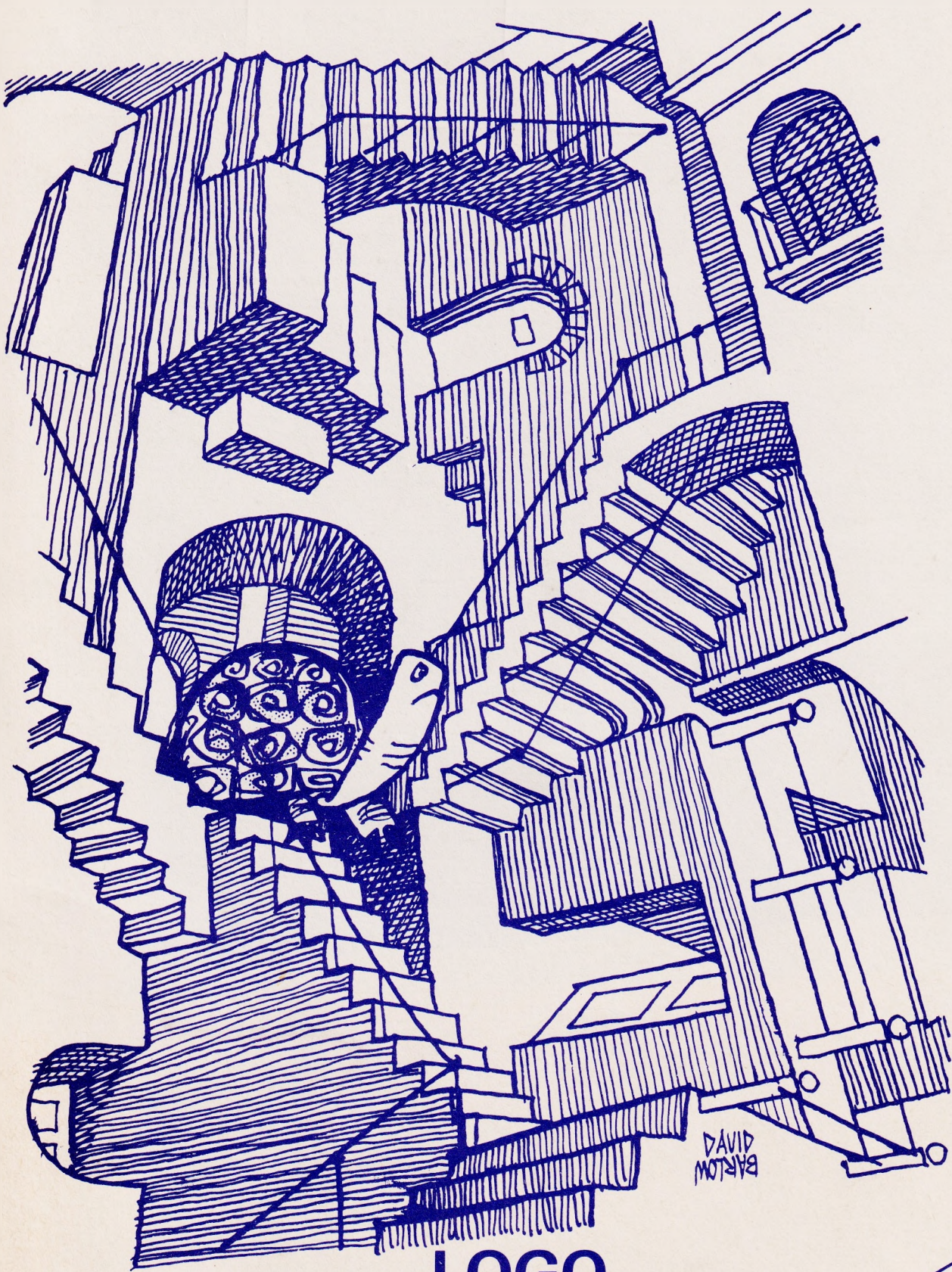


# M I C R O -

# S C O P E



## LOGO

Special

Newman College



MAPE



BLUG

---

# Contents

Editorial	1
The spirit of LOGO discerned <i>Allan Martin</i>	2
Why BLUG? <i>Derek Radburn</i>	7
LOGO – where next? <i>Julian Pixton</i>	8
Simple LOGO in primary schools: a structured or unstructured approach? <i>Helen Finlayson</i>	10
Approaching LOGO <i>Richard Noss</i>	13
St Maurice School LOGO Project: a rationale <i>Tony Mullan</i>	15
But what's the use of asking kids to have the computer draw a house? <i>Catherine Berdonneau</i>	20
What do children think about turtles? <i>Beryl Maxwell</i>	23
Turtle graphics in the secondary school <i>Kim O'Driscoll</i>	25
Languages for young programmers <i>Tim O'Shea</i>	26
LOGO implementations	30
Books	32

*Editorial team* Allan Martin, Derek Radburn (BLUG)  
Roger Keeling, John Lane (*MICRO-SCOPE*)

© Newman College/MAPE and BLUG 1983

ISBN 0 602 22692 9 ISSN 0264-3847

Sponsored by the Department of Industry

Correspondence to the Editor: Newman College, Bartley Green, Birmingham, B32 3NT  
tel: 021 476 1181

**MAPE (Micros and Primary Education)** is open to individuals and institutions.

The current subscription of £7.50 p.a. includes direct mailing of **MICRO-SCOPE**.

Application forms from: Barry Holmes, St Helen's C.P. School, Bluntisham, Cambs.

**BLUG** subscription is £7.50 p.a. Secretary: Pam Valley, Shell Centre for Mathematics,  
School of Education, University of Nottingham.

Published by Heinemann Computers in Education Ltd  
in partnership with Ginn and Company Ltd.

Typeset by Castlefield Press, Northampton.  
Printed by Biddles of Guildford.

---

# MICRO-SCOPE LOGO SPECIAL

## Editorial

At last we have got it together! Here is the long-planned, long-promised *MICRO-SCOPE Special* on LOGO as a programming language for use in schools.

This issue has been shaped through collaboration between the *MICRO-SCOPE/MAPE* team and the newly-formed British LOGO User Group (BLUG), who provided the majority of the articles very quickly. Other promised and important articles were less prompt, which in turn led to delays in rescheduling the publication process. We apologise to all readers for stretching their patience, and particularly to BLUG for missing the target date of their first annual Conference, held successfully in September with Seymour Papert in attendance.

The material that follows needs very little editorial comment. Allan Martin provides an introduction to the LOGO atmosphere, an atmosphere experienced as a very real thing by those of us who attended the BLUG Conference. Derek Radburn, BLUG's Chairman and himself a primary school Headteacher, describes BLUG's origins and aims. Julian Pixton shows how paths can be developed along two of the many avenues possible in a LOGO environment. Tony Mullan, Helen Finlayson and Richard Noss explore some of the issues involved in thinking about using LOGO in school. The articles by Catherine Berdonneau and Beryl Maxwell concentrate very much on

the LOGO users, children in French and British primary schools. Kim O'Driscoll's note serves to point out that LOGO's possibilities extend into the secondary school, that LOGO isn't just for the very young. Finally, Tim O'Shea compares some of the programming languages available for young programmers. We have appended two more items, a list of currently available LOGO implementations, and details of some books which will provide readable introductions to LOGO.

The BLUG Conference has undoubtedly been the most recent major LOGO event in this country. An international attendance of 200 people, keen to learn and open to new ideas, made it an exciting experience for all who attended. Particularly notable was the mixture of interested people of all kinds: teachers, advisers, civil servants, journalists, researchers, business computer consultants, publishers, psychologists, and more. The main speakers – Deborah Booth (a primary school teacher), Peter Ross (Edinburgh University) and Seymour Papert – represented this diversity, and expressed the feeling of those who mix with LOGO that they are dealing with a thing of power.

We hope this collection will enable you to sample something of the LOGO potential.

November 1983

# The spirit of LOGO discerned

Allan Martin  
*St Andrew's College of Education,  
 Bearsden, Glasgow*

## PART I WHAT IS LOGO?

### Answer 1: LOGO is a programming language

LOGO is one of a number of computer languages to have been developed in the field of Artificial Intelligence. Researchers in Artificial Intelligence (AI for short) attempt to understand human thinking processes and behaviour patterns (such as language or vision) by trying to develop computer-based simulations of these.

In pursuing this end they have found it convenient to develop computer languages suitable to these particular activities. One of these languages is LISP (short for LISt Processing language). LISP is highly logical and has a powerful facility for handling and manipulating lists of items as complete units; however, LISP programs have a tendency to be difficult to follow when they reach any degree of complexity.

LOGO was developed out of LISP in 1968 as part of a research project to create a language for the teaching of mathematical ideas through programming. Its creators, principally Feurzeig and Papert, intended that it should be easy to learn, easy to use, and easy to read. Throughout the 1970s, research was carried out on LOGO, chiefly at Massachusetts Institute of Technology (under the direction of Seymour Papert) and at the Artificial Intelligence Department at Edinburgh University (under the direction of Jim Howe). Some of this research was concerned with the use of LOGO in teaching mathematics, but its value in many other learning areas (including music and juggling) has also been demonstrated.

As a programming language, LOGO can handle three types of data objects. These are numbers, words, and lists. Any of these three types can be accepted by LOGO as single items. Thus the following variable declarations are all acceptable:

```
MAKE "X 206
MAKE "Y "GREETINGS
MAKE "Z [RATS GLUE PLASTIC
BANANAS]
```

Here X is assigned a number as its current value, Y a word, and Z a list. Data-handling operations can be carried out on all three types of data object:

```
PRINT :X + 44
will produce 250
```

```
PRINT BUTLAST :Y
will produce GREETINGS
```

```
PRINT LAST :Z
will produce BANANAS
```

The function of the operations BUTLAST and LAST can be deduced from these examples. The ability to manipulate lists easily is one of LOGO's most powerful features, and makes LOGO accessible to many areas beyond the mathematical.

LOGO programs are built up through the use of *procedures*. A number of LOGO commands can be combined to form a simple procedure. Simple procedures can be used as commands in more complex procedures, and so on.

```
TO SPACE
  CLEARTEXT
  REPEAT 10 [PRINT []]
END
```

This procedure, SPACE, which clears the screen (using the command CLEARTEXT) and prints ten empty lines, can itself be used as a command in the procedure TITLE.

```
TO TITLE
  SPACE
  PRINT [WELCOME TO THE WORLD
  OF LOGO]
  WAIT 20
END
```

WAIT may be defined as a delaying procedure requiring an input (in this case 20). The procedure TITLE may now be used as a command itself. A LOGO program therefore consists of a hierarchy of procedures, each one fairly short and thus hopefully easy to understand and to alter if necessary. This hierarchical structure contrasts with the serial nature of programs written in a language like BASIC.

### Answer 2: LOGO is a theory about thinking and programming.

The LOGO programming process can be seen as a direct analogy of the thinking process. Seymour Papert spent several years in Europe working with Jean Piaget, and his views on LOGO as a vehicle for thought show the influence of a tradition in European thinking

which seeks to interpret human behaviour and human cultural products as evidence of 'deep structures' which govern the way in which people think and interact. Piaget's work lies within this tradition: he sees the development of the child's thinking abilities as a process whereby simple structures are generated and tested, and later (at an appropriate stage of development) combined with other simple structures to form a more complex or more abstract structure. This latter forms the basis of further intellectual growth. The building up of intellectual capacity is then a continuous process of generating more complex and abstract thought-structures out of simpler ones. In this way a hierarchy of thought-structures is built up.

The similarity between the building up of LOGO programs and the building up of thought-structures is clear. When we develop a LOGO program we go through a process which parallels that of thinking. Individual procedures are written and tested, then put together with others to form super-procedures, and so on. For Papert, this similarity between LOGO-work and thought-work is what makes LOGO a 'tool to think with' in a way which other programming languages cannot be.

### **Answer 3: LOGO is a philosophy of education.**

The parallelism between LOGO activity and thinking lies at the base of the 'LOGO approach' to education. In developing his powers of thinking, the child builds up structures of thought by exploration of the world around him. Thus, faced with the challenge of a new problem or situation, concepts previously mastered are combined to produce a new insight, one which can then be transferred to the child's thinking about similar situations. The proponents of this view of intellectual development place emphasis on exploration and discovery as important elements in learning, and these are generally accepted as essential parts of the experience which the primary school can offer. Exploration and discovery are also central elements of the LOGO learning experience. The LOGO user may explore situations and think out problems by building up LOGO programs out of procedures constructed by himself. Different sets of LOGO facilities place the user in different 'micro-worlds' which can become environments for exploration and therefore for learning. The possibilities offered by 'turtle geometry' give only one of the sets of micro-worlds which current implementations of LOGO can make available.

There is an ongoing debate in the world of primary education about the teacher's role in the process of discovery learning, the central

question being 'how guided should discovery be?' This debate is mirrored in the LOGO community. All would accept that LOGO learning is a process of discovery through the achievement of insight into the nature of a problem or situation. Differences only emerge over the degree to which the process should be guided or channelled. Papert takes what is perhaps a consciously utopian line, arguing that placing children in a 'computer-rich environment' and allowing them to explore is sufficient for acceptable learning to occur. Ultimately, he suggests, schools and teachers are unnecessary, for computer-rich learning environments can be made available without such institutional trappings. However, in a world where schools still exist, many researchers and teachers who have used or are using LOGO in the classroom would raise two points:

- a) Gaining knowledge of a learning area by complete discovery is a very time-consuming and inefficient process. Supplying tools for discovery and even hints for the direction of exploration can speed up the process while retaining the insight-gaining experience.
- b) In schools, where many activities compete for involvement in the learning programme, LOGO must be justifiable in currently intelligible curricular terms if it is to be given time during the school day.

These qualifications have resulted in variations in approach between different research and development groups, and even between different projects carried out by the same group. Some LOGO activities thus look more highly organised than others, or are slanted in different curricular directions. It is important to note however that all major LOGO developers agree in viewing LOGO learning as in essence a discovery process. Clearly, LOGO-use has important implications for what is taught and the way in which it is taught.

## **PART II WHAT FACILITIES CAN BE USED THROUGH LOGO?**

Teaching mathematical ideas is at present possibly the most common LOGO use in schools. However, LOGO is essentially a programming structure upon which various 'facilities', or tool-kits of relevant procedures, can be hung. The facility for turtle graphics is one of a number of actual or potential toolkits; thus teaching mathematics through turtle graphics must be seen as merely one of many possible applications of LOGO. LOGO is not a 'maths-oriented' language. Some of the facilities accessible through LOGO are indicated in the following paragraphs.

## 1. Turtle graphics

Turtle graphics is undoubtedly the most familiar facility which LOGO offers. As a visually rewarding and easily usable system, it is often seen as a convenient introduction to LOGO programming, either as a thinking exercise or as a vehicle for understanding particular mathematical concepts. In practice most activity with LOGO in education so far has concentrated on the use of turtle graphics. Some confusion in this area has arisen, due to the fact that turtle graphics packages are also available in non-LOGO contexts, for instance, in Apple Pascal or Atari PILOT. There are also available, particularly for the BBC microcomputer and the RML 380Z, turtle graphics programs (usually written in BASIC) offering a limited range of LOGO-like command structures. These programs should not be confused with full implementations of LOGO. Where, as with the BBC micro, no implementation of LOGO is available (or likely to be for some time), inexpensive turtle graphics programs may form a useful stop-gap sampler of LOGO-like activities. However, where a LOGO implementation is available (as in the case of the 380Z), the power and the range of

facilities which LOGO offers make turtle graphics packages seem very much a false economy.

