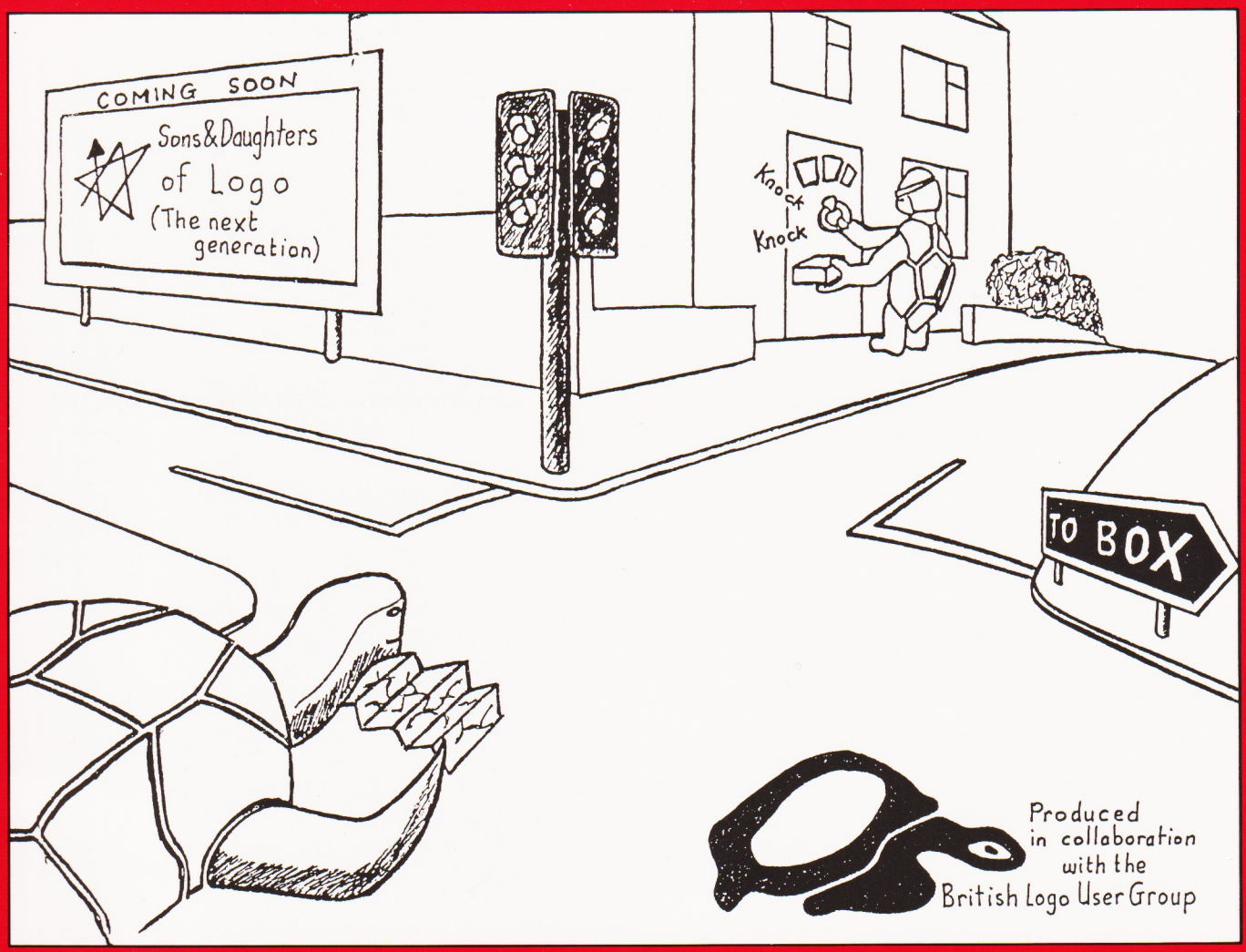
MICRO-SCOPE

Logo Special

► Autumn 1992



- ► Floor robots
- ► Screen turtles
- ► Plans and routes
- ► Logo and language
- ► Go adventuring
- ► Turtle pie charts
- ► Logo in control
- ► Where next?

NEWMAN COLLEGE with MAPE

Contents

Editors Chris Robinson and Senga Whiteman

ISSN 0264-3847

Correspondence to the Editor: *MICRO-SCOPE*, 99 Foxcote, Wokingham, Berkshire RG11 3PG
Tel: 0734 733718

MAPE (Micros And Primary Education) is open to individuals and institutions. The current subscription of £15.00 p.a. UK, £20.00 p.a. overseas, includes direct mailing of **MICRO-SCOPE**.
Application forms from: Mrs G Jones, 'The Old Vicarage', Skegby Road, Normanton on Trent, Notts NG23 6RR.

Published by Castlefield (Publishers) Ltd.

Individual copies from: Castlefield (Publishers) Ltd., Newton Close, Park Farm Industrial Estate, Wellingborough, Northants NN8 3UW. Tel: 0933 679677

MAPE reference for Income and Corporation Tax relief on membership fee: T1644/23/1986/MT
Charity Commission reference: OSP-292898-R Reg. No. 292898
MAPE is grateful for the support received from Acorn, Apple, Commodore, Cumana, Microvitec and Research Machines.

VAT Number: 544 8661 18

Produced by The Castlefield Press, Wellingborough.

MICRO-SCOPE
Logo Special

Welcome to Logo

Chris Robinson (BLUG and Deputy Head, Horndean Middle School, Hampshire)

This *MICRO-SCOPE Logo Special has been produced through the co-operation of MAPE and BLUG.*

What is special about Logo?

