and adventures — and good they are, too. Nonetheless, it is a pity that more educational games are not published; the little ZX-81 with even a basic 1K memory can be used to teach many of the fundamentals of mathematics or, with more memory, English.

The example programs included in this article, therefore, are all written within 1K. The reason is simple. To be good, a teaching game need not be complicated. Preferably it should teach only one thing, though that is not to say that mixtures of approaches are not desirable.

It should also, if possible, have a graphics display to help keep the attention of the child. Another good reason for keeping games simple is that many ZX owners, or owners of other machines, are young people like myself who have children at school. With the ZX available, it would be a waste if it were not used constructively to further a child's education.

Initially most people probably write programs of the $6 \times 5 = ?$ variety. They are useful but not especially interesting. Try the program in figure one.

It is very simple; the program prints out 10 rows of graphics squares, with the number of squares printed at the end. It can be used to help to teach children who can count who have difficulty in associating those numbers with the written number — for instance, seven.

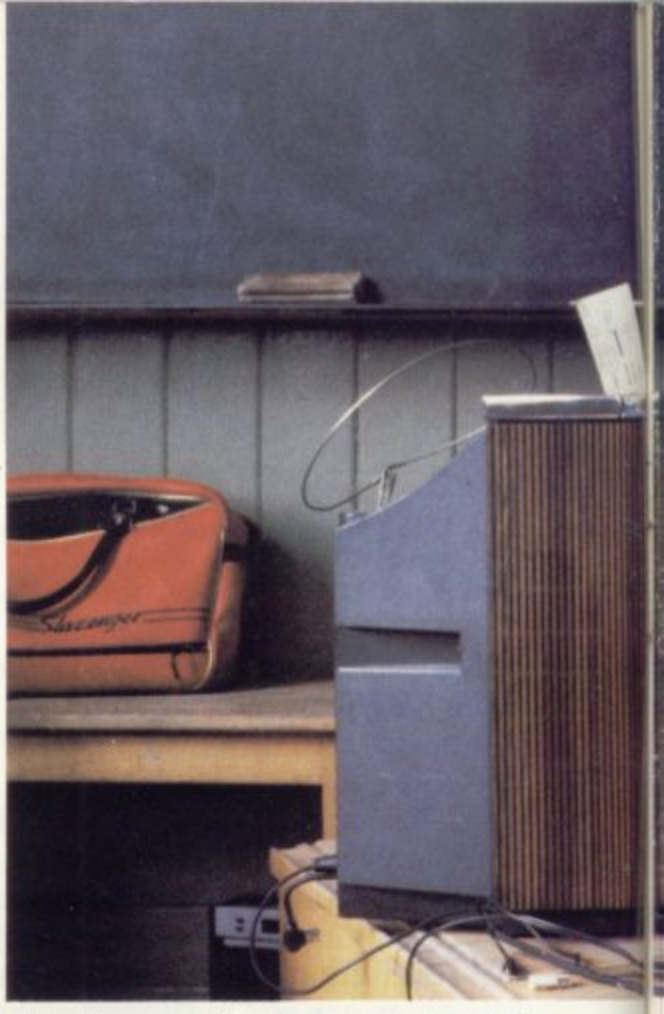
As you can see, I am starting at the very first basis of counting. My daughter has started learning to write, or record, numbers greater than 10, and this kind of game can be a great help. A program I have used flashes rows of squares for her to input the number — figure two. It was written specifically to help her understand the way in which numbers above 10 are recorded. The same program, of course, can be adapted easily to give help with numbers above 20.

Although the programs may appear limited, they are still giving valuable practice in learning skills which, if not understood fully, can lead later to difficulty in comprehending, for instance, the value attached to a carried 1 — in 9 + 9 we "carry one" which is really 10. You may say that is well known but rest assured there are plenty of children who find the true value of a carried figure a total mystery.

> 'The little ZX-81 with even a basic 1K memory can be used to teach the fundamentals of mathematics mathematics'

Computer-aided learning can give those children valuable lessons, for they will know immediately if they are correct or incorrect. In classes where books are marked once a lesson, the incorrectness of what they have done may become apparent only at the end of the lesson — or next day.

Once we have the child understanding counting to a certain total, speed practice becomes important in helping the child to rely less on physical cues of number and begins to depend on their internal memory of where they are in the count. The games should show varying numbers of shapes quickly on the screen, for which a running total has to be kept by the child, to be entered at the end of the run.



HISSING SID is an example of such a program. Snakes appear on the screen, varying numbers each time. A total is input at the end, which is checked — figure three.