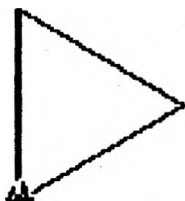
These procedures indicate the way in which the complex may be built up easily from the simple.

```
TO TRIANGLE
  REPEAT 3 [FORWARD 100 RIGHT 120]
  END
```

```
TO SPINTRI
  REPEAT 12 [RIGHT 30 TRIANGLE]
  END
```

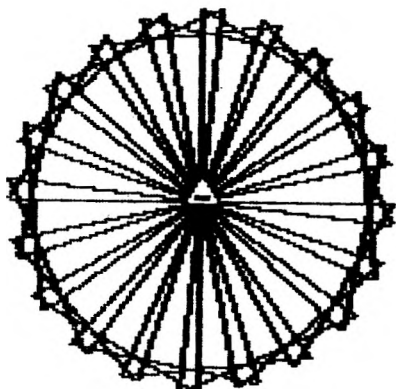
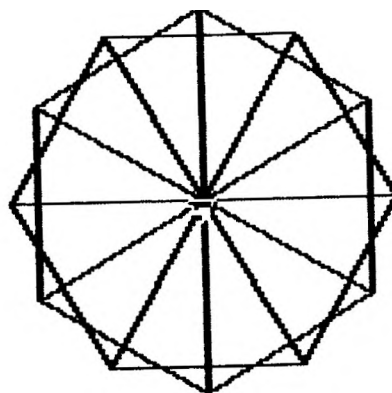
```
TO SPINTRIN :NUM
  REPEAT :NUM [RIGHT 360/:NUM
    TRIANGLE]
  END
```

The procedure TRIANGLE draws a triangle of three equal sides. SPINTRI combines twelve such triangles into a wheel-shaped design. SPINTRIN enables you to combine any number of triangles (shown as the variable NUM) into a wheel-shape. Thus SPINTRIN 15 forms a wheel of fifteen triangles.



```
TO TRIANGLE
  REPEAT 3 [FD 60 RT 120]
  END
```

```
TO SPINTRI
  REPEAT 12 [RT 30 TRIANGLE]
  END
```



SPINTRIN 20

```
TO SPINTRIN :NUM
  REPEAT :NUM [RT 360/:NUM TRIANGLE]
  END
```

## 2. List processing

LOGO offers powerful facilities for dealing with words. Here are some examples:

```
TO MONKEYS
  REPEAT 6 [PRINT[MONKEYS LIKE
    BANANAS]]
END
```

The outcome of the procedure MONKEYS is fairly clear.

```
TO ECHO
  PRINT[TYPE A WORD]
  MAKE "ITEM REQUEST
  PRINT (SENTENCE : ITEM : ITEM
    : ITEM)
END
```

The procedure ECHO asks for a word to be typed in, then prints it out three times. Typing in SHOUT will result in the printed output SHOUT SHOUT SHOUT.

```
TO EATS :ANIMALS :FOOD
  PRINT (SENTENCE [WHAT DO]
    :ANIMALS [LIKE TO EAT ?])
  IF REQUEST = :FOOD THEN PRINT
    [RIGHT] STOP
  PRINT[HERES NEWS]
  PRINT (SENTENCE :ANIMALS [EAT]
    : FOOD)
  PRINT[NOW TRY THIS]
  EATS :ANIMALS :FOOD
END
```

The procedure EATS must be supplied with two parameters in square brackets (i.e. as lists). EATS [MONKEYS] [BANANAS] will produce continued asking of the question 'What do monkeys like to eat?', with suitable prompts until the answer 'bananas' is provided. EATS can be applied to any combination of animals and their favourite food, e.g.

```
EATS [BLACKBIRDS] [WORMS]
EATS [BIRD-EATING SPIDERS] [BIRDS]
EATS [CERTAIN FRENCH PERSONS]
[FROGS LEGS]
```

Notice that because the parameters to EATS are lists, descriptions of more than one word are possible.

These examples give only a hint of LOGO's processing power, a power which makes LOGO not only suitable for children to play with words, but also for teachers and learners of any age to write complex interactive programs.

## 3. Numerical operations

The definition of simple operations followed by their combination into complex ones likewise characterises LOGO's number-handling abilities. Normal numerical operators can be used with LOGO, e.g.

```
PRINT 12 * 9
PRINT 264 + 1032
```

Supplying inputs to procedures enables custom-built operations to be performed.

```
TO SQ :NUM
  OUTPUT :NUM * :NUM
END
```

The procedure SQ will generate the square of a number supplied. Thus the command PRINT SQ 9 will print 81,

```
TO SQADD :Y :Z
  PRINT (SQ :Y) + (SQ :Z)
END
```

SQADD will square two numbers supplied, add the squares together and print out the result on the screen. Thus SQADD 2 3 will produce 13.

```
TO EXP :NUM :POWER
  IF :POWER = 0 THEN OUTPUT 1
  OUTPUT :NUM * (EXP :NUM :POWER-1)
END
```

Recursion, that is, the ability of a procedure to call itself (seen in EATS), enables LOGO to offer complex calculations in concise form. The procedure EXP raises the number supplied to the power supplied. Thus PRINT EXP 4 2 will produce 16 whilst PRINT EXP 4 3 will produce 64.

## 4. Music

Another toolkit of procedures enables most LOGOs to offer a music facility. The Terrapin version for the Apple II uses the command PLAY followed by two lists, one of pitches, one of durations.

```
TO C. CHORD
  PLAY [8 12 15 20] [40 40 40 40]
END
TO F. CHORD
  PLAY [13 17 20 13+] [40 40 40 40]
END
```

The procedure C. CHORD plays the notes C, E, G, and C in succession, while F. CHORD plays F, A, C, and F. The numbers representing the pitches can be read off a diagram of the staff. C. CHORD and F. CHORD can be combined together to make FC. TUNE.

```

TO FC. TUNE
  C. CHORD
  C. CHORD
  F. CHORD
  C. CHORD
END

```

Since a wide range of pitches and durations may be chosen, the possibilities of creating individual bars and combining them in different ways are considerable. The potentiality for exploring rhythmic, melodic and harmonic patterns in music by the exercise of building it from simple components may be imagined. The possibilities will be even greater if LOGO implementations appear which will support more than one channel of sound simultaneously.

### 5. Control

As yet still under development is the application of LOGO to the control of equipment external to the computer. In the early years of its life, LOGO was used to control the Floor Turtle, a dome-shaped robot capable of following LOGO commands to draw lines. However, in the United States and in this country, the construction of LOGO-type languages for control applications is being actively pursued. One early candidate for such application is the BBC Buggy, however, in the next few years we can expect to see LOGO in measurement, in switching operations, and in the control of robot devices.

### PART III WHERE DOES LOGO GO FROM HERE?

Having looked at some of the facilities which the LOGO language offers, it remains to ask where current LOGO developments may be heading. Here are some possibilities.

#### 1. Widening range of applications

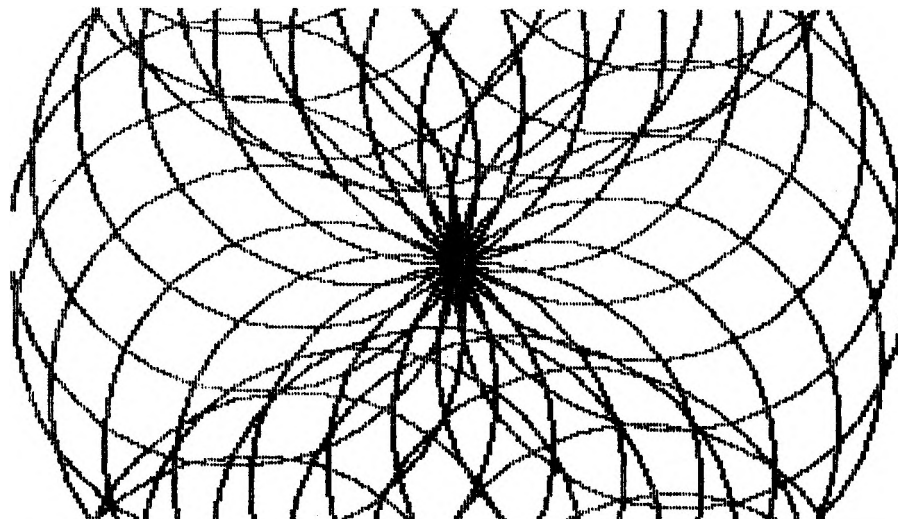
An existing area of application which is still moving on is graphics. Recent implementations of LOGO support sprites (moveable graphics objects which can move in front of or behind each other), multiple turtles, and sprites whose collisions can be sensed and acted upon. Music is another area where new implementations may offer more facilities, such as multi-channel sound. As mentioned above, more control applications may be expected also. We must in addition be prepared for applications as yet not considered.

#### 2. New implementations

Full LOGO implementations are now available on the Apple II (two versions), Texas Instruments 99/4a, IBM Personal Computer, RML 380Z and 480Z, Commodore 64, and Atari 800. Of these, the Atari is possibly the most advanced. However, LOGO has been promised for the Sinclair Spectrum and the BBC microcomputer, and one looks forward to see what facilities these will offer.

#### 3. Increasing use in education

LOGO makes an ideal language for developing computer familiarisation and computer use, as well as general skills of structured thinking and problem-solving. As such, it should have a place in both primary and secondary schools. In the primary school, LOGO can form the ideal environment for many computer-based activities. In the secondary school, it may ultimately supplant BASIC as the language of Computer Studies. Further testing and evaluation lies ahead before such things will happen. Clearly, however, there is a future for LOGO.





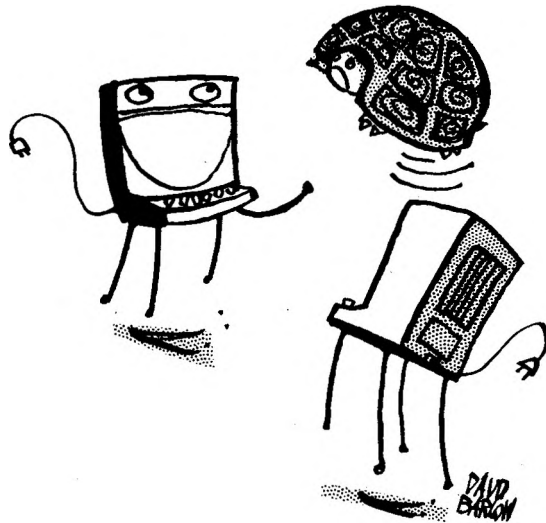
# Why BLUG?

**Derek Radburn**  
*Chairman, British LOGO User Group*

It's an unprepossessing acronym, isn't it? But it's just sufficiently discordant to be memorable (and vulnerable to snide remarks about **BLUGGERS!**). The British LOGO User Group, to give it its full name, was formed in the latter part of 1982. The coming together was a result of people of varying degrees of experience and different motives of interest in LOGO having shared a week at the Edinburgh LOGO Summer School of 1982.

Underlying the Group's formation was the conviction that LOGO had something special to offer to learning (and not just children's learning). LOGO directs the learner's attention not just to results but to the means by which those results are achieved. LOGO offers the learner the opportunity for the practice and development of the intellectual skills of analytical and sequential thinking – a problem-solving approach. At the child level, LOGO, through its resemblance to natural language and its extensibility, makes computing power accessible to children as programmers. Its stimulus to the imaginative and creative use of the computer has a potency which goes beyond intellectual gain to affect the learner's self-image in a way the closed 'drill and practice' software can never do. (This well-nigh unquantifiable aspect of attitude, particularly self-attitude, is one which is too often overlooked.) I must apologise for this seeming digression, but it is all germane to the causes which led to BLUG coming into being.

BLUG has, as its prime aim, 'the promotion of LOGO as a thinking tool'. To achieve this, the Group's chief activity is communication. We want to make people aware that there is more to LOGO than just turtle graphics – that nearly 80% of the implementations of full LOGOs are dedicated to the list-processing powers of the language. As a Group, we attach particular importance to the philosophy which underlies LOGO. Misused, LOGO is just as vulnerable to becoming a new item of pedagogic orthodoxy as any other development, but not if its use is true to the philosophy, which we see as an integral part of the LOGO package. We aim to communicate ideas and examples of use involving LOGO. This involves putting our membership in touch with each other and with anyone else using LOGO. We shall make software available to our members and give



*Terms explained:* HEADING

information about machines, implementations, and literature pertinent to LOGO. A subsidiary role, but an important one, which we hope the Group will develop will be to influence the standards of implementations appearing in machines; there is some evidence that this is happening already.

The way in which we are setting about doing this is by the quarterly publication of a Newsletter, entitled *Logos*. This will be sent to BLUG members, together with a complete membership list.

The first annual BLUG conference was held at Loughborough University in September 1983. BLUG's President, Seymour Papert, was both a speaker and a participant.

We shall be publishing, later, the first annual LOGO Almanack, which will have a content which will be of a more academic nature than the Newsletter, and we hope will soon establish itself as an authoritative reference on LOGO. We also have hopes of making the first items of software available to members soon. Our membership now approaches 400, despite the fact that we have not done much active seeking of members so far. We even passed by the opportunity for publicity at the time the *Horizon* programme, 'Turtle-Talk', was screened. We preferred not to be inundated and appear inefficient, but to grow slowly and effectively. We feel we have now established an effective organisation and are ready to go out and seek members. At present BLUG has no sponsors.

So, in conclusion, if you are interested in LOGO and would like to be involved in BLUG and its aspirations, we shall be happy to hear from you. Write to Pam Valley (BLUG Secretary), Shell Centre for Mathematics, School of Education, University of Nottingham.

# LOGO – where next?

**Julian Pixton**  
*Hillary Junior School,*  
*Walsall*

LOGO is the global name for a growing family of computer language implementations. Characteristic features of these languages include the ability to define procedures as new words, which can then be used exactly as primitive commands and functions. These defined procedures may also include variables which are local to a given procedure, which permits true recursion.

LOGO also embodies a full list structure. That means the language can manipulate lists which themselves can be made up of the names of other lists, which themselves may be lists of lists and so on.

Newer implementations of LOGO are shortly to become available with elements of parallel processing and sophisticated message passing in order to facilitate speedier graphics programming.

## 1. Multiple Turtles

Conventionally, in the LOGO graphics subset known as turtle graphics, complex figures are constructed by drawing one piece of the object at a time, until the figure is complete.

Newer implementations will soon offer the possibility of drawing all the component parts of a complex shape at once, using several turtles.

Multiple turtles provide many extra possibilities. It is often possible to simplify programming by giving individual turtles different sub-tasks in a program. We can make drawings by assigning turtles to the tasks of drawing individual pieces of the whole. In this way, the drawing seems to evolve instead of just appearing, piecemeal. At a more serious level, multiple turtles can be used to illustrate concretely the process of multiprogramming.

New turtles are created in MIT LOGO by means of the HATCH command.

The form of the command is:

```
HATCH turtle number Procedure for the
  turtle
```

For example,

```
HATCH 1 SQUARE 20
```

The turtle number is the name or label of the turtle. It can be any number between 1 and 254. Turtle 0 is the master turtle.

SQUARE is the name of the procedure which we are telling turtle 1 to run. The number following SQUARE is the value to be passed to the local variable within SQUARE.