Probably Logo's greatest asset is its versatility. The potential of this educational computer programming language is explored at different levels within this magazine. Whether you teach reception or top juniors, whether you are a novice or an expert, whether you use a battered Beeb or a Nimbus network, whether you want help or ideas, whether you are after philosophical discussion or light entertainment, I hope you will find something inside for you.

Mike Doyle, the Chairman of the British Logo User Group, provides the introductory and valedictory articles, looking at the past and future of Logo, whilst the bits in between represent what is happening now all over the world.

There are articles about Logo's best-known feature, turtle graphics, using floor robots or screen representations, and what to do after you've drawn your square.

Next some language uses of Logo have been explored. Have the mathematicians dominated the computer scene for too long? Logo is a *language*. Perhaps now is the time to consider its potential in this direction? I personally became convinced of this potential when helping in a school's technology week introducing three girls to computer control. Only one of them spoke English and I spoke no Urdu or Gujarati. However, it wasn't long before the three of them were 'conversing' with the machine in words they understood!

Computer languages, however, are a mathematician's tool and other mathematical aspects of the language are further explored with some challenges to top junior teachers.

Finally, I have included an article about computer control – a potentially exciting application involving many diverse areas of the curriculum for which Logo is ideally suited.

What is BLUG?

The British Logo User Group is a mutual support group for educational professionals in Britain.

Its aim is to develop good practice in the educational use of the computer language, Logo.

As part of the Eurologo movement, BLUG publishes a journal, *Eurologos*, available free to members. The current issue is available for £5.00 to non-members from BLUG, PO Box 43, Houghton-on-the-Hill, Leicestershire LE7 9GX. Members' contributions are published in the newsletter, *Logos*.

Other occasional publications include National Curriculum support materials and joint publications with other organisations.

The latest mathematics National Curriculum document established Logo as the only appropriate language to use from the earliest primary years to University level.

Recent developments in hardware and software have enabled Logo to break out of its Turtle straightjacket. It is now possible to provide the environment envisaged by the originators of the language in which children can develop and test concepts through appropriate communication with computers.

The problem facing teachers is that their local advisory support has a changed function. This was reflected in exceedingly high demands made on BLUG. The group is now emphasising its role of a *mutual* support group.

BLUG has strong European links and has been involved in regular bi-annual European conferences since 1987. Athens is the venue for the 1993 conference; contact Chronis Kynigos, Dept. Informatics, University of Athens, 19 Klemenous Street, 10675 Athens.

Helplines

It is extremely satisfying to put the finishing touches to a Logo program you have just produced and sit back to enjoy the fruits of your labours. However, it can also be very frustrating when things do not work out as you wish. The following contacts are provided by BLUG to assist when all else fails.

You may write to me at 3 Cowdray Park, Hill Head, Fareham, Hampshire PO14 3SA. If I don't know the answer, I might know a man (or woman) who does. Or you may telephone Mike Doyle on 0756 794601.

N.B. Various versions of Logo (notably the RM implementation) on different machines may offer different features or work in a slightly different way. The listings provided in this magazine attempt to be 'LCSI standard' as far as possible. Where anything is machine specific, this has been detailed in the text.

Logo. What is it?

Mike Doyle
Chairman, BLUG and teacher of pupils with special needs in Bradford

Logo UK: a potted history

Those of us who, now greying, recall the early 1980s, will remember that Logo was hailed as a 'Good Thing'. Computer scientists at MIT had, in the 1970s, shown that even toddlers could be taught to program computers. At the time this was considered revolutionary. A debate began: should children be programmed by computers or should they program them themselves? Should teachers put children in front of the 'Toddlers' Tireless Table Tester', or should the children themselves write the program? This, of course, was an extreme view, but by providing a concrete object – the turtle – as a focus for children's language, the MIT team did demonstrate that young children could sensibly and profitably program a computer. The two most important Logo-related books were published in 1980: *Mindstorms* by Seymour Papert and *Turtle Geometry* by Harold Abelson and Andrea diSessa – fascinating, both. Put them together, however, and suddenly Logo = Turtle. Teachers focused, like young children, on the one comprehensible concrete object in the computer maze – the floor turtle.

With hindsight, it is not surprising that many in education came to believe that *Dart* was Logo. Nor should it come as a surprise that Logotron were unwilling to develop a Logo for the Acorn Archimedes, preferring to re-compile the now defunct Acornsoft Logo for the new machine. If we look at INSET material – the *Primary Logo Pack* from the Advisory Unit is an excellent example – we find that the fact that Logo is a list-processing language is completely ignored. UK Logo developments were, largely, focused on integrating turtle graphics with other activities. Background screens for turtling were produced in drawing packages. RM's Nimbus Gallery was used to export a simple turtle graphic drawing into *PaintSpa* for colouring in and thence to *Write-On* to write about it, or even (most exciting of all!) to *Art&Time* for animation. Logo's lowest point of degeneracy came with the emergence from Northampton LEA of *TinyLOGO*, a piece of software almost identical to *Crash* (in the original MEP Primary Pack) in which you could draw a right-angled triangle, the sides of which were all the same number of units long! Not quite what Papert had in mind when he talked of children controlling computers.

There was one aspect of Logo which offered escape from turtles – technical control. Unfortunately, some government apparatchick decided that teachers didn't really need Logo and probably wouldn't notice the difference anyway. So NCET published *Contact*. Like *Dart*, this looked like Logo, but didn't work like it.