There is plenty of fun while learning to count with this program, especially if it is altered to print the total at the end, after a short pause, to give children a chance to shout the answers. They love to see who can be first with the correct answer. Figure four shows how to do this.

This program can be altered in both speed of display and numbers of snakes. For larger numbers alter line 20 and for less time between displays alter line 100. If you make it long enough and quick enough it becomes fairly testing, even for adults.

I wrote a similar program on the Pet at school and tried it on my own class, as well as the class of a colleague.

My class, by now fairly blase where micros are concerned — some of them prefer me to use the ZX their fathers or brothers own one and they can crib my programs — enjoyed it a great deal. The colleague, who was interested in

what the computer could do, was surprised by the way in which some of the less-motivated children were captivated by the game. She was particularly impressed with the way in which one boy, who was finding addition a trial, began to store the numbers in his head; then, staring into space for a minute or so, as he totalled them, he finally delivered his answer in a very positive way.

Gone were the uncertainties he had about written sums, when he would often be at the teacher's desk complaining that he could not understand. Replacing it was the desire to beat the computer, to have his answer before it flashed on the screen.

BRIDGES is a game in which the children attempt to build a bridge by answering addition questions correctly. Each correct answer puts another span on the bridge. As before, they are adding blocks, rather than numbers — figure five.

When the game is run, blocks, in two rows, appear on the screen. If they are added correctly, a span is added to the bridge. If they are not added correctly, there is no penalty; the problem resumes again and again until it is solved correctly.

That is important, as there is no worry that the child will feel he or she has failed on the first few attempts if the bridge is not nearing completion.

Those with 16 or more K of memory might like to improve this program. For instance, the bridge may fill the whole screen and every time there is an incorrect answer a man might walk along the bridge and fall off, to land in a boat, from where he is returned to the bridge to climb and try again. Whether you use these ideas or not, it is important to pay careful attention to what you want your program to teach and never to take for granted the idea that children understand the logic of mathematics.

Figure 1.

```
 10 FOR I=1 TO 10
 20 FOR J=1 TO I
 30 PRINT "▒ ";
 40 NEXT J
 50 PRINT I
 60 PRINT
 70 NEXT I
```

Figure 2.

```
 10 LET A=(INT (RND*10) +10)
 20 CLS
 30 FOR I=1 TO A
 40 PRINT "■ ";
 50 NEXT I
 55 PRINT
 60 INPUT B
 70 IF A=B THEN GOTO 10
 80 PRINT "WRONG , IT WAS ";A
 90 FOR I=1 TO 100
100 NEXT I
110 CLS
120 GOTO 10
```

Figure 3.

```
  5 CLS
 10 LET C=0
 20 LET A=(INT (RND*5) +1)
 30 FOR I=1 TO A
 40 LET B=(INT (RND*6) +1)
 50 LET C=C+B
 60 FOR J=1 TO B
 70 PRINT "(3 SPACES)-▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒"
 80 PRINT
 90 NEXT J
100 FOR K=1 TO 70
110 NEXT K
115 CLS
120 NEXT I
130 PRINT "HOW MANY SNAKES?"
140 INPUT D
150 IF D=C THEN GOTO 5
160 PRINT "THERE WERE ";C;" SNA
KES"
170 FOR I=1 TO 100
180 NEXT I
190 GOTO 5
```

Figure 4.

Change these lines
130 for u = 1 to 150
140 next u

Do not forget to delete line 150.

Figure 5.

```
  1 LET S=0
  5 PRINT
  6 PRINT
  7 PRINT
 10 FOR I=1 TO 10
 20 PRINT "▒▒(10 SPACES)▒▒"
 30 NEXT I
 40 LET A=(INT (RND*9) +1)
 50 LET B=(INT (RND*9) +1)
 60 FOR I=1 TO A
 70 PRINT AT 15,I;"█"
 80 NEXT I
 90 FOR I=1 TO B
100 PRINT AT 17,I;"█"
110 NEXT I
120 LET S=S+1
130 INPUT D
135 IF D<>A+B THEN LET S=S-1
140 IF D=A+B THEN PRINT AT 3,S+
1;"■■"
150 IF S=10 THEN GOTO 300
160 PRINT AT 15,1;"(12 SPACES)"
170 PRINT AT 17,1;"(12 SPACES)"
180 IF S=10 AND D=A+B THEN GOTO
400
190 IF D=A+B THEN GOTO 40
200 GOTO 60
400 PRINT "YOU BUILT A BRIDGE"
```