Having defined SQUARE as,

```
TO SQUARE : LENGTH :XPOS :YPOS
  SX : XPOS          SY :YPOS
  REPEAT 4 (FORWARD : LENGTH
    RIGHT 90)
  END
```

we can try out this simple multiple turtle program.

```
TO TRYOUT
  HATCH 1 SQUARE 50 30 60
  HATCH 2 SQUARE 40 180 90
  HATCH 3 SQUARE 60 100 90
  SQUARE 20 150 120
  END
```

Notice that each turtle has its own procedure and its own set of values for the variables. The last call of SQUARE has no HATCH preceding it, so it is addressed to turtle 0. Turtle speed obviously slows down as the number of turtles on the screen increases.

One interesting exercise is to try and adapt an existing program which utilises recursion into a routine using multiple turtles. The famous tree program is ideal.

```
TO TREE :SIZE
  IF :SIZE <2 (STOP)
  FORWARD : SIZE
  RIGHT 15
  TREE ( 3 * :SIZE/4)
  LEFT 30
  TREE ( 3 * :SIZE/4)
  RIGHT 15
  BACK : SIZE
  END
```

Here is the same program written using multiple turtles:

```
TO TREE : SIZE
  IF ME = 0 (CLEAR SY0)
  IF :SIZE > 6
    ( FORWARD :SIZE LEFT 30
  HATCH 1 TREE ( 3 * :SIZE/4)
  RT 60
  HATCH 2 TREE ( 3 * : SIZE/4)
  VANISH )
  ELSE
  END
```

The elegant and powerful mathematical ideas embodied in the TREE program are worth investigating in detail. Recursion is a little complex, but it is so powerful that it is well worth the effort to understand.

Comparing the speed of the multiple turtle version with the earlier version is interesting. It will give you some idea why the concept of multiprogramming is worth learning about.

Trees are so easy to draw with multiple turtles that you could quickly draw a complete forest. Using the previous TREE procedure we can add:

```
TO FOREST
  BACKGROUND 1
  SX 236
  REPEAT 3 ( SY 10
    SX XLOC ME + 40
    HATCH 1 TREE 20
    SX XLOC ME + 40
    HATCH 2 TREE 30
    MANYCLOUDS)
  END
```

```
TO CLOUD :SIZE :X
  SETHEADING 90
  REPEAT ( :SIZE/6)
  ( MAKE :X RANDOM ( :SIZE/2)
    PENUP FORWARD :X/2 PENDOWN
    FORWARD :SIZE - :X PENUP
    BACK :SIZE - :X/2
    SY YLOC ME - 2 )
  END
```

```
TO MANYCLOUDS
  PENCOLOUR 2 SX 10 SY 180
  CLOUD 60
  SX 100 SY 164
  CLOUD 30
  SX 190 SY 176
  CLOUD 65
  END
```

Here we have the beginnings of a microworld. With 255 turtles to play with, you can redefine some of them as cars, some as people, some as animals, some as aeroplanes, and then animate them with simple commands.

## 2. Using Lists.

Children can have lots of fun with programs that print random sentences.

Many children experience great difficulty in understanding grammatical categories, the differences between nouns, verbs and adverbs, etc. For some of them it is an inability to work with logical categories, but for many it is the inability to appreciate what grammar might be used for.

Using LOGO, you can put a child into the position of teaching a computer to manufacture strings of words that mimic English.

Firstly, they have to teach the micro to choose words from an appropriate list. To construct these lists, the child needs to classify the words he wishes to use into suitable categories.

For many children this leads to a real understanding that words can be sorted into different groups or sets and, consequently, not just to a conception of fundamental grammatical rules, but to an actual change in the child's formal relationship to grammar.

In designing random sentence programs, it is useful to have a procedure like PICKRANDOM that takes a list as input and outputs an item from the list chosen at random.

PICKRANDOM is a LOGO primitive in some implementations shortly to be released, but if it is not present in your version of LOGO, you can define it by using lists and recursion.

PICKRANDOM is implemented in terms of the associated procedures PICK (which outputs the Nth item of a particular list) and LENGTH (which gives the number of items in a list).

```
TO PICK :N :X
  IF :N = 1 OUTPUT FIRST :X
  OUTPUT PICK (:N - 1) ( BUTFIRST :X)
  END
```

```
TO LENGTH :X
  IF :X = ( ) OUTPUT 0
  OUTPUT 1 + LENGTH BUTFIRST :X
  END
```

```
TO PICKRANDOM :X
  OUTPUT PICK ( 1 + RANDOM (LENGTH
    :X )) :X
  END
```

Once you have PICKRANDOM it is easy to generate simple random sentences of the form NOUN - VERB by picking words at random from lists of nouns and verbs. For example,

```
TO VOCABULARY
  MAKE " NOUNS ( TEACHERS SCHOOLS
    COMPUTERS THEATRES DANCERS
    CARETAKERS )
  MAKE " VERBS ( FAIL TEACH CHANGE
    EXPLORE THINK CLEANUP )
  TALK
  END
```

```
TO TALK
  PRINT SENTENCE
  (PICKRANDOM :NOUNS)
  (PICKRANDOM :VERBS)
  TALK
  END
```

This would lead to the following kind of output;

## VOCABULARY

DANCERS EXPLORE  
 CARETAKERS TEACH  
 SCHOOLS CHANGE  
 TEACHERS CLEANUP  
 THEATRES THINK  
 etc. etc.

Obviously there are a number of ways that this can be refined. For example, you could easily make more complex sentences by adding more lists containing other parts of speech, such as adjectives and adverbs. Further sophistication could lead to matching singular with singular and plural with plural.

*Beware!*

LOGO is certainly no toy language, as some recent bizarre pseudo-implementations might suggest. The idea behind turtle graphics is to provide an early and easy entry into communicating with a computer. It is designed for total beginners with no prior mathematical knowledge. This conception is analogous to a very young child learning to speak. Turtle talk is like baby talk. It is a simple subset of a vast, rich and complex language – LOGO. Anything that does not offer this depth and richness is not LOGO. At best, it is merely a misguided educational dead end. At worst, it is a cynical rip-off.

## Simple LOGO in primary schools: a structured or unstructured approach?

**Helen Finlayson**

*Department of Artificial Intelligence,  
 Edinburgh University*

The Edinburgh turtle, a small robot which can move around the floor in response to simple commands in the LOGO language, is now becoming available to primary schools, as a means of extending the use of micro-computers. This paper addresses two problems. For what purposes can the turtles be used, and what teaching style should be adopted? In Edinburgh we have been using the prototype turtle with primary school children over the past 18 months, and this is a personal view based on this experience. There is also work currently under way developing the use of the turtle for mentally and physically handicapped children in special schools, but this is not considered here.

### 1. What educational purpose can be served by using a turtle in a primary classroom?

The turtle was devised as a medium for exploring mathematical ideas, to enable children to be mathematicians doing mathematical things rather than learning about mathematics (Papert 1972). It can be used to explore the properties of number, as linear or angular displacement; to investigate geometric ideas of regular shapes, and the relationship between the angle turned and the shape produced; and to see mathematics as being about the creation and recognition of

patterns. The turtle provides a learning environment where children can experience mathematical rules. It can thus also be used to back up normal classroom lessons, to give children greater understanding of abstract relations.

The turtle can be useful in other types of learning which are not overtly mathematical. It can be used to teach general problem attack skills, by the analogy to producing a turtle drawing. The child has the aim of producing a particular drawing with the turtle. He can learn to approach the problem in a top down manner, breaking it into subproblems by recognising component shapes and the relations between them, and then planning each of the subproblems, testing and debugging them, and finally putting the whole drawing together. These component processes can be used in approaching problems in any domain (Polya 1957), and practice in such turtle exercises can give valuable experience and act as a basis for the discussion of other problems.

A third approach is to use the turtle as an introduction to programming in the LOGO language. Important programming notions such as procedures, subprocedures and recursion can all be learned through the simple subset of the language available with the Edinburgh turtle run on the OKLOGO software. Because the programming activities all produce drawings, it is a very rewarding way to learn programming, and after this children can transfer to micro-computers and use the full LOGO language with great confidence. They can then approach other

fields of learning through programming, such as creative writing or grammar (Sharples 1978, Rowe 1978).

## 2. What teaching style should be adopted? – the theoretical issue

### a) *Discovery versus rule learning*

The theoretical issue of discovery learning versus rule learning has been debated for a long period of time, and the results of research into the problem have not been clear cut (Shulman 1970, Wittrock 1966). One argument for a discovery approach is that it produces a structurally different learning outcome; one which relates the new knowledge to the learner's existing knowledge. This 'relational' learning permits different approaches to a problem to be recognised, and allows for thinking about the problem (Skemp 1976). A rule learning approach, on the other hand, produces instrumental learning, which is not necessarily related to previous knowledge, and which tends to fail when different approaches to a problem are used. (Mayer and Greeno 1973; Egan and Green 1972).

### b) *Structured versus unstructured learning*

The turtle was clearly designed as a tool for discovery learning, and has been used by Papert and his fellow workers at MIT in a very unstructured manner, where children have been set no goals and given very little guidance on what to do. However, discovery learning does not necessitate an unstructured approach, and a great deal more guidance can be given to a child working with the turtle, without removing the essential exploratory element of experience. To what extent would this be a desirable and practical thing to do? Does the unstructured approach produce any undesirable learning effects, which could be overcome by structuring the learning situation a little more?

## Experimental observations

In one typically unstructured study at MIT (Papert *et al.* 1979) children spent a total of 25 hours working with LOGO over a 7 week period. The children chose their own projects and worked with them with the minimum of guidance. Most children learned a lot from the considerable amount of time that they were spending on the computers: however, for others the learning experience was less successful. An analysis of the children's work identified the mathematical and programming concepts which they had used, but in some instances these embedded concepts were not discovered by the child, because, although he had used them, they were not made explicit. Two of the 16 children



'Don't cry, love, we'll have him down in no time.'

studied in detail did not learn to program. One child used the computer only as a typewriter, and did not use the turtle at all. Another child limited her work to a 'microworld' using only a few values for input to both the forward and turning commands. In another study Perlman (1974), using younger children with a floor turtle and slot machine, found that children would keep repeating the same actions over and over again until they became bored, but would not try to explore different alternatives. These studies show that the learning implicit in a situation is not always grasped, and some children show an inability or lack of desire to explore a situation.

In Edinburgh we were interested in developing a system of using the turtle for the exploration of mathematical ideas. We spent the first term in a primary school using an open-ended unstructured approach with small groups of children working on the Edinburgh turtle. In the early sessions several observations were made which are summarised below.

1. It was very time-consuming; very few children were able to use the turtle in a half-morning session.
2. Children wanted to do complicated drawings before they were confident of the simpler actions.
3. They were much more successful when they had time to think and plan beforehand – one Primary 6 boy drew a parallelogram in his first session (after one class introductory lesson), having already planned it on paper.
4. Most children prefer to ignore their mistakes and go on to something else, rather than learn from them and debug their program.
5. Many children apparently got bored with the turtle, because they could not think of anything to do with it, or became frustrated because they could not get their overambitious plans to work, and so gave up.

In practice we found that children vary a lot in their exploratory behaviour, and on consideration it seems that most successful unstructured learning situations probably have a great deal of hidden structure in them. The problems children seem to have of not knowing where to explore and what to attempt, or of avoiding frustration, could be overcome by the intervention of a teacher, with just a hint or a suggestion to guide the child.

From our observations we put forward the tentative hypothesis that children have different levels of tolerance for uncertainty, which affect their performance in an unstructured situation. Children with low tolerance levels will seek to put structure into the situation to reduce the uncertainty or, if they are unable to do this, will seek to avoid or leave the situation. Confident children, on the other hand, will enjoy the unstructured learning environment and experiment, perhaps not always in the way a teacher would expect.

A great many of the observations of the way children operate with the turtle, from our experience, and from written reports (Noss 1983, Maxwell 1982), can be interpreted as 'coping behaviour' by children with a fairly low tolerance of uncertainty. Some children are well practised at finding structure to put into a situation, if it is not supplied. They will follow any vague suggestions from the teacher as closely as possible, use only a self-imposed limited set of commands, or copy what someone else has done. Their pleasure in recreating a design which is a direct copy of another (Noss 1983) may be less of the joy of creating and more of relief in successfully overcoming the uncertainty of the situation in an acceptable manner. They may also borrow structure from their more formal lessons, as in one school where children investigated reflections using the turtle, having just been studying them in mathematics.

### A guided discovery learning approach

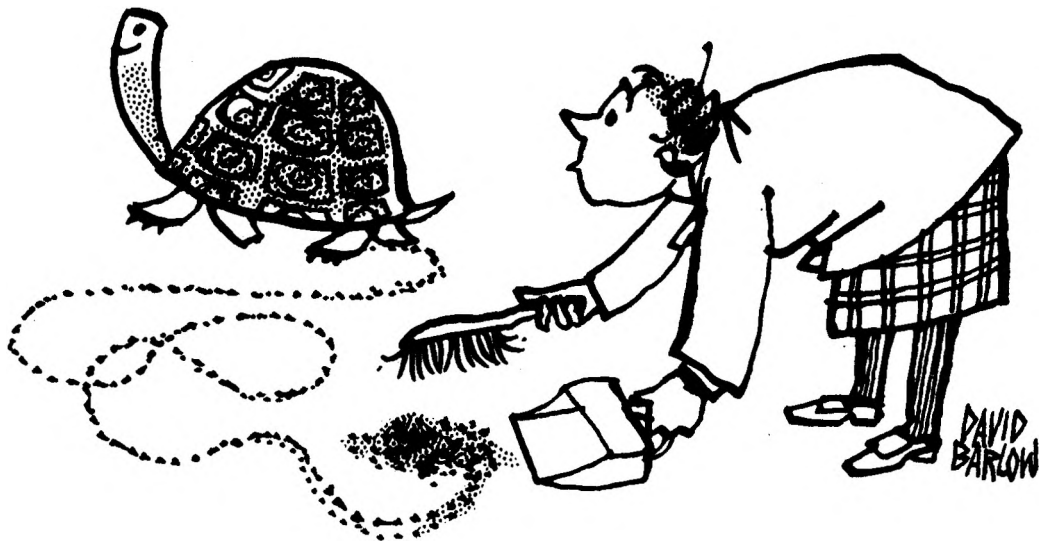
If, as the hypothesis suggests, many children do need structure in a learning situation, then it is surely better to provide it in a well-thought-out way, rather than letting them find it where they can. It also became clear that if children were to make best use of their time on the turtle, they should be encouraged to:-

- a) Think about what they were going to do beforehand, and possibly plan it on paper and keep a record of it.
- b) Be confident in the use of direct commands before going on to building procedures.
- c) Keep procedures fairly short to make debugging them a practical alternative.

Thus a structured learning approach has been adopted. Workcards have been developed to introduce new programming techniques in order of difficulty. Each card contains suggestions for mini-projects which children can pursue, although at all times they are encouraged to try out their own variations and approaches. These workcards are currently being used in an evaluation study.