Fortunately, the reaction of the toy trade has been somewhat more enlightened. Lego International, after a flirtation with their own software, went to the source of Logo and commissioned Technical Control Logo from LCSI, Seymour Papert's firm. Lego UK followed suit and adopted *Control Logo* as standard.

Logo: turtles and other topics

Logo is a computer language.
So, what is a computer; and what is language?

- A computer is a machine which extends our language capabilities. It is particularly good at following long and complicated sequences of instructions – ones which give people headaches.
- Language is a lexicon and a grammar, words and the rules for combining them (Chomsky).

Logo is no more an aspect of mathematics than it is of English. It is a language which people and computers can have in common. It is the means by which you and I can have a conversation with a computer. Although our conversation will be rather limited, it can be educationally very valid.

Turtle geometry

One recent and valuable topic of conversation is 'Turtle geometry'. This is a new geometry. Put simply, it is the geometry of giving directions to the nearest bus-stop: go along so far, then turn left . . . etc. Turtle geometry exercises can be carried out by children in PE, or playing with a floor robot, or instructing a screen turtle.

Turtle graphics

'Turtle graphics' is a computer-screen graphics microworld. Here the turtle can not only move in a turtle geometric manner but also within a map-like framework of coordinates and bearings. Good turtle graphics implementations let the turtle change shape, move fast and slow, fill areas with colour and shade them with pattern, and provide multiple turtles. For all these activities vocabulary is provided so that pictures can be created by writing instructions, not just by clicking and dragging. The screen takes us further in our language use than the floor turtle. We can now ask the turtle to tell us about itself. For instance, if we have told the turtle to **right 90** we can find out which way it is now looking by asking it to **show** us its **heading**.

For work in mathematics, the screen enables us to add coordinates – polar and Cartesian. To use the turtle for Cartesian coordinate geometry we use words such as **setx, sety, setpos, pos, distance**; for polar coordinates we have **setheading** and **heading** combined with **forward**. We may mix these three geometrical systems (Turtle, Cartesian and polar) freely in our Logo as we do in normal language. But, when teaching, we must be quite clear which we are using and why.

Logo: the language

'Logo' has far more topics of conversation than Turtle Graphics. One is described in detail elsewhere in this Special – 'Logo and the concept keyboard'. But the most widely-used extension in the UK is probably *Control Logo*: the vocabulary of **turnon, sense?, waituntil**, etc.

World-wide, the best-known extension is the text-oriented vocabulary of *LogoWriter*. Though little used in the UK, elsewhere in the world children are using Logo Writer to explore the difference between "computer" and "people" language. They do this by talking to the text they have written on the screen. What does this mean? Here's an example. If they write,

```
This draws a square:
repeat 4 [right 90 forward 50]
```

and **select** (highlight) the line "repeat....." then the Logo instruction **run selected** will draw a square. But, if they select "This draws ..." and enter the same instruction then Logo will reply, "I don't know how to ..."

Like all living languages, Logo is still developing. Soon we will be able to ask Logo to **create** objects on the screen, a graph for instance, which we will be able to talk to like the turtle.

Logo is not, under any circumstances, the property of mathematicians. It is a language which children can use to develop a concept which we oldies find alien – the fact that machines can understand language. Children need to learn that machines can do things with language that our brains find impossible.

Pip mats

Nick Clabburn
Bousfield School, Kensington & Chelsea

Pip mats

The children were initially asked to design individual Pip mats: sheets of A3 paper on which a route could be designed for Pip's travels. These were intended to act as introductory tasters to the project and to familiarise the children with Pip's commands. From this exercise I was able to see who was struggling with the command concepts. I was also able to inject an element of fun into the proceedings by encouraging the children to invent amusing titles and illustrations.

Much hilarity was invoked during demonstration time when children were asked to program Pip to travel on their mats. It was some time before they became familiar with writing a program and keying it in, instead of moving Pip in laborious individual stages. It was interesting to see that many children found it difficult to hold commands in their own memories and that the concept of programming procedures seemed alien to some of them at first.

At first the function of this exercise appeared limited, but gradually the children invented more complex procedures for Pip. One example was the challenge of making Pip turn as many circles as possible at a given symbol on the mat, or reversing into allocated spaces. The pupils were addressing mathematical, logistical, social (peer-group sharing and acknowledgement of other's achievements and ideas) and graphics-based problem-solving activities which might have been boring if approached in other ways.

At this point I felt the project had started well, with the children having great fun and learning quickly about the capabilities of Pip. I was determined to use Logo to a greater degree in order to consolidate the notion of building procedures. By now the children were keen to start making their larger games. I wish I had taken the time to laminate the mats as this would have made them more permanent; many Pip mats were used by other staff in classes throughout the school to demonstrate the functions and possibilities of Pip.

I am fortunate enough to have four non-English-speaking children in my class who work very hard and enjoy writing in their own languages. These children, with the aid of a support teacher, produced Pip mats and instructions for moving Pip in their mother tongues: Urdu, Japanese, Spanish and Farsi.

The children thoroughly enjoyed using each other's mats. The feedback on the project was positive and they were clearly delighted with the versatility of Pip. By now I felt the children were ready to work in groups to produce their group games and to work individually on their booklets. Many children felt the need to refer to *MicroSMILE – The First 31* to check angle estimations. Others (particularly those who had had extra experience with the Maths coordinator) chose to use Logo to plan their Pip procedures. This reinforced my desire to use Logo more. The children seemed to enjoy the logic involved, probably because much of my own teaching is of an intuitive and diverse nature.