### References

- Egan, D. and Greeno, J.G. (1973) 'Acquiring cognitive structure by discovery and rule learning' *J. of Educ. Psychol.* 64 (1) 85–87
- Maxwell, B. (1982) *LOGO at Crabtree School* (AUCBE, Hatfield)
- Mayer, R. and Greeno, J.G. (1972) 'Structural differences between learning outcomes produced by different instructional methods' *J. of Educ. Psychol.* 63 165–173
- Noss, R. (1983) *Starting LOGO; Interim report of the Chiltern MEP LOGO Project*
- Papert, S. (1972) 'Teaching children to be mathematicians v. teaching about mathematics'. *I. J. Maths. Educ. Sci. Technol.* 3 249–262
- Papert S. et al. (1979) 'Final Report of the Brookline Project' *MIT AI Memo No. 545*
- Perlman, R. (1974) 'TORTIS: Toddler's own recursive turtle interpreter system' *LOGO memo 9*
- Polya, G. (1957) *How to solve it* (Doubleday Anchor Books, New York)
- Rowe, N. (1978) *Grammar as a programming language* (Creative Computing)
- Sharples, M. (1978) 'Poetry from LOGO' *DAI Working Paper No. 30*.
- Shulman, L. S. (1970) 'Psychology and Mathematics Education' In E. G. Begle (Ed) *Mathematics education* (69th yearbook of NSSE, Chicago)
- Skemp, R. R. (1976) *Mathematics Teaching* 77 20–26.
- Wittrick, M.C. (1966) 'The learning by discovery hypothesis' In L. S. Shulman and E. R. Keisler (Eds) *Learning by Discovery: A critical appraisal* (Rand McNally, Chicago)



## Approaching LOGO

**Richard Noss**  
*Co-ordinator, Chiltern MEP LOGO Project*

For many teachers, particularly primary teachers, LOGO seems like a light at the end of the tunnel. Having stared at the endless succession of 'drill and practice' programs which make up the bulk of available software, it is easy to be lulled into the belief that LOGO will solve all problems by itself. It won't.

It is easy to underestimate the difficulties involved. After all, Papert's vision is of a computer-rich culture in which the restrictions of school are replaced by a learning environment in which the child is free to create his or her own curriculum. Unfortunately that is not the reality which we face at the moment. Instead, we have severe limitation on staffing and resources, a restrictive and examination-oriented secondary curriculum which has a spin-off into the primary's, and a situation in which most primary schools have yet to take delivery of their first micro.

In this situation, it is important to face up to issues of classroom organisation realistically. How many children should use the machine at a time? How much time should be devoted to LOGO? How can the machine be allocated fairly to classrooms within the school? What age groups – if any – should have priority on the machine? There are many such questions;

although many of them will begin to disappear as LOGO-speaking machines come down in price, they will remain with us in the immediate future. The intention of this article, however, is to concentrate on only one issue – and one on which the increasing availability of hardware will have only a marginal effect. How prescriptive should a LOGO curriculum be? More fundamentally, should there be a LOGO curriculum at all?

It is not surprising that this question has formed the basis of much of the discussion about LOGO. The essence of LOGO is that it allows the child a high degree of control over his/her learning environment. Ideas *belong* to the child; discoveries are the child's own. As a consequence, the issue of teacher-direction looms especially large. It is always difficult to decide when to intervene and when to hold back and allow the child to experiment, make mistakes and discover by herself. In LOGO, it is doubly difficult. It is not surprising that the two major centres for LOGO research throughout the seventies – at the Massachusetts Institute of Technology in the USA, and at the University of Edinburgh – have adopted widely differing approaches to this problem.

The MIT approach, under the direction of Seymour Papert, has concentrated on a project-based approach. Children have been encouraged to formulate their own problems, and to learn the necessary elements of the LOGO language in

their search for workable solutions. No attempt has been made to link LOGO activities explicitly with the existing curriculum of the school; indeed a central feature of LOGO is seen as its function of transcending the traditional subject borders.

Edinburgh's approach has been radically different. It has been based on sets of worksheets which introduce, in a linear way, a hierarchy of programming concepts (procedures, editing, inputs, recursion, list-processing, etc), and then uses these ideas to enrich specific mathematical topics, found in existing published materials.

Does this polarisation provide the primary teacher with a useful starting point in attempting to devise an approach to using LOGO in the classroom? The answer is a qualified yes. But in order to provide such help, it is necessary to distinguish between two aspects of the debate.

The first is a tactical difference. That is to say, it is concerned with a difference in materials used, type of activities, etc. To an extent, this has to depend on the age range and experience of the children and on the teaching style of the individual teacher. Learning to program is certainly not a trivial task – even in LOGO – and there is no disagreement about the need to teach the child how to use the language. Worksheets, workcards, word of mouth, whole-class discussion, children teaching each other; these are the tactical choices which have to be made. The Edinburgh approach is designed to teach programming through worksheets, while the MIT approach is founded upon the child discovering on his or her own. It is this contrast between child-discovery and teacher-direction which is usually seen as the major source of difference between the two approaches.

However, there is a more fundamental difference – a strategic rather than a tactical one. It is here that the real source of difference emerges. More importantly, it raises the first question which any teacher considering the introduction of LOGO into the classroom must answer – why LOGO?

The single biggest misconception held by teachers and others about LOGO is that it is designed to teach the ideas of geometry. In the first place, of course, turtle graphics represents only a small fraction of LOGO's capabilities (for example, only about 20% of Apple LOGO's inbuilt 'primitives' deal with turtle graphics). But this is not the main point. Let's just restrict ourselves to turtle activities, which will, after all, make up the major part of most children's early activities. To argue that turtle/LOGO activities are designed to teach geometry is like arguing that learning to read will enable you to understand a newspaper. Both are true. Both miss the point. Learning LOGO, like learning reading,

provides the learner with a powerful thinking tool, a powerful means of expression. Just to push the simile a little further, both activities provide a rich fund of material and experience *in the learning process itself*. The best way to learn to read is to read interesting books. The best way to learn LOGO is to engage in LOGO activities. On the way, you can't help but learn lots of other exciting things.

What are the implications of all this for classroom practice? Firstly, that the issue is not so much about how to structure LOGO activities. It is much more to do with ensuring that the richness and power of the language are offered to children; and that both the content and form of the activities are seen by them as being under their control. If that seems very general, it is supposed to be. It does not seem useful to offer a proposed LOGO teaching-sequence; children need to be free to develop their own. It might, however, be useful to offer some possible guidelines. In the spirit of LOGO, these are not meant prescriptively – merely as a set of personal suggestions based on observation of introducing LOGO into a number of primary classrooms.

1. Let the child keep control. The central feature of LOGO is that it allows the child to control her own learning. This does not preclude direction into productive activities offering help when necessary, or even suggesting useful avenues to explore.

2. Whatever happens, let the children try things out on their own first. LOGO gives a concrete meaning to the cliché that we learn from our mistakes.

3. Allow for plenty of exploration. This can be of the 'fundamental' type (e.g. discovering that the turn in a square is 90°), or it can involve exploring the effect of playing with or modifying a procedure. Children need time to explore.

4. Don't be afraid to offer models. Procedures written by friends or even the teacher are valuable. Following an idea and exploring it in a new and personal way is as valuable as thinking of the idea in the first place.

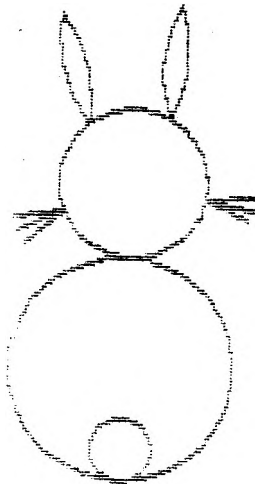
5. Don't introduce the child to new ideas out of the blue. New ideas are best based on need: if a child can see the relevance of a new idea, she is much more likely to feel it belongs to her, and to understand it.

6. Trust LOGO. Remember that it is a programming language, and not just a software package. As such, it offers, first and foremost, an environment which encourages the skills of planning, modifying, analysing and debugging – i.e. *thinking*. The specific content – geometric, algebraic, linguistic or whatever – is embedded in LOGO.



This last point needs illustrating. It is important because it is, as I pointed out above, at the heart of the divergence between the Edinburgh and MIT approaches. The following is a sequence of activities undertaken recently by two fourth year junior children. I have chosen this example because it represents a very elementary level of programming development. The only LOGO ideas the girls have met is that of procedure and subprocedure. They are working with a floor turtle. The plan was to draw a rabbit. There is no space to describe the process in detail; notes will have to do.

The question is: what did they learn?



1. How big should the body be? Repeat ? (FD 8 LT 5) but . . . How many repeats?
2. What about the head? How can we make the circle smaller? (The girls decided to find out and in fact spent two or three sessions researching the relationship between the inputs to forward, left and repeat).
3. Where should the whiskers go? How can they be incorporated into the head?
4. The tail must be drawn halfway round the body otherwise we'll have to retrace our steps.
5. What turn should there be at the top of the ears?
6. Why is the second ear lop-sided?
7. Why isn't the turtle facing in the right direction? etc. . . . and later . . .
8. Can we make a procedure for drawing circles so that we don't have to write a new one each time we want to change the size?

None of this takes account of the discussion between the two girls. By the end, words like 'input' had become useful tools to help them think about the idea of varying the relationship between distances, turns and repeats.

The girls did learn a lot about geometry. They also learned about planning, debugging, editing, and analysing problems as well as setting the scene for more sophisticated non-geometric mathematical ideas such as variables. I was about to write that what children learn from LOGO depends on one's priorities. In fact it depends on theirs.

---

## St Maurice School LOGO Project: a rationale

### Tony Mullan

In any educational activity there are two main considerations: firstly there is the philosophy of the activity, and secondly the methodology. Developments with the child as a programmer using the computer language LOGO are no exception. The philosophy is exemplified by the phrase 'the child is in control of the computer rather than the computer in control of the child'. This philosophy must be put into historical relief against the educational philosophy prevalent in America at the inception of the language. American educational philosophy had been heavily affected by the writings of the psychologist B. F. Skinner. Thus the main thrust of computer-aided learning was also governed by these stimulus-response precepts. Papert, affected by his years spent with Jean Piaget in

Geneva, decided he wanted to go in the opposite direction. Papert maintained that programming the computer was a fundamentally different way of interacting with a computer than traditional CAL. He suggested that the child who programmed the computer would have a far greater knowledge and understanding of the computer itself. He would develop problem-solving abilities and insight into his own thinking.

This begs several questions. Firstly, what do we mean by the 'child in control of the computer rather than the computer in control of the child? The computer in control of the child suggests a package that limits the element of choice the child has with respect to the events affecting him. In a drill and practice program the child has no control over the general flow of the program or the information presented. These decisions have already been made by the teacher

in consultation with a programmer. Within simulations children have more control over the information, but their control over the flow of the program and the consequences of any action on their part is again limited. Within an information retrieval program children have control over the data available to the program, and the questions they wish to ask of that data: they have no control over the structure of the program or the way it presents information. Using a programming language the child has no control over the structure, syntax or semantics of the language but has considerable control over the way he wishes to use the language. The parallel with regard to the use the child makes of his natural language is obvious. Thus it is more a question of degree of control rather than absolute control. Possibly the word 'control' is the wrong one to use in this context – rather we should consider how meaningful the conversation with the computer is in the different modes. What is perhaps more important is the qualitative aspects of the child's interaction. What is certain is that in suggesting the child program the computer one has to examine carefully what the objectives of the activity are.

'LOGO has its roots in Artificial Intelligence. Artificial Intelligence is not the study of computers, but of intelligence in thought and action. Computers are its tools, because its theories are expressed as computer programs that enable machines to do things that would require intelligence if done by people' (Boden 1977, preface)

It is perhaps here we might search for reasonable objectives. Simplistically, Artificial Intelligence seeks to model a portion of reality on the computer. In Papert's terms, it seeks to create a microworld. To take the creation of microworlds as our prime objective suggests several sub-objectives. These sub-objectives include the computer skills that are needed to model a micro-world. Procedures, sub-procedures, super-procedures, variables, iteration and control are part of these.

Traditionally, education has been concerned with the introduction of certain skills with scant regard for their final application. Here, in contrast, skills are introduced as they are needed by the child to continue with his learning. These skills are the ones needed for the child to develop micro-worlds of his own. The danger is apparent of taking Papert's phrase 'education without curriculum' too literally. The same danger was apparent in the 1960s over the 'readiness' criterion. It was argued that it was pointless to try to teach a child something before he was ready to learn. What was overlooked was the need to structure a learning environment to promote readiness. It was also

suggested that acceleration through the Piagetian stages was not possible. This was disproved by Bryant (1974). The same thing applies to programming in the primary school. The child is not going to use procedures or variables meaningfully until he has assimilated these concepts into his established schemata. It is also true to say that the majority of children will not spontaneously realise that a program can be constructed from procedures. The parallel of the use of zero in mathematics is obvious. It took humanity thousands of years to evolve the concept of zero, so to expect a child to 'discover' this for himself is asking rather a lot of the child.

### The Edinburgh and Brookline Projects

Examination of the two main LOGO projects to date would seem to indicate a differing approach to introducing programming to children. With the Edinburgh Project (Howe *et al.* 1979 and 1982), the objective was to examine the use of LOGO in the normal classroom, and to this end a methodology was evolved that included the use of worksheets in a structured approach. In the Brookline project (Papert *et al.* 1982), a different approach was used. This could be called a free expression programme that allowed children to develop their own projects and so to assimilate the computer skills outlined above. The Edinburgh project used 18 children who had on average 4 hours a week with LOGO on an individualised basis. The usual situation in the primary classroom is one computer between thirty children, and invariably that computer will have to be shared with the rest of the school!

There are considerable similarities between the projects. For instance both are Piagetian in their outlook. The teaching in the Edinburgh project was influenced by the following guidelines.

1. Learning should be built on existing knowledge.
2. The use of familiar concrete objects as models or metaphors makes teaching and learning more manageable.
3. Classroom teaching should be a partnership between teacher and child, with the child being given as much responsibility as possible for choosing, formulating and solving problems within the broad aims and objectives laid down by the teacher.
4. Feedback to the learner is crucial (Howe *et al.* 1979).