Games

Having mastered the fundamentals of Pip control, the children worked constructively in groups to produce imaginative games requiring skilful programming. I was particularly pleased with the topical themes used, such as the Gulf War and the Docklands Development, and the diverse ways the games were constructed. It was necessary for reasons of space that only one game at a time could be constructed, since the children had shown a strong desire to make their games fairly large and to cover a considerable part of the available floor space.

Mission Impossible

The aim of this game was to rescue people from various catastrophes. The ground plan had three scenarios carefully chosen to be completely different: seaside (sea rescue), country (chased by bulls) and city (fire). The group built the game using paper, Lego bricks and felt pens. Points were gained for rescuing people and penalties incurred if a building or a policeman were knocked over. To get round the course, they had to use a centimetre ruler to measure accurately how many centimetres Pip could travel. One

Figure 1 *Pip. (Picture reproduced with permission of Swallow Systems.)*

player also needed to do a reverse run over the bull to rescue someone. Great accuracy was required to achieve this.

The whole class came in and were asked to estimate the distances. Unfortunately, all the angles on the board were 90° so I set the group the task of getting round the game without using right angles. They found that 89° worked perfectly well. I then informed them that they could not use any degrees in the 80s or 90s. They contemplated 45° and decided it would be quite wrong. It was fascinating to watch the children tackle problems and solve them by group discussion.

Maze

In this, Pip had to collect objects (mainly gold). Items also had to be bought, for example a sword to stick on to Pip to knock down the Troll in order to gain 400 gold coins. Pip itself was elaborately disguised with a paper costume.

Desert Storm

Here were mines and a treasure chest. If two mines were hit the player (Pip) went back to the beginning.

Pirates' Island

This game had two different routes, with obstacles and treasure. If Pip took one route outwards, it had to take another back.

All the games showed great flair and imagination. The children enjoyed playing them and modifying and refining the rules.

Books

The children were now ready to produce their handbooks about Pip. I made it clear that these should be produced as professionally as possible, so I demonstrated the school's ring-binding machine. I specified that all the books should measure 21 cm × 15 cm. I also stressed the need for a list of contents such as 'Pip Controls', 'Rules of Game' and 'Further Uses of Pip'. By this time the children appeared to be highly motivated and keen to start on their books.

We only had one computer in the classroom and many children wanted to work simultaneously on word processing programs such as *Caxton* and *Write*. Computers were borrowed from other classrooms, or children encouraged to travel to available machines. Some children

felt comfortable in producing booklets merely stating the functions of Pip; others wanted to give fuller explanations and then the *Caxton* format made the text too large for the size of the books. The photocopier was used to reduce some of this *Caxton* work to the appropriate size: a relevant link here with maths work on percentages. I found it difficult to ascertain the accuracy of the text in other languages, so parents were asked to check.

Some children were very slow in developing word-processing skills, and they were allowed extra time at breaks to improve their techniques. Some children still insisted on writing their books by hand. I felt I could not entirely discourage this, since the children clearly took great pride in their work and were fulfilling many of the criteria set for the production of the books. I would, however, expect the children to provide evidence of word-processing skills at a later date. The majority of the class clearly developed their word-processing skills. I was particularly impressed by the hard work and attention to detail shown by the children in producing the books. Great attention was shown to graphic details and care was taken to use appropriate word-processing packages. Some children were sufficiently motivated to use desk-top publishing systems outside school which provided a good constrast to those usually available to us.

A strict deadline was set for completion which I felt reflected the realities of industry. Only a minority failed to complete their books in the set time of four school weeks.

Afterwards

Having completed the Pip mats, the games and the books, the children presented a class assembly to the whole school, inspired by the theme of control technology, and using dance and poetry. It was most rewarding to see the books being put to use by children from other classes.

I was particularly pleased with the creative elements of the project. I feel that the children have successfully managed to merge IT with creativity. They realised and fulfilled Pip's potential, and utilised much associated software and hardware.

Editor's postscript: Rubberised play mats for use with Pip are now available from Swallow Systems, 32 High Street, High Wycombe, Bucks HP11 2AQ; Tel. 0494 813471.

This article has been reprinted from 'Share IT', with the permission of ILECC.

A turtle case study

Theresa Mungall
Senior Teacher in Learning Support, Tayside Regional Council

It was a P5 class of 28 children who were involved in the piloting of the Roamer.

At the time of the pilot the ongoing work of the class did not lend itself to the integration of Roamer activities and although the class teacher was later to regret this, she did not feel justified at the time in altering her plans. However, many of the activities undertaken were relevant either to work covered eg right angles, or general class work eg writing poems, craft etc. The activities with the Roamer were therefore not integrated thematically, but where more possibility for generating ideas and seeing potential may have occurred, the activities were still in the context of the children's normal learning.

Activities

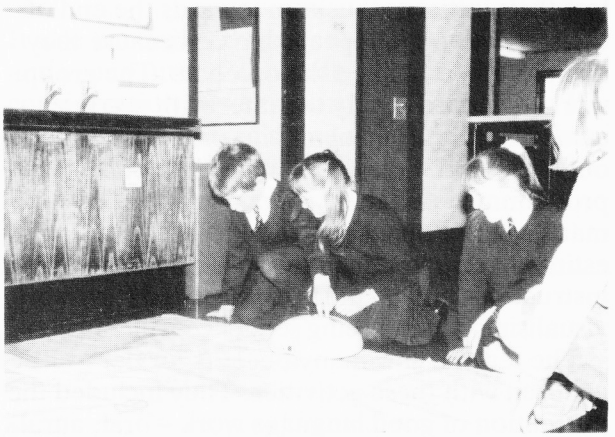
The children made 'costumes' for the Roamer which made its tasks more meaningful, eg it was made into a police car, had to complete various routes, had to do so in given time, had to use its siren etc. The children were sometimes asked to record their procedures, complete a piece of related writing or make diagramatical representations. The children made poems about Roamer and printed them out on the computer. They investigated shape and, by insertion of a pen through the centre of the Roamer, they could make outlines of the shapes on large pieces of paper. They then recorded the procedures they made to complete each of the shapes. The completed procedures were built up from the formation of

Figure 1 *The 'jacket' to accompany the police siren.*

Figure 2 *The Roamer with its various 'jackets'.*

Figure 3 *The control panel is still accessible.*

Figure 4 *The children discover an error in their procedure.*



Figures 5 & 6 *A group of children working out the procedure that will program the Roamer to navigate the roundabout.*

sub-procedures which resulted from their experimenting with the amount of turning involved. The REPEAT key was introduced when the children saw the significance of this.

Some of the children posed their own problems, eg making a roundabout for a car to drive around. The Roamer allows the planning to take part in stages. They then worked out the path he was to follow, recording the small manoeuvres as the task progressed until the whole manoeuvre was built up. This was a group activity with each child doing different tasks in turn and it involved much interaction. The teacher thought it developed skills of working cooperatively, listening to others, forming hypotheses and it fostered attitudes of consideration for others, persistence and patience. When asked about their problem, the children

said they had 'to work together', 'everybody had different ideas', 'we had to sort out the best ideas' and 'it took a long time'. When asked, they said they enjoyed working it out, and that it was 'better than school work', 'something different', and that 'we have the only one in Arbroath'. They also added that they liked it better when they could choose their own problems and the teacher commented that they usually came up with better ideas than she did, and to date had coped with completing their own assigned tasks. The children then demonstrated the outcome of their planning and discovered a flaw in their original procedures. This was to be remedied and the children were observed on task. It took 55 minutes to complete and out of the five children involved only two of the girls lost their initial enthusiasm towards the end of the activity and it appeared to be because they had already predicted the outcome. The group worked very naturally together with no-one assuming total control and no-one being left out or behind. Some recorded while one programmed and they all contributed ideas and made suggestions. They showed ability at estimating, predicting outcomes, sequencing instructions and comments suggested they were visualising before seeing sequences run through.

The teacher saw many benefits for her children with these activities. They included the promotion of good language work – oral, aural and written, together with the investigation of mathematical concepts and the opportunities for developing cooperative attitudes towards problem-solving activities. She felt she was now in a position to better appreciate the potential of the Roamer in other areas of the curriculum.

The children tended to be in ability groups for the activities attempted as she felt that the more able children left the less able behind, but she could imagine that with integration into other areas of work, mixed ability groups would be appropriate at times when different skills were involved.

The less able children made worthwhile use of the Roamer although they used less complicated procedures or took more steps to solve their problems. One child with behavioural difficulties who had caused problems in the class at times had valuable experiences, both learning and social. He was able to demonstrate two procedures he had worked out and was obviously proud of these. The teacher recounted an incident that occurred during his first experience of this type of investigational group activity with the Roamer. The boy's correct prediction was rejected by the other members and this resulted in disruption. However the matter seemed to have been resolved by sensitive intervention by the teacher and work resumed. The teacher thinks the boy learned a valuable lesson from this experience. He has discovered the need to wait, appreciates better the need to rethink and talk over situations, and she believes that reinforcement would be achieved with more of this type of work which may have a long-term beneficial effect on his ability to interact socially in a way that is acceptable to her and the other children.

The class were enthusiastic about the work they had undertaken with Roamer and it would appear that even in this short trial they have gained from it. Perhaps more importantly this activity has gone some way to demonstrate the potential it has to enhance and extend the ongoing work of this class.

During the course of the trial a P2 teacher became interested in the possibilities of the use of Roamer with her current Environmental Studies theme about the Post Office. After consultation with the P7 teacher it was decided that certain P7 boys could be asked to make a game involving Roamer for the P2 children to use.

Two boys from the P7 class who had been identified as experiencing difficulties in some areas of the curriculum were selected and in this case were given the problem and the design brief. Roamer was to be used in a game and dressed up as Postman Pat, with the task of delivering letters to houses in the High Street. The two conditions were that it was to be easy enough for the children to cope with and that it should be interesting. From the boys' reporting it was discovered that ideas for the street plan were from a map devised by the children themselves. They then proceeded to reproduce this on a large plastic sheet, made the houses from cardboard boxes, and lastly created the jacket which was to cover Roamer, while allowing the function keys to be easily accessible. The rules of the game were decided after much trial and error and experimentation with the Roamer. It was decided that the game should start at the Post Office. Cards for delivery were to be collected by the player and they were then to have five minutes to deliver as many cards as possible. (No major traffic offences were permitted!) Their next task was to teach the P2s how to operate the game. This was done with an initial group and was to have a 'cascade' effect.