The main objectives of the Brookline project were the computer programming skills of sequential process, procedures, editing, debugging, variables, conditionals, loops, recursion

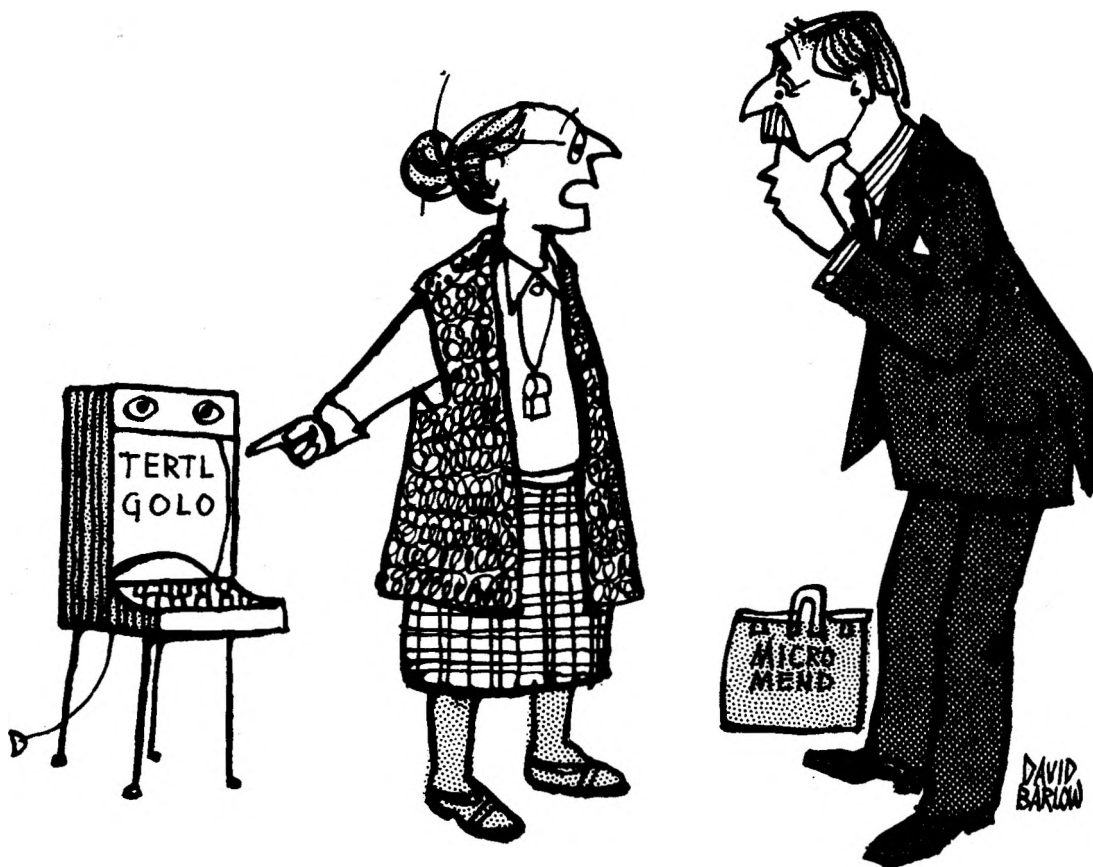
and interactive programs. As was pointed out it was in the methodology that the difference occurred. Again however, in the Brookline Project the children came four at a time to work on four computers, and had a teacher whose sole job was to interact with these children. A ratio of one to four is pretty good.

The main question to be answered is – should there be an overall structure within which the child can operate, or should there be a *laissez-faire* situation where the development of the requisite skills is a matter of hopeful conjecture? I am much persuaded by the former, where I as the teacher structure the environment in such a way that the objectives can be reached. Within this environment the children are able to choose from a set of projects devised by me or, if they wish, to develop their own projects. At all times we as teachers need to be evaluating the learning the children are achieving. I am much persuaded by Scriven's view (Scriven 1967), dividing evaluation into 'formative' that assists in the development of the curriculum and 'summative' in order to assess the merits of the curriculum after development. Papert is more revolutionary – in fact his criterion for failure is that the statisticians have to check for significance! We cannot ignore, as Papert would like to, constraints of economics and of parental and professional viewpoints. I am also aware that other teachers who do not have the knowledge

and experience may wish to embark on an exploration of LOGO. These teachers are more likely to do so with a methodology that is not too dissimilar from that which they practise in their present duties.

### St Maurice School

With the above ideas in mind the methodology I developed was structured to the extent that children had choice but the choices were directed towards the objectives of the project. I want children to gain the ability to write *non-trivial* programs. For instance, older children could write CAL software for the use of younger or less able children. I do not see turtle graphics as the beginning and the end of programming in the primary school. There are areas of mathematical modelling to be explored, not ignoring the possibilities of textual manipulation, etc. To do this, children need programming tools, a knowledge of the virtual machine of LOGO (O'Shea *et al.* 1981) and pertinent metaphors to relate this learning to other learning and schemata already present. The function of teaching in the primary school is to facilitate the transfer of intuitive learning into a symbol system capable of abstract manipulation. In this case the abstract symbol system we are aiming for is the



'If you ask me, it's dyslexic!'

computer program, and the reason we are aiming for it holds its rationale in Artificial Intelligence in that the child will possibly understand some aspect of reality in greater detail through modelling that reality on a computer.

The methodology I employed therefore was flexibly structured. The aims of the project were to develop an ability to write proceduralised programs, and to introduce the concept and develop the use of variables.

The following ten stages of development were built into the structure, children transferring from one activity to another as their need became apparent.

1. Introduction to the turtle and free exploration of the turtle commands in immediate mode.

2. Introduction to procedures and the evolution of the 'Turtle Dictionary' as a metaphor for thinking about procedures.

3. Understanding the connection between the *name* of a procedure and the *instructions* that define that name.

4. Introduction to subprocedures and their meaningful use in a program.

5. Introduction to variables and the development of a metaphor to think about variables in computational terms.

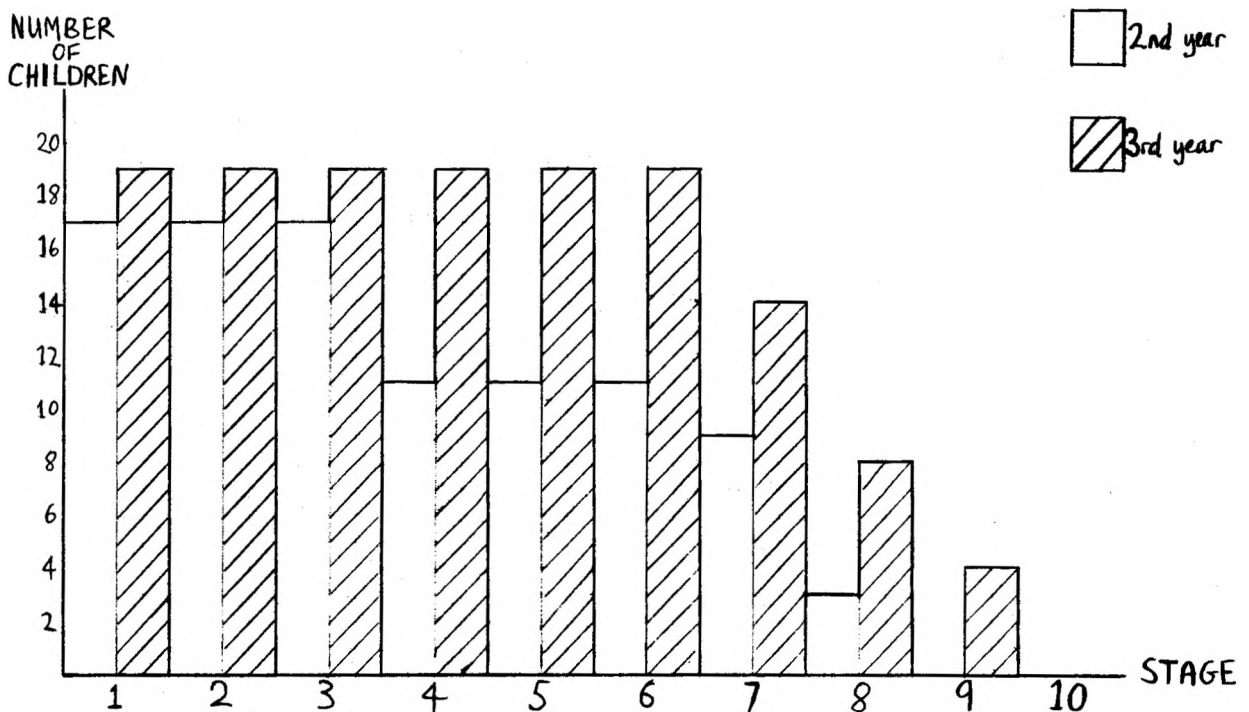
6. Writing simple procedures using variables.

7. Use of variables in procedures and subprocedures. This needs further development of the virtual machine to accommodate the transfer of variable values from the calling procedure to the subprocedure.

8. Using variables in superprocedures by declaring constants as variable values.

9. Using arithmetic operations on variables.

10. Introduction to the idea of stop rules and simple ideas in program control.



Graph to show number of children in identified stages.

The graph above gives my evaluation of the stages reached by the 36 children in the project.

The worksheet example gives an indication of the type of material the children had available. They include geometric and environmental projects. However, the children were free to develop their own projects. These invariably contributed to the overall aims of the project.

### Conclusion

Space precludes my describing the project in detail. The draft report runs to 25 pages! Generally throughout the project the motivation level of the children was high. They all made significant advances both in their learning regarding programming and in attitudes to school work.

Some of the evaluative comments of external observers may be pertinent to this discussion.

'The children not only handled the ideas put before them confidently but they also showed considerable confidence in using the micro-computer and the associated worksheets.' – David Owen, Primary Maths Adviser.

'The more able children were obviously deriving a tremendous benefit from this work for it involved theoretical and practical problem solving, and required discussion and co-operation. The less able were proceeding at a slower pace, as with any other form of school work.' – R. Butler, Headteacher, Laira Green Primary School.

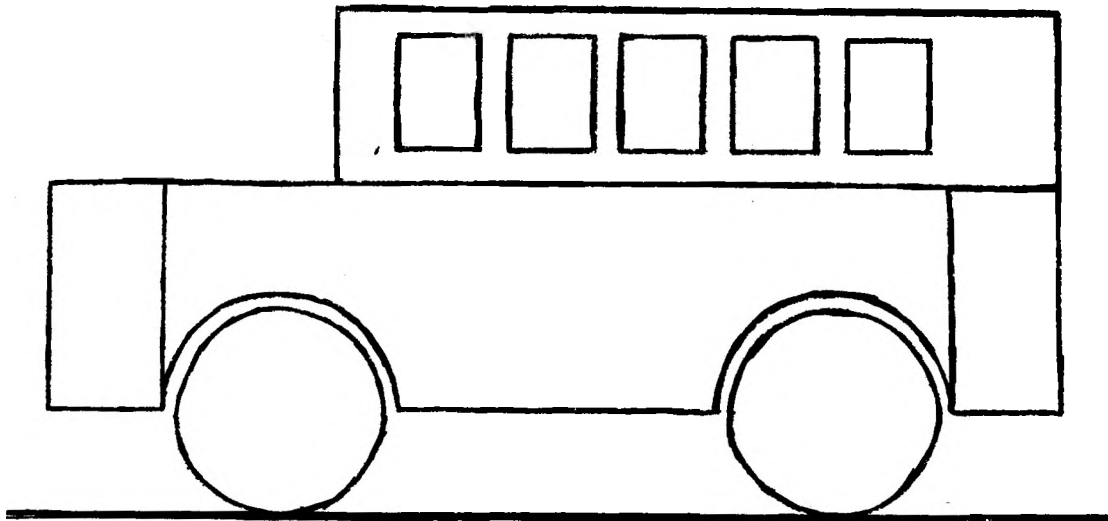
Regardless whether my argument regarding structure is right or wrong, it is apparent that the children are learning, and learning in a meaningful way.

### Bibliography

1. Papert, S. (1980) *Mindstorms – Children, Computers and Powerful Ideas* (Harvester Press)
2. Papert, S., Wall, D., DiSessa, A., Weir S. (1979) Final Report of the Brookline LOGO Project, *Logo Memo 53* (MIT)
3. Howe, J., Ross, P., Johnson, K., Plane, F., Inglis, R. (1982) *Learning Mathematics through LOGO programming; the Transition from the Laboratory to the Classroom* (University of Edinburgh)
4. Howe, J., O'Shea, T., Plane, F. (1979) *Teaching mathematics through LOGO programming: An Evaluation Study* (University of Edinburgh)
5. Bryant, P. (1974) *Perception and Understanding in Young Children* (Methuen)
6. O'Shea, T., Du Boulay, B., Monk, J. (1981) 'The Black Box inside the Glass Box: Presenting computing concepts to Novices', *International Journal Man-Machine Studies* 14 pp. 237–249
7. Boden, M. (1977) *Artificial Intelligence and Natural Man* (Harvester Press)
8. Scriven, M. (1967) *The Methodology of Evaluation in Perspectives on Curriculum Evaluation*

## Large Project 3

Use procedures you already have, and write any others you may need, to draw this or something like it.



This is just an idea – you may want to add things of your own.

Write down how you are going to solve the problem of writing this program.

# But what's the use of asking kids to have the computer draw a house?

**Catherine Berdonneau**  
*University of Paris VI*

This is a dialogue between a parent (P) whose child is working with LOGO at school, and a member (L) of the French LOGO Research Group.

The LOGO Project came from a common concern of researchers in Artificial Intelligence and in Education about what could be expected of computers in education. The work started in Cambridge, Massachusetts. Since the late 1960s the Project grew, including various disciplines (mainly mathematics, music, physics, biology, grammar . . .), and extending to various countries (Canada, Scotland, Australia, Germany, France . . .). The word LOGO denotes at the same time a theory of learning, a programming language, and a set of computer-controlled pedagogical devices.

The children talk a lot about LOGO when they come home, but not always in a very understandable manner to their parents. For instance, they refer to the turtle, which can either be a robot crawling on the floor (protected by a dome), or a small triangle on a television screen. One makes this 'turtle' move with very simple orders: FORWARD, BACK, RIGHT, LEFT: the turtle has a pen which can be in an upward position (so that moving the turtle does not leave a trail) or in a downward position (so that one can make the turtle draw designs).

One of the favourite projects of the children — be it in the States or in France — is to have the turtle draw a house. Hence the question:

(P) But what's the use of asking kids to have the computer draw a house?

(L) First of all, they are not *asked* to draw a house, they are completely free when they make up their project.

(P) What is a project?

(L) It is 'something' a group of children decide to realise with the computer. They are responsible for the choice of the subject. To find an idea is seldom a problem: they generally have too many ideas, and one of the first difficulties is coming to an agreement, either by making a synthesis of each child's suggestion or by a subtle negotiation among the group. The way they carry out their project also is up to them. And last but not least, *they* take the decision to consider the project as completed at some time.

What I find particularly interesting is that there is a very large variety of houses. Just in

one second grade class I observed this year, we find a flat, a terrace-roofed house as in hot countries, and a very traditional house with a pointed roof. None of these houses looks like the ones we had last year with fifth graders. Not only do the shapes differ but also the strategy for drawing the design. For the terrace-roofed house, kids respect the chronological order, building the walls first — a square — and then doing some carpentry for the roof. Fixing the pointed roof on top of the house is not at all simple. On the contrary, the building defies all gravitation laws, since the roof is attached to the first wall before the opposite wall comes back to the ground level, where the floor is drawn at last!

(P) Anyway, it would be much simpler to use a sheet of paper and a pencil to draw a house!

(L) Neither for us nor for the kids is the goal to draw a house. Our objective is to provide the pupils with an environment and a tool which allows them to be responsible for what they learn. The adult is not the one who possesses and transmits knowledge, but a pedagogue in the etymological sense of the word, the one who leads the child, in order to help him follow the most fruitful path in his travel toward knowledge. Children do somehow feel this, and above all intend to make the machine obey them. They are thus brought to think about what their reasoning produces, as the machine shows it, the computer being a kind of 'mirror of their thinking'.

(P) But do they learn anything?

(L) Let us analyse a few specific projects, all dealing with houses.

## The Orange Flat\*

Starting on an idea suggested by one's usual surroundings, the first task is to come to a simpler figure.

This means making choices among a multitude of details, keeping only the most important ones and forgetting all those which are not really meaningful (Figs. 1 and 2).

Then one has to determine a scale and use some technique for counting. Do not forget that these

\*Groups keep their work in coloured folders, and projects get referred to by the colour of the folder.

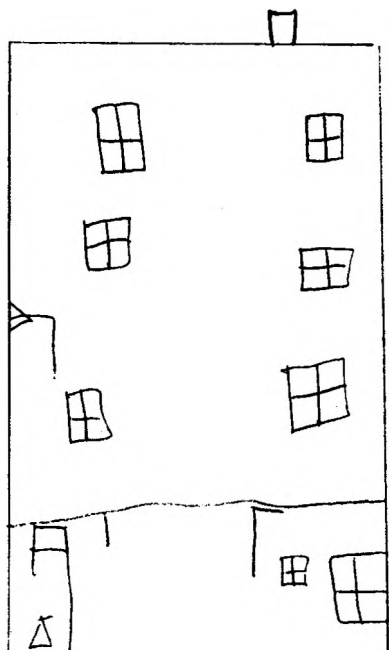


Fig. 1

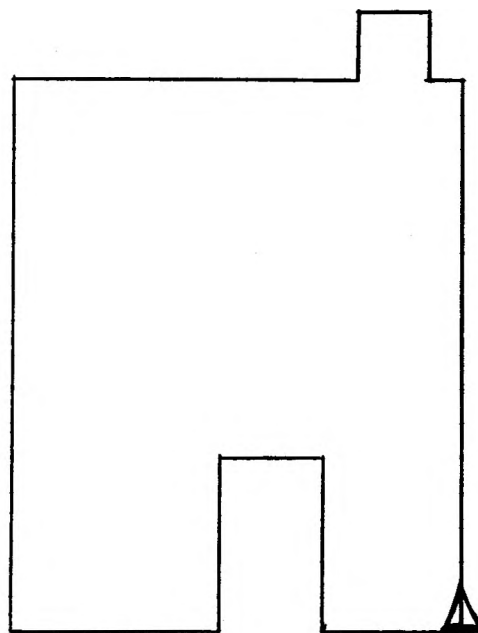


Fig. 2

children are second graders, a level where multiplication is still unknown.

Thus they use a translation method, reciting the rhyme on two tones: one, two (in a whisper) three (with a louder voice), four, five, six, . . .

This results in a very tiny figure. They change their scale and also adopt a new technique (which is a very good preparation for multiplication). Instead of iterating a translation, they use a repeated addition: five (for the first square), plus five is ten plus five . . . (Figs. 3, 4.)

Since reciting the rhyme and adding are not completely mastered at this level, it often results in discussion, or even strong disputes, regarding the results (be they right or wrong).

Mixing up right and left causes unpredictable shapes for the building, which provide occasions for laughter. The chimney, especially, is very hard. So what satisfaction when the desired pattern is obtained! (Fig. 5.)

**The Red House**

In contrast to the previous building, which was described sequentially, the red house is from the very beginning thought through to the end product in a structured way.

In fact, this group chose as their first project to draw a square with a side of 100 turtle steps. They decided, for their second project, to use this object. The reason for the odd shape of the roof is the instruction we gave the class for their first projects (the only restriction imposed on their imagination): 'Choose a design which follows the lines of your grid paper'. (Fig. 6.)

Putting the roof on top of the walls requires several attempts. The logic of adult builders does not necessarily match the children's logic and after having separately prepared stone-work and

1 square = 3 turtle steps

```
FD 48
LT 90
FD 3
RT 90
FD 6
LT 90
FD 6
RT 90
FD 30
```



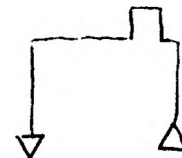
```
FD 80
LT 90
FD 5
RT 90
FD 10
LT 90
FD 10
RT 90
FD 10
LT 90
FD 50
LT 90
FD 80
```



1 square = 5 turtle steps

Fig. 4

```
FD 80
LT 90
FD 5
RT 90
FD 10
LT 90
FD 10
LT 90
FD 10
RT 90
FD 50
LT 90
FD 80
```



good

Fig. 5

Fig. 3

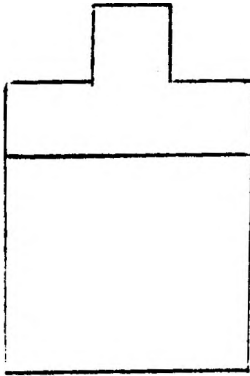


Fig. 6

carpentry (pre-fab is useful here too!), they try  
**ROOF**  
**SQUARE**

but the turtle doesn't have enough room on the screen to draw the second side of the square, and returns the message:

**OUT OF BOUNDS**

So they try the reverse order. (One of the girls warns the group, 'This is going to cause a massacre!')

Fig. 7

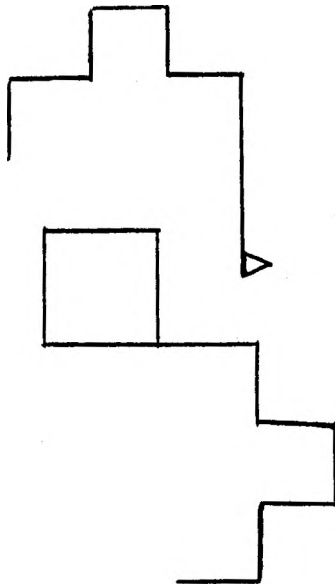


Fig. 8

During the next class, they only stay a very short time at the computer, just to try:

**SQUARE**  
**LEFT 90 FORWARD 100**  
**ROOF**

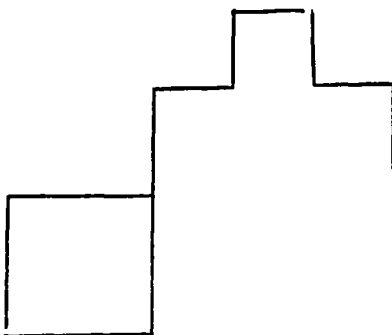


Fig. 9

No doubt, it's improving! Next week, the girls notice that the square is smaller than the roof. They consequently build a new square with side of 150 steps. But again they get an 'out of bounds' message. This time, a boy makes the group move ahead, suggesting that they start the square backwards. He therefore obtains one wall and the floor with

**BACK 150 RIGHT 90 FORWARD 150**  
 but the command **RIGHT 90** they typed faces the turtle in the wrong direction. He balances this with **LEFT 180**, thus denoting a real mastery of some properties of the rotation group. At that point, he is the only one having such mastery of this topic, and the others will need many more sessions watching how this works before they will be able to do the same.

### The Pointed Roof House

The same group now skips to a pointed roof house. A first problem is overstepped with the determination of the value which puts the turtle in the right position for the roof pitch. One of the boys explains, using his hands to express his thoughts: 'For the roof, it's like that  $\angle \triangle$ , so you just need half of this  $\perp \lrcorner$ . Since this  $\perp \lrcorner$  is 90, then this  $\angle \triangle$  is going to be 45.' (Fig. 10.)

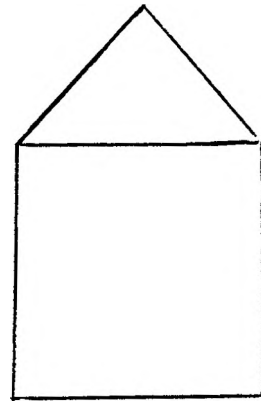


Fig. 10

Meanwhile they also enriched the masonry with a door. The roof will bring new problems to solve, such as angles in a triangle, side length in a triangle . . . They generally solve these kinds of problems in a successive approximations way, which is a general method widely and fruitfully used in mathematics and often not used in schools.

(P) Then, it is useful to let them have a computer draw houses?

(L) Yes, indeed: instead of contemplating mathematics, they make it.

We are grateful to Thomas O'Brien of Southern Illinois University, Edwardsville, for permission to reprint the above article, which first appeared in his project journal 'SEEDBED'.



# What do children think about turtles?

**Beryl Maxwell**  
Crabtree Primary School

I am now into my third school year using a floor turtle with junior children in a Hertfordshire school. Often asked by teachers and others my views on LOGO and floor turtles, I thought it was time the children themselves had a say. On various occasions I have asked children to write about their thoughts, and here are some of their feelings about turtling!



My first experience was a short three-week spell, and when the turtle had to be returned I asked some fourth-years to tell me what they felt they had got out of the turtle.

*Paul* 'What I got out of turtle. I did not get very much out of turtle because we did not get enough time. But on the turtle I learned about angles, because to draw the picture you had to get the angles right. You did not just get fun in drawing the pictures. It was much more fun watching other people drawing their ideas with the turtle and also watching them make mistakes.'

Here are other points they made.

*Richard S.* 'It also taught me to be methodical in my instructions.'

*James* 'If you get one degree wrong it puts your whole drawing wrong.'

*Barry* 'With turtle I learned quite a bit about angles because instead of an angle being  it was sometimes .

*Richard W.* 'I learned much more looking at it than being taught by voice.'

*Bobbie* 'The turtle was a lot of fun mixed with education. It is much more fun than normal school work though it was still hard work.'

The following school year the turtle returned for a whole term. In daily use, the children had much more experience. Here are some comments made by the third-year juniors on 'What I thought about turtle', after six weeks.

*Paul A.* 'On the turtle you can't say I'll do this, you have to think about it. The turtle is very good because it gets you thinking.'

*David* 'I think the turtle was fantastic. Because we did not have to do no work.'

*Matthew* 'I thought the turtle was a great success. I liked it because it stopped us from having our maths lessons.'

*Paul R.* 'It is not easy to use, in fact it is hard because you have to think which way you want it to go and you have to use your brain.'

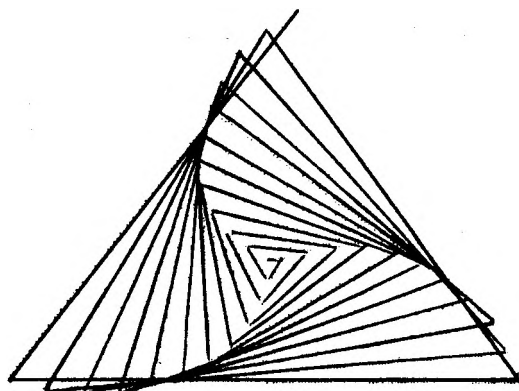
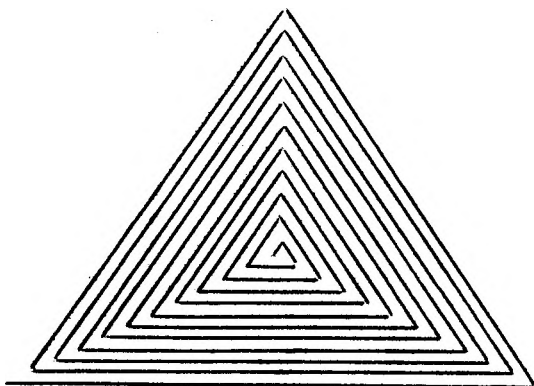
*Daren* 'When we were using the turtle I didn't get bored at all. I enjoyed every second of it.'

*Charles* 'If there was a lesson I would not like to miss it.'

*Richard* 'I enjoyed looking and learning about turtles. I have seen the BASIC language and I found that boring.'

Alison summed their views up by saying 'I thought the turtle was a fascinating machine with many educational and interesting possibilities. Because I enjoyed the turtle so much I would like to find out more about it and explore its many possibilities.'

This school year saw us with our own floor turtle. Now, after four months of turtle experience, my present fourth-year junior class expressed their views.



*David* 'I think we get a lot out of the turtle because it helps us learn about degrees and angles. My cousin has a ZX81 and the language we use is BASIC, but I find the language we use with turtle (LOGO) is much easier to understand. The turtle also makes you think about what you are going to do. I have already made one procedure called goalie. It has taken us a very long time and his legs were a problem but it works now.'

*Karen* 'At first I thought it was a load of rubbish but now I have changed my mind completely. It is easy to learn and does not confuse me.'

*Chris* 'I think LOGO is a very good computer language because it does not all have to be learned, because it is made up of the English language. The turtle enables you to have control over what you are doing.'

*Peter* 'I feel myself that the turtle is not just a geometrical robot that draws but it helps children get on with each other as in using turtle group decisions are vital.'

*Rosamund* 'It helps me learn things in an interesting way. I don't think anyone could *not* learn things from the turtle. Just sitting at a desk doing sums is not nearly as inspired as the turtle. If we just had a screen turtle I don't think I would enjoy it nearly as much as I enjoy having both the screen and floor turtle.'

*Linda* 'It is good fun, but you also have to do quite a bit of mathematical thinking, so you learn a lot.'

*Jonathan* 'Turtle LOGO is excellent. I find it much easier to use than normal BASIC language. My favourite part of turtle work is simulating how our planning will work before seeing it on the screen and floor. The turtle is very rewarding and when all plans are perfected the excitement builds up and if you could hear the shouts of delight when everything comes out right! To sum this up: (a) maths isn't just using abacus and cards; (b) the turtle is a precise, intricate machine for our age group.'

*Philippa* 'I get a lot of fun out of working out what we are going to draw on the turtle when it is our turn. It is also fun trying to find out where you have made mistakes (if you do). When you finally get your picture right it is very pleasing and satisfying. I would recommend the turtle to any junior school. Many people may think the turtle is a toy but it is not at all. It is designed to make learning a little pleasanter. The turtle is a very good name for the turtle because if it was called a robot many girls would not use it and if it was called a doll many boys would not use it. Personally I think the turtle is a good buy.'

*Jonathan W.* 'I think that the turtle and the language LOGO is fun but it also teaches you how to do procedurs and how to use computers. If you plan the day before, you have a longer go the next day and also if you get it wrong you have time to change.'

*Kirstie* 'From the turtle I have had great enjoyment and interest. The turtle is not all that easy to control, because when the command is entered, it is final. Also it is quite difficult to know what command to put in, and how far to move or turn.'

*Mary* 'LOGO is a good language and easy to understand. Recently I've been having extra fun with the turtle, taping my own procedures. When we make procedures we draw our picture on graph paper. We then give our instructions to the turtle. Our picture often goes wrong. It is great fun and a challenge to find the mistakes. So far our group has made an elephant, monkey, giraffe and a lion. I think that the turtle has helped me with my maths. It has also helped me to get used to computers. I can now understand how a computer can store things in its memory and use tapes, etc.'

*Andrew C.* 'LOGO is a computer language designed specifically for learning. In Crabtree we are very fortunate to be able to use the turtle, and I know most schools would die to have one. Although there are thousands of programs for computers I prefer LOGO. I have learned a lot from it.'

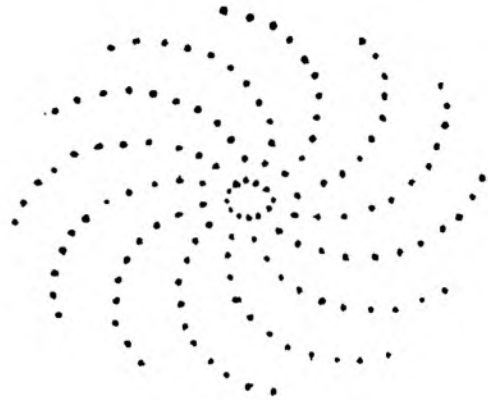
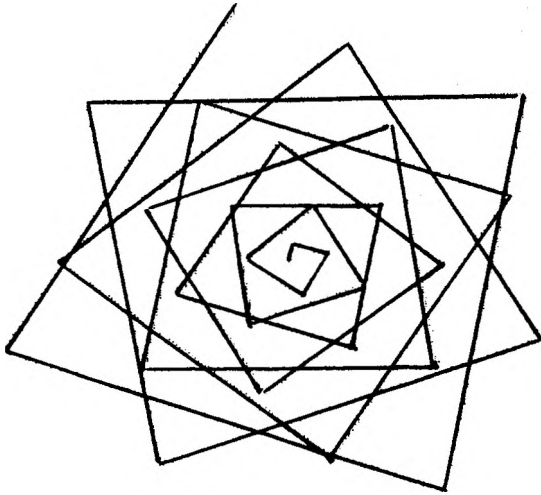
*Paul H.* 'The turtle is great to use because you have more freedom than when you work in a book.'