The boys thought that the children 'took to it' easily and that they understood it better than they had anticipated because 'they were having fun'. They also said that they let them do what they wanted so that they 'could find out what they can'. The biggest problem was the number

Figures 7 & 8 *'Postman Pat' game devised by two P7 boys, to be used by the P2 class while doing their project on 'the Post Office'.*

of degrees involved in turning and the boys thought it best just to tell the children what to do to make the correct angle of turn.

When asked if their project was a success, the boys said, 'P2 thought it was great.'

This activity involved thinking logically to solve problems by all the children involved. It also encouraged the development of paired and group interaction skills, and fostered consideration and empathy for others in the P7 boys. It gave them a sense of achievement that may at times have been lacking in the normal course of their school work.

Floor robots in a primary school

Chris Taylor
School of Education, Exeter University

'Wonderful, fantastic, it's so robust,' – these are comments about Pip from the Headteacher of a Somerset primary school. Pip had been loaned to the school for a period of six weeks for evaluation prior to an in-service course I was due to be running at the University. The school already had a Roamer (which was being used primarily by the older juniors), and had jumped at the chance of the use of Pip. It provided an excellent chance for me to evaluate the two machines side by side in the classroom and to determine to what extent they were comparable.

Pip was shared between the middle and top infants and the lower junior class. Minimal instruction was given to the children as to how to use it; they were largely left to get on. A fully integrated plan of work had not been drawn up; it was more a question of giving the tool to the children in order to see what they could do with it. It was quickly apparent that they could cope with the angles and distances, even though Pip has to be programmed in degrees and centimetres; this did not prove a great problem to the children. There is a plug which multiplies all the amounts by 10, but this was not used. The children quickly understood the concept of taking Pip for a walk, and giving it sets of instructions to complete a journey. Indeed they soon developed the strategy of using Pip to measure and determine where a path should go, and then creating a story around the journey, rather than just setting up a path and trying to drive Pip along it. This was undertaken by a group of children: one operating Pip, one recording the instructions, and one setting up the path, placing obstacles etc.

The children quickly mastered how to control Pip; they worked out ways of using it which gave them opportunities to measure and estimate distances and amounts of turn. They set themselves problems to be solved and then worked out strategies for solving them, and when they had completed their mastery of the machine, they taught their teacher to use it!

From these observations, what do floor robots have to offer in educational terms? They appear to act as a concrete environment through which children can develop mathematical concepts of distance, direction, more and less and approximation. They provide a stimulus for the development of problem-solving skills and strategy development. They act as a medium for discussion, collaboration and pooling of ideas. They provide children with a concrete means of explaining abstract ideas. They also act as a means of empowerment, in that children can take control of aspects of their working environment and through this control develop self confidence and self esteem. As such, floor robots have a great deal to offer the child with special educational needs in terms of providing a concrete medium to think with.

How do Pip and Roamer compare? The most obvious comparison is of cost: Pip costs almost three times as much as Roamer (without additional accessories). However, when the cost of rechargeable batteries and a pen kit have been added to Roamer the cost differential is less significant. Pip is certainly very robust. It has been trodden on and dropped without coming to any harm. Roamer feels less robust, and so I haven't had the courage to give it a full robustness test. Pip's rechargeable batteries are very effective; in full time use in the school it was found that one charging would last for three days (although the manufacturer prefers daily charging). The lantern batteries for Roamer cost about £6.00 a pair and were not totally reliable, although rechargeables are available as extras. Pip is much more straight forward to use; its controls are easier to operate and instruction sequences are simpler. Roamer offers more features: the ability to redefine distances and angles, control of length of note as well as pitch, and the ability to add on a range of control devices. Both firms can supply computer interface kits so that procedures can be saved and reloaded. The round shape of Roamer is attractive and cuddly, but the rectangular box of Pip gives a clear indication of front and back; it also cries out to be decorated and given character. Pressing a key stops Pip in the middle of a procedure; with Roamer it has to be switched off, losing any work that has been done.

In my opinion, these two machines both have a place in the primary classroom as they address slightly different markets. Pip seems more suited for use with younger children, with children with special needs, and in an environment where maltreatment is likely. Roamer offers more sophistication, and closer links with the procedural thinking of Logo. Ideally, one might use Pip as an introduction, develop further work through Roamer, and then move on to a wider variety of means of control such as Lego buggies and control technology. They both offer a means of using IT to address some essential educational aims without needing the expense or complexity of a full computer workstation.

Using Pip in a special school

I was subsequently able to lend a Pip belonging to the School of Education to a local special school for a fortnight, so they could try it out with their children and evaluate it. The children concerned were eight years old, and have a variety of physical disabilities, mainly caused by some form of cerebral palsy. In addition to their physical disabilities, learning difficulties were also present to some degree.

The questions I wanted to ask were:

'Can these children physically manage to use these machines?' and

'Are these machines of educational use to these children?'

Pip was used by them on their project afternoons, and the teacher concerned gave them structured tasks to do in order to cover various educational concepts. Physically, they were able to use Pip, although to help them, a key-guard was made from a piece of plastic with holes cut in it, blanking off some of the keys. This key-guard was fixed on top with adhesive tape. Most of the children then found pressing the keys to be fairly easy, although one child, who had trouble in bending his fingers, found it rather difficult. Another child used a pointer attached to his head to press the keys; this seemed very successful. One problem was that the keys auto-repeated if left depressed, so sometimes a number such as 9 became 99 or 999, without the child knowing it had happened, the only indication being a repeated beep, and a rather strange response in the program.