*Tim* 'I think the turtle is FAB. We are put in groups of four to experiment on this computerised turtle. To draw a simple square all you have to do is this:

Pen on. Repeat 4: FOR 20  
RIGHT 90  
LIMIT Pen off.

and as if by magic you will have a square!'

*Jane* 'I think the turtle is very useful. It helps to improve skill in both design and mathematics, and is a good start towards using more complicated computers. It has helped me to understand degrees. The turtle uses a very simple language called LOGO. This language includes the easiest of words, that are not difficult to remember when you are using the computer. It teaches you how to use more advanced computer terms, at the same time. I have enjoyed the turtle very much. It is fun to make procedures, and 'de-bug' them afterwards. It provides an atmosphere of enjoyment and excitement.'



## Turtle graphics in the secondary school

**Kim O'Driscoll**  
*MEP South West*

The initial experience of using a microcomputer is an important one, especially for the older learner whose confidence in approaching a new learning situation has been diminished by little success. My first experiences of using LOGO turtle graphics with teachers who had never used a micro before promoted the idea of using it as a medium for computer familiarisation with lower secondary school pupils.

A local secondary school teacher is presently teaching a course, Introduction to Computing, to the lower school. The course consists of ten once-weekly half-hour periods. He was enthusiastic to try out Apple LOGO, and so was lent a complete system for a two-week period.

The tentative conclusions we can draw after such a short period are that the pupils (coming to him in groups of thirty) responded well to turtle graphics. The good response extended right across the ability range. Turtle graphics was introduced using Bigtrak, and also by considering the possible ways in which a person, blindfolded, could be given instructions to move around a classroom avoiding obstacles.

The most important point seemed to be that the pupils developed well at their own rate on their individual programming projects. In practice the problem of having only one micro to thirty pupils presented fewer difficulties than had been envisaged. Each child had sufficient

time to test out a particular piece of programming whilst the others tentatively drew out on squared paper what they thought might happen to the turtle, or rewrote their programs, discussing their work with others.

Perhaps the most encouraging sign is that pupils of all abilities developed sufficient enthusiasm to carry on working out of school time and those who had progressed quickly were not held back by others. It would certainly seem that LOGO could be an important tool for a brief computer familiarisation course. A final observation is that primary school children who had already met LOGO would without difficulty move to the level at which they had been previously working. In this way, the transference of activities from primary to secondary schools could be made less problematical.

The ideas raised only tentatively in this note will be explored in more detail throughout the next term, together with an attempt to use some of LOGO's list processing facilities. Significant findings will be documented.

I would be very interested in hearing from anyone who has been using LOGO in a similar way, as an exchange and comparison of experience can only be of value. I can be contacted at:

Computer Centre  
Plymouth Polytechnic  
Drake Circus  
Plymouth PL4 8AA  
Tel. (0572) 21089

# Languages for young programmes

Tim O'Shea

*Director of the Microelectronics in Schools Project, Open University*

## Introduction

In this paper I shall briefly discuss five criteria (viz. syntax, semantics, mechanism, culture and practicality) which should be applied to any programming language that is being considered for use by young or novice programmers. I shall apply these criteria to four programming language families – BASIC, Prolog, Smalltalk and versions of LOGO that are often touted for school use. In this short paper I cannot properly explain these languages: further details can be found in O'Shea and Self, 1983. None of these languages is a 'best buy': the given criteria are helpful if you are making a choice to fit some particular educational context.

Some properties that it is reasonable to expect of any first programming language include simplicity, transparency, consistency and visibility. That is to say, the language should be simple to use and read and it should be possible to 'see the works' whilst programs are running. There should be a simple conceptual model on which the language is based that relates to the events that occur when programs are run. There should be a reasonable level of user aids provided, including some degree of prompting and spelling correction. Finally, the names of the constructs provided in the language should be very carefully chosen and the error message should be carefully worded and relate directly to a simple conceptual model for the language.

## Five criteria

The first important issue is what **syntactic rules** you have to use. The syntax is reflected in the rules for putting together statements and in the punctuation of the language. The syntax should be simple and expressive, and different things should *look* different. It is important to check the relationship of the syntax to that of school mathematics, and also its ease of teaching and being checked. One good test of syntax is to see how easy it is to implement a spelling corrector.

The second important issue, which relates closely to the first, is the **semantics** of the

language. This depends on the rules for determining the *meaning* of programs. Considerations under this heading include the degree of ambiguity in the language and the availability of simple conceptual models. One should check how easy it is to read other peoples' programs, and regard a need for extensive commenting as an indication of deficiencies in the language. Another important question is whether the editor and filing system follow the same general rules as programs in the language itself.

The third issue is what **mechanisms** are available to explain what is happening while a program is running. In particular, it should be possible for you to put yourself in the place of the computer and run a program in your head or on paper.

The fourth issue to be considered is the **culture** from which the language developed. Is it primarily scientific, technological, commercial, psychological or educational? What are programs in this language in general written for?

The final issue is that of **practicality**. The most elegant programming language is useless in the classroom without textbooks, a reasonable response time and a reasonably-sized implementation that can be used with the actual micro-computers found in schools.

## 1. BASIC

The syntax of BASIC is surprisingly complex. There are lots of idiosyncratic rules and sometimes the same symbol (e.g. '=' ) means different things in different contexts. Quite often the space character is used as a delimiter. This of course is problematic as (unlike other symbols such as ';' or ':=' ) it is not printed. The semantics of the commonly available BASICs are very unclear and depend on particular implementations. Some of the structured BASICs are even more of a problem, as they combine the sort of jump or GOTO statements that turn programs into spaghetti with procedure-defining facilities that do not relate easily to mathematical functions or predicates. The simplest mechanism to use for BASIC is assignment to variables. To explain the execution of BASIC programs a useful analogy is pigeon-holes or letterboxes.

BASIC is essentially a teaching version of Fortran, a language which nowadays is recognised as having serious design errors. One important

aspect of BASIC's culture is the teletype (rather than the screen, keyboard and pointing devices) as the means of creating and running programs: this has had a bad effect on the design of BASIC editors. Another important aspect is a view of learning where the emphasis is on activities which are easy to achieve rather than on mastering important concepts. The practicability of BASIC cannot be challenged. It is available on all the common microcomputers and there are some quite good textbooks. Better editors are becoming available but in general the error messages are obscure and unrelated to any simple conceptual model of the 'BASIC machine'.

## 2. Prolog

Prolog has a simple consistent syntax which requires the use of parentheses. Some dialects of Prolog distinguish variables and literals by the use of lower and upper case leading letters. This obviously confuses beginning programmers. Prolog clauses do not have to be ordered, which makes it possible to have a comparatively simple syntax. The semantics of Prolog are also very sound and the meaning of individual clauses can be read off in a straightforward way. But the mechanism by which a Prolog interpreter handles a set of clauses of any size is very hard to follow. If any device (e.g. the 'c t') is used to control the search through a set of Prolog clauses then it requires a high degree of technical sophistication to understand or explain what is happening.

Prolog stems from an old-fashioned sixties Artificial Intelligence approach, based on the use of automatic theorem proving. It is currently undergoing a great revival as a result of Japan's declared intentions to use Prolog in its fifth-generation project. Other aspects of the Prolog culture include an emphasis on uniform methods and notations and explicit attempts to teach logical reasoning techniques. In education Prolog has started to be applied to a variety of database-related topics in the humanities. Prolog is now available on a variety of microcomputers. However, the existing user interfaces are very poor and there is a shortage of good textbooks.

## 3. Smalltalk

This sophisticated simulation system is not generally available but is worth following as it represents a very high standard of graphics facilities and user interface design. It has a baroque syntax with very many markers, conventions and icons. The semantics of Smalltalk include general objects whose properties can be inherited by more particular objects and message-passing between objects. It is difficult to explain

the mechanism by which Smalltalk programs run. One explanatory device is to use an 'infinite' number of actors communicating via a blackboard in the sky.

Smalltalk comes from a high technology culture with a strong emphasis on aesthetics, so Smalltalk screens are attractive to look at, and the computing facilities are the most powerful that it has ever been suggested should be available to the individual user. An important aspect of the Smalltalk culture is the concept of the computer as 'dynabook' – that's to say, a portable device which can be used for music, drawing and writing as well as the conventional applications of computers. Smalltalk is not likely to be available for school computers in the next few years so it cannot score any points for immediate practicality.

## 4. LOGO

This language has a quite simple syntax particularly when infix operators (such as '+') are avoided. Prefix operators (e.g. Add 34) make it possible to avoid the need for extra syntactic markers. The semantics of LOGO procedures are made straightforward by the way that user-defined procedures obey *exactly* the same rules as primitive procedures already available in the system. The main difficulty with the semantics of LOGO is that drawings or tunes generated by procedures are side-effects and therefore not directly manipulable after creation. The mechanism used for drawing – the 'turtle' – is self-explanatory. However, to explain procedure execution it is necessary to appeal to the regular stage army of little men.

The LOGO culture is part Artificial Intelligence Programming (the Lisp language) and part Piagetian developmental psychology (body schemata leading to the turtle). This combination of one of the best programming languages with a line of educational thinking that is orientated to helping children to learn by doing has resulted in a very healthy culture which has generated a range of pleasing educational applications in mathematics, language, problem-solving and many other topics. There are now some good implementations of LOGO (for example, for the Apple II and Research Machines 380Z) and some good textbooks.

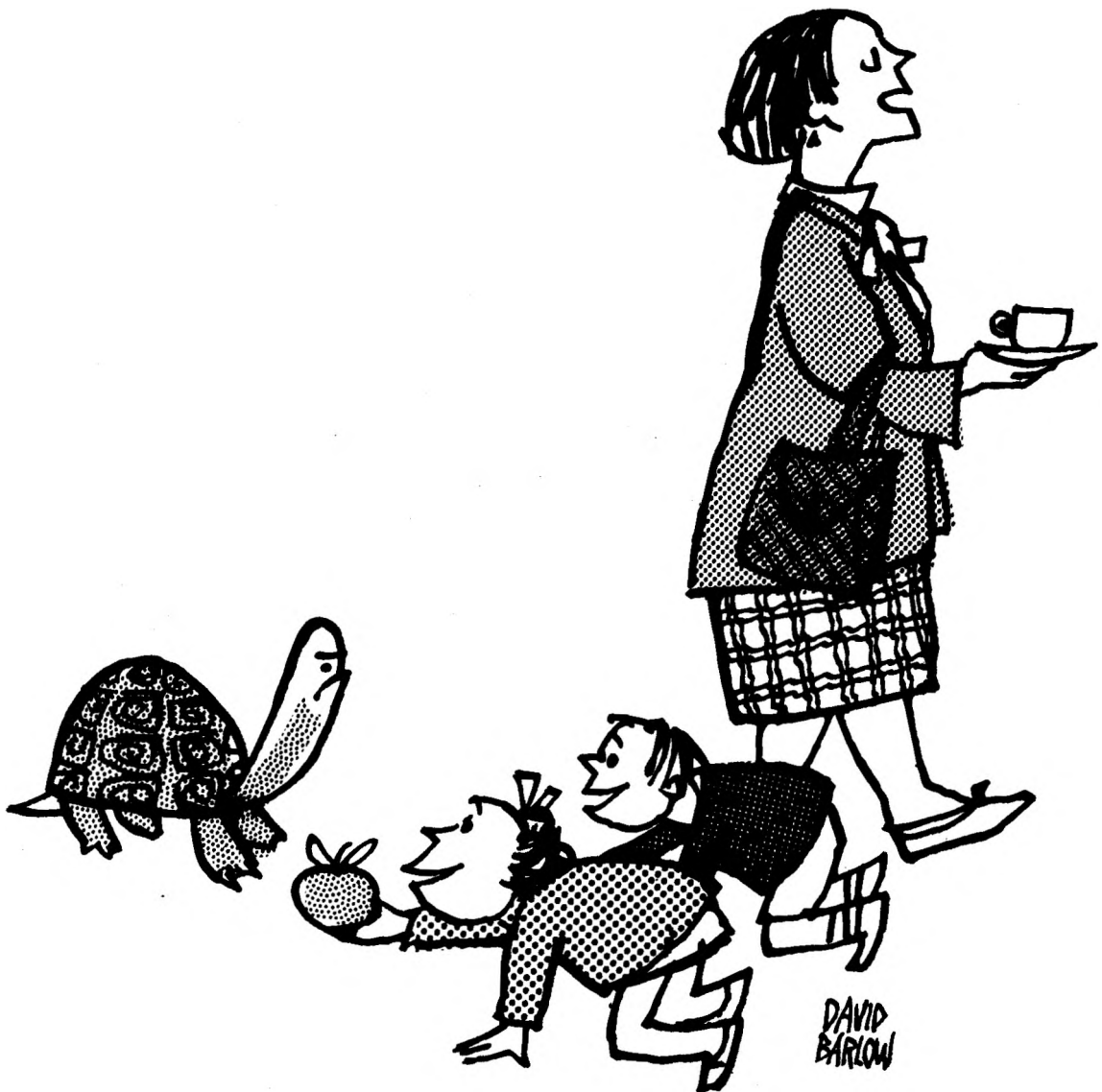
### Language families

It is important to realise that there is no 'British Standard Prolog'. All four languages discussed come in related families whose individual members can vary considerably. For example there are 'structured' BASICs and there are Prologs with turtle graphics. There are also

impostors lurking on the sidelines of any successful language. These can usually be distinguished by their use of prefixes (e.g. *Rosettatalk*) or postfixes (e.g. *LOGO Challenge*).

For example, to qualify as a member of the LOGO family we should expect a language to have (amongst other features) turtle graphics; list-processing; procedures that obey the same rules as system primitives; facilities for defining recursive sub-procedures; and logical predicates. It is important to ensure that you are dealing with a genuine member of the family because otherwise your pupils will not have access to other aspects of its culture. For example, using a fake LOGO makes it impossible to follow the problem-solving strategy of breaking a problem into sub-problems (via procedure definition) that is the basis of so much LOGO work.

However, even within a family there is variation. My personal list for good LOGO qualities includes requirements for a good screen editor, preservation of the name value distinction in procedure titles (e.g. TO 'POLY' SIZE and not TO POLY SIZE), error messages that relate to a simple conceptual model for LOGO, the ability to pass procedure names as arguments, no English noise words (e.g. 'AND' or 'THEN'), real (i.e. multi-level) lists, no infix operators and genuine extensibility (so that the editor or filing system can be augmented). But features like the above are matters of taste and my main motivation is teaching and explanation. Accordingly I value consistency (say in procedure titles) and simplicity (say in banning infix) as it then becomes easier for the learner to read programs and predict their behaviour. Obviously if you



*'If you want it to move, try the on/off switch.'*

wished to reduce typing load or make LOGO similar to some aspects of school mathematics you might wish to make other choices.