Pip was used to trundle up and down the table (with estimates of distance being undertaken), to turn round in circles (starting work on angles), and to go from one child to another. The repeated key presses became something of a problem here, as there was no visual display of the instructions given. However, some useful work on estimating distances was undertaken, and amounts of turn, debugging of programs and discussion of what instructions they wanted to give Pip.

Was the exercise successful? The children were well able to control Pip, although the precision and accuracy of their instructions were

substantially poorer than might be expected from able-bodied children. They were, however, able to be involved in some useful mathematical work and discussion. Some of this must be related to lack of spatial awareness and experience, some from their inability to give precise commands, and to see those commands. Given more time, development work could have taken place on the key-guard, a proper board with a lipped edge could have been made to stop Pip falling off the table, and Pip could have been decorated to give it some personality. I will also suggest to the manufacturers that an option of being able to deactivate the auto-repeat is made available. One aspect of surprise to me (with a background in mainstream education), was how long it took to get the children ready to work, by strapping on splints, placing them securely at the table and by preparing their workplace. This was unavoidable, and it was noticeable how much physical work the teacher had to do, lifting and moving them. Not a job for someone with a weak back!

The class teacher seemed very pleased with how the children had got on with their robot. Further research will be needed to evaluate just how far such devices can be used to develop mathematical thinking and spatial awareness, and whether they can help children to make up for their shortcomings in mathematical experience due to a lack of spatial mobility.

An evaluation of floor robots in special needs

Lynda Duthie
Primary Support Teacher for Maths, Somerset

This report is from a case study of two special needs boys, aged 10 and 11, as they are introduced to Logo through the use of Roamer and Pip, two programmable robotic toys, and as they attempt their first 'task' unaided and unsupervised.

There is little doubt that the boys enjoyed working with both Roamer and Pip. They remained 'on task' for considerable lengths of time when working with me and, more significantly, when they were left on their own. Considering the fact that both had been described as having little motivation or concentration, this was remarkable. The teacher's reports state that this was particularly true of Derek who 'generally tends to flit from activity to activity'.

Session 4 required the boys to send Pip to visit five different toys; after an hour of trying on their own on a Friday afternoon they were still unable to send Pip to visit one toy. I quickly attributed this to my mismatch of task to ability and attempted to redeem the situation by praising them for their persistence.

On watching the video recording of them working, it became clear that Derek had persevered throughout and this had kept Len going. The class teacher feels sure that Len would not have sustained his interest for anything like so long if he had been on his own.

For whatever reasons, it is true to say that of the two, Derek was more eager and more persistent in trying things out mentally, whereas Len preferred to trust the paper and pencil method which he could manage quite satisfactorily, eg when working out what to press to make the Roamer do two complete turns. . . . both knew to double 360° but Derek came up with 730° mentally while Len did a careful sum on paper and came up with 720°. I let them try these out and Derek was able to recognise and correct his error with no prompting from me. He was easily able to work out mentally that a half turn was 180°.

Len, in particular, appeared to have difficulty in remembering the order in which he had pressed in numbers only seconds after he had done so and in spite of calling them out himself as he pressed them in.

On separate occasions when they were trying to add mentally they both confused the order of digits in their heads which might indicate a lack of understanding of place value. However, I suspect it is more to do with a left/right confusion which is, in fact, a subskill of spacial orientation/space organisation (Sharma's second pre-skill). I noticed that the wrong arrow key was often pressed when responding to a called-out instruction.

Estimation was another pre-skill they were not able to show evidence of here, preferring to rely on their ability to use an established method of working out. When they were pushed (by me) into having a sensible guess Len especially was reluctant to commit himself, eg:

> 'Six, nine . . . three, no, four, no, six hundred and ninety-four, no six, six hundred and ninety-six.'

Rounding up or down didn't happen even when I gave examples. After three sessions with me looking at the number of degrees in a full turn, Len's suggestion was:

> 'Three hundred and eighty-seven, no, seventy-eight, no, three hundred and fifty-seven. Let's try that.'

Even as we did, he adjusted his amount by one each time so we tried 357, 358, 359, 360 . . . at which point I congratulated him and asked why he thought it would be a complicated number with several different digits. He just shrugged. Perhaps his expectation is because all numbers are difficult to remember.

The boys found estimating distance with Pip was more difficult than with Roamer. In fact, the introduction of Pip in Session 3 confused them. Having realised that the units of distance were different, ie 30cm for Roamer and 1cm for Pip, estimations with Pip were less accurate. Although the teacher's view is that the boys tend to have difficulty adapting to change, the large numbers required to program Pip possibly added to the complexity of the problem. After Session 4 they told me that they preferred Roamer 'because he's easier and moves faster'. A race proved they were right about speed. Interestingly, a month later when reflecting, they said they preferred Pip in spite of Roamer being 'a bit posh'! The teacher reported that Derek had now realised the need for standard units, a leap which he believed the transfer from Roamer to Pip had contributed towards.

It was evident that both Roamer and Pip held appeal in themselves but the practical nature of the activity also seemed to help keep Len and Derek involved. On each occasion when left on their own they would start off collaborately but after about 20 minutes this would become more a case of taking turns. However, the class teacher felt that this was longer than they would usually maintain an interest when working with someone else. The boys said they would have liked a Roamer each and it would be interesting to see how they would work a) on their own and b) together on the same task but with a Roamer each.