### Cheese metaphor

LOGO and BASIC are both mid-sixties programming languages and they took over ten years to 'mature' into languages that could develop reasonable user interfaces and be taught. Smalltalk and Prolog are comparative youngsters from the seventies and so we should expect them to become easier to use in the future. Following a cheese-maturation metaphor we have:

- (a) BASIC – 'processed cheese slices', easy to use, synthetic and unpleasant taste.
- (b) LOGO – 'mature farmhouse cheddar', old-fashioned, reliable, keeps well and tasty.
- (c) Prolog – 'smelly goat's cheese', acquired taste, growing market, admired by connoisseurs and French.
- (d) Smalltalk – 'Lymeswold', produced by best cheese makers, combines features of other popular cheeses, looks promising and experts waiting for it to be mature enough to eat.

Returning to our original five criteria, BASIC scores best for practicality but I would avoid it because its impoverished syntax and semantics lead to bad and clumsy programming habits, its mechanism is hard to explain and it is associated with an educationally impoverished culture. Prolog wins easily on syntax and semantics and obviously has potential for many applications in the humanities and languages. Smalltalk must be our hope for the future – a very expressive language designed for interactive uses via and with computer graphics. Versions of LOGO vary but the real ones are closely associated with an educational philosophy which will please many primary school teachers. Its sophistication as a computer programming language (as exhibited in its Lisp-based semantics) makes it appropriate for computer science teaching to older children. One way of expressing what is wrong with BASIC is to note that it obscures what happens 'in the computer' when programs are run, but does not offer any high-level expressive power. The other three languages are both more powerful and more transparent. In general, attempts to graft features from one programming language to another (e.g. BASIC + procedures, LOGO + databases, or Prolog + turtle graphics) lead to a hybrid mutant with a great loss of simplicity and consistency. Grafting languages together has very bad effects on the syntax, semantics and mechanisms.

### Conclusions

Choosing programming languages is an important and difficult task. A language limits or expands what you and your pupils can express and reason about. It also connects you to an educational philosophy which will be expressed in work sheets, textbooks and future extensions to the language.

The syntax and mechanism affect the learnability of a language and must be examined carefully. One standard error is to add a sugaring of English words (e.g. 'IS', 'THEN', 'AND') to improve readability, which (while offering apparent readability) confuses the user by obscuring the real syntax and mechanism and may lead a user to add his own English words and be confused when they are not understood by the computer.

The culture will condition the amount of experience of different teaching applications and the quality of educational thinking. From this point of view LOGO is the clear winner. The semantics affect what is easy to express and understand; from this point of view BASIC is useless (except for a subset of old-fashioned computer science teaching), and the other languages offer different advantages.

As far as practicability goes, there is no doubt that BASIC scores best on this now and we can reasonably expect Smalltalk to become the best (perhaps ten years from now) because of the quality of its user interface design and graphics. If I were a teacher I would want to work with LOGO or Prolog depending on the teaching task, and I would agitate for real LOGOs to be available for commonly available micro-computers and for the implementation of Prolog to be cleaned up from an educational point of view. As a very rough rule I would hope to use LOGO with younger children (3–7 years) for problem solving and languages, Prolog with middle-school children for humanities and LOGO with older children (14–18 years) for mathematics and computer science.

---

### Reference

Tim O'Shea and John Self (1983) *Learning and Teaching with Computers*, Harvester Press.

# LOGO implementations

The following list describes full LOGO implementations available, or shortly to be, as at October 1983.

## Apple II+ and Apple IIe

### *Apple/LCSI LOGO*

64K machine, disc based, about 14K user memory available.  
Cost c. £122.

Apple (UK) Ltd, Eastman Way, Hemel Hempstead, Herts HP2 7HQ. Available through Apple dealers.

### *Terrapin LOGO*

64K machine, disc based, about 11.5K user memory available. (Synonymous with Krell LOGO, almost.)

Cost c. £95.

Terrapin Inc., 380 Green Street, Cambridge, Mass. 02139, USA. Available through some Apple dealers.

## Atari 400 and 800

Atari LOGO, 48K machine, ROM cartridge based, 18K+ user memory available (Atari 800).  
Cost possibly c. £60, available late 1983.  
Atari International (UK) Inc, PO Box 407, Blackhorse Road, London SE8 5JH. Available through Atari dealers.

## Commodore 64

CBM/Terrapin LOGO, 64K machine, disc based, 14K user memory available.  
Cost unknown, due early 1984.  
Commodore Business Machines (UK) Ltd, 675 Ajax Avenue, Trading Estate, Slough, Berks SL1 4BG. Available through Commodore dealers.

## IBM PC

### *IBM/LCSI LOGO*

128K machine, disc based, amount of user memory unknown but large.  
Cost unknown, available late 1983.  
IBM retail centres around the country.  
Available through IBM retail centres and IBM dealers.

### *DR LOGO*

256K machine, disc based, 192K user memory available on 256K machine (can be 1 megabyte on fully expanded PC).

Cost c. £100.

Digital Research (UK) Ltd, Oxford House, Oxford Street, Newbury, Berks RG13 1JB.

### *Waterloo LOGO*

128K machine, disc based, amount of user memory unknown but large.

Cost £145.

University of Waterloo, Ontario, Canada.  
Available in UK through Roundhill Computer Systems Ltd, Axholme, London Road, Marlborough, Wilts SN8 1LR.

## Mattel Aquarius

Aquarius LOGO, 48K machine, ROM cartridge based, amount of user memory unknown.

Cost £30, available early 1984.

Mattel Electronics (UK). Available through Mattel dealers.

## RML 380Z

RML LOGO, 56K machine with high resolution graphics board, disc based (can be a server to a network of 480Z with special version), 20K of user memory.

Cost £60.

Research Machines Ltd, PO Box 75, Oxford OX2 0BW. Available from manufacturer.

## RML 480Z

RML LOGO, 64K machine, ROM cartridge based – possibly disc based also, amount of user memory unknown.

Cost c. £60, available late 1983.

RML address above.

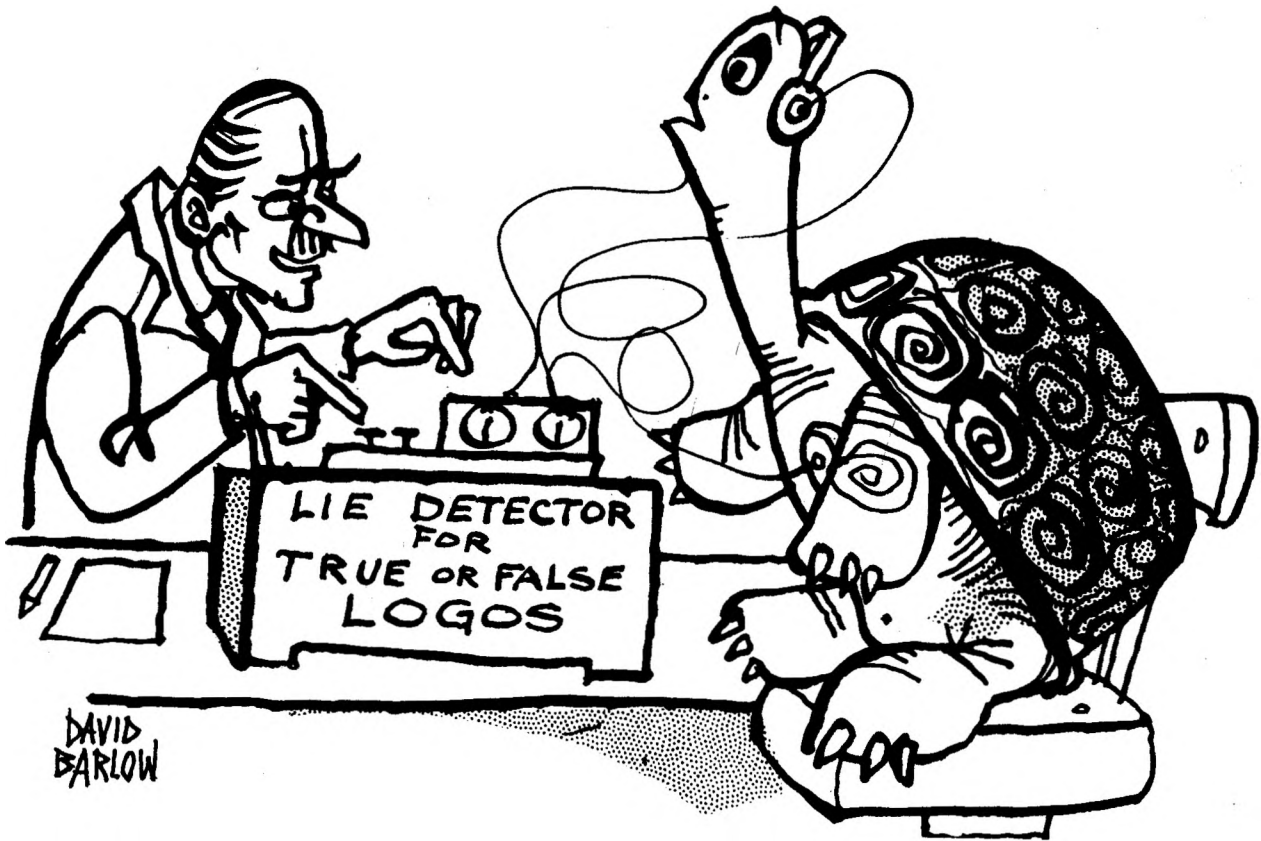
## Sinclair Spectrum

Sinclair LOGO, 48K machine, tape based or ROM based, 8K user memory available.

Cost c. £30, expected early 1984.

Sinclair Research Ltd, 25 Willis Road, Cambridge CB1 2AQ. Available through Sinclair and Sinclair dealers.





#### Texas Instruments 99/4A

TI LOGO II, 48K machine, ROM cartridge based, amount of user memory unknown.

Cost unknown, new LOGO II version available November 1983.

Texas Instruments Ltd, Manton Lane, Bedford MK41 7PU. Available through TI dealers.

Note: by full LOGO is meant an implementation which includes turtle graphics,

but is based on list processing. Full LOGOs have a high degree of consistency in the way in which they work.

There is no full LOGO presently available for the BBC computer; a recommended surrogate until full LOGO is available is the *DART* program, available from AUCBE, Endymion Road, Hatfield, Herts AL10 8AU, at cost of £13 (or £6.50 to members of BLUG).

*Derek Radburn, BLUG*

## Books

The following books, all currently available in this country, say more about LOGO.

Firstly, the original classic and a compulsive read:

### *Mindstorms: Children, Computers and Powerful Ideas*

Papert, S. (Harvester Press, Brighton, 1980)

£9.95 hardback (ISBN 0 85527 163 9)

£3.95 paperback (ISBN 0 71080 472 5)

Arguably the most important source of information about the rationale underlying LOGO – why it goes beyond being just another programming language.

### *LOGO for the Apple II*

Abelson, H. (BYTE Books – McGraw Hill, Peterborough, N.H., 1982)

\$14.95 paperback (ISBN 0 07 000426 9)

A book providing a grounding in LOGO (the MIT version used on the Apple II computer).

### *LOGO Programming*

Ross, P. (Addison Wesley, London, 1983)

£7.95 paperback (ISBN 0 201 14637 1)

The first British book on LOGO, written by a member of the Artificial Intelligence Department at Edinburgh University.

### *Discovering Apple LOGO*

Thornburg, D. (Addison Wesley, Reading, Ma., 1983)

£9.95 paperback (ISBN 0 201 07769 8)

Deals with turtle graphics.

### *Learning with LOGO*

Watt, D. (BYTE Books – McGraw Hill, Peterborough, N.H., 1982)

\$14.95 paperback (ISBN 0 07 068570 3)

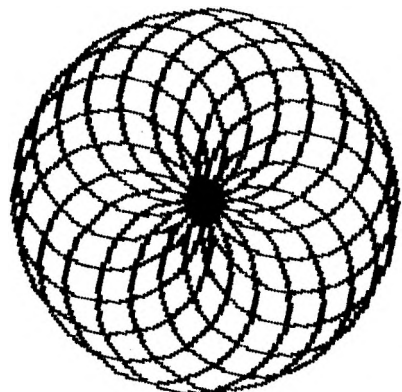
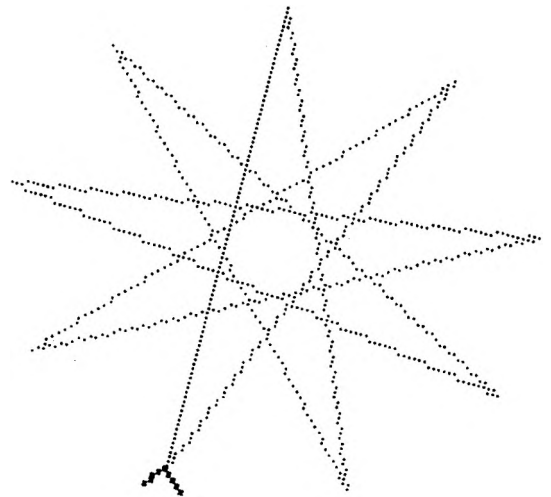
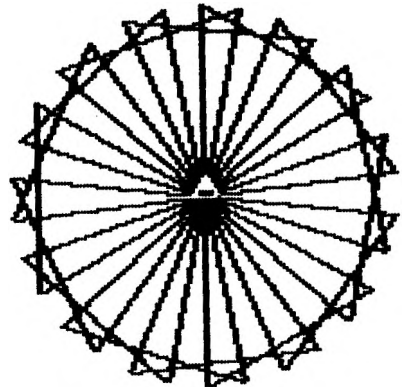
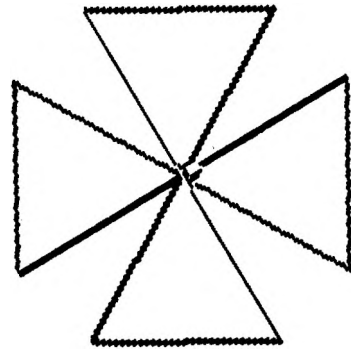
Deals fully with LOGO, and with American classroom usage.

### *Learning LOGO on the Apple II*

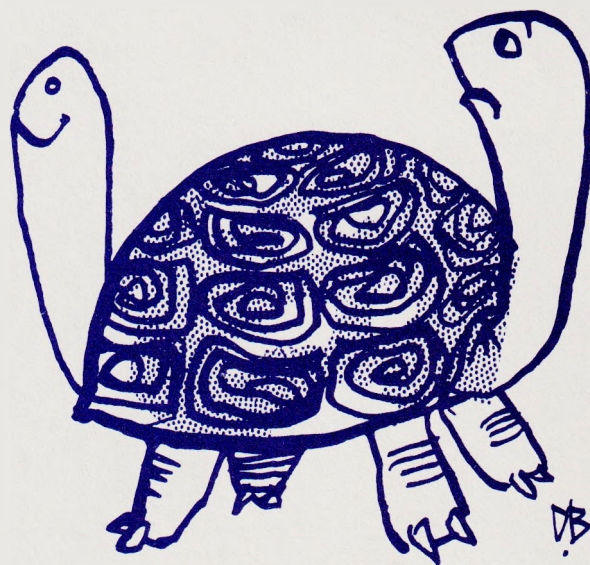
McDougall, A., Adams, T. and P. (Prentice Hall of Australia, Sydney, 1983)

About £11.00 (ISBN 0 7248 0732 2)

An excellent introduction.







**Jointly published by**

Ginn and Company Ltd  
Prebendal House  
Parson's Fee  
Aylesbury  
Bucks, HP20 2QZ

Heinemann Computers in Education  
22 Bedford Square  
London WC1B 3HH .

£1.95

ISBN 0 602 22692 9

ISSN 0264-3847