The nature of their talk followed the same pattern. Initially they would make, and listen to each other's, suggestions but soon they would become quieter and less interactive with each other. They rarely asked questions of one another. However, just because they weren't obviously engaged in meaningful discussion didn't mean that they weren't thinking about what they were doing. Intervention from me sometimes served to spark off interaction between them and careful questioning could help break the task down into more manageable stages. They admitted to liking being asked questions 'because it makes you think more.'

I managed to get the boys to record their programs but they needed help and, although Derek did try, were not able to do so usefully when I wasn't present. This may have had more to do with the fact that they were slow and unconfident writers than with failure to see that this might help them to remember what they had done and be able to make simple amendments.

Without doubt one of the key advantages with Roamer/Pip is the ease with which simple programs can be tried out and amended or extended in the 'concrete'. That being the case, the boys' amendments often backfired because they forgot what they were amending.

I do see there being a strong link between the systematic approach of Logo and the development of sequencing ability, the crucial pre-skill in which both Derek and Len are weak. I would recommend that they have regular exploratory and task-set time with Roamer, sometimes with the teacher and sometimes on their own. They will need encouragement and help in recording and revising their programs. Sufficient time should be allowed to build on the development of their attention span, ie sessions of half an hour to an hour in length.

Further collaborative work should be arranged with regular feedback sessions where the boys can verbally explain, sometimes to the teacher, sometimes to other children, what they have done, why they did it, and what happened. This, of course, is good practice with children of any ability, and added emphasis has come about through the National Curriculum for Mathematics AT1, where communicating about mathematics is a developing strand.

Logo – a personal perspective

Mick Harwood
Deputy Headteacher, Blakenhale Junior School, Birmingham

I'm not much of a writer; I'm more of a doer. So, when asked to write this article (in fact I *volunteered*), I agreed on the understanding that it was a personal viewpoint – so here goes.

I'm not going to mention the National Curriculum once . . . oops, too late, I already have. OK, I'm not going to mention the . . . 'it', again.

There are lots of ways that Logo fits into 'it' – how, why, and where you will have to work out and decide for yourself.

I've been a Logo enthusiast since the early days of Logo and I've always been interested in turtle graphics. I've always used Logo on Research Machines equipment, from the original cassette versions on the 480Z, to the current full Logo implementation on the RM Nimbus.

I was excited about the way I could get shapes, patterns and colours on a screen with very limited programming knowledge. However, I always experienced difficulties and disappointments in getting my enthusiasm across to children (except again for the real Logo enthusiasts). So every year I would go through the routines of drawing and building squares, triangles, triangles on top of squares and call them houses, patterns with squares, patterns with triangles, circles, patterns with circles, etc. All well and good, but it never seemed to get any further.

The children were always dealing in abstract mathematics, squares and other shapes on blank screens. Whether it was the children's enthusiasm or mine that waned is difficult to say, but I got fed up with squares, triangles and patterns, etc.

Then I attended a Saturday morning West Midlands MAPE session at Newman College. (I thought I'd just slip that advertising plug in!) Two teachers from Staffordshire were demonstrating what they'd done with Logo and *PaintSpa*[1] pictures and *Gallery*[1] (an extension to Logo) in their Island Project.[2] *Gallery* allows *PaintSpa* pictures to be imported into and exported from Logo.

Instead of directing the turtle around an obstacle course of squares, triangles and circles, the children now had to guide the turtle (now in the shape of a galleon) between and around a lovely *PaintSpa* picture of some islands. Instead of squares and triangles, the children were asked to build a house on a *PaintSpa* picture of an actual desert island. Instead of octagons, they were asked to direct the turtle (now in the shape of a spider) and draw a spider's web between two trees.

I was hooked and re-enthused. Not only could I see exciting possibilities for children, I could see exciting possibilities for me.

I've always liked programming and messing about with procedures and I couldn't wait to have a go at some scenarios similar to the Staffordshire Island Project.

My only minor disappointment with their project was that because the galleon was drawn from a side view, it looked fine as long as it was travelling from left to right across the screen. It did look rather strange though in other orientations, especially travelling right to left when it then became upside down.

I wondered how I could improve on this. (Typical teacher – we always think we know best and can do better!) This problem could be overcome by choosing scenarios with views always looking down from above, ie in plan.

I decided that my first venture would be racing cars and Grand Prix circuits, and so *Logo Grand Prix 3 & 4* was written. The '3 & 4' arose from its suitability for upper juniors (years 3 & 4). I suppose it should now be called *Logo Grand Prix 5 & 6*.

It was necessary to draw pictures (using *PaintSpa*) in plan of a number of Grand Prix racing circuits, and then write the appropriate start-up file for Logo. It worked, and the children (and teachers) loved it. It was so easy to use that even an adult could use the program.

The children worked in pairs to guide the turtle (now redefined as a racing car) around a number of Grand Prix tracks. Initially they guided the car around the circuits with FORWARDS, LEFT and RIGHT commands. As cars don't really move forward, stop, then turn on the spot, and then move forward again, I introduced the children to the ARCL and ARCR commands which draw arcs of varying sizes and angle to the left and right. The children liked it even more – it was more realistic.

I then showed the children how to string all of their commands together, by writing down a few on paper, and then putting them into a procedure, seeing if that bit worked, adding a few more instructions, writing them on paper etc and then editing the procedure to tag these new instructions onto the end. Eventually the children had a string of instructions which made a racing track appear on the screen, and then made a racing car work its way around the circuit, simply by typing in one word, whatever word the children chose to call their procedure.

They had used Logo to write their own program. I couldn't keep the children off the computer now, not that I wanted to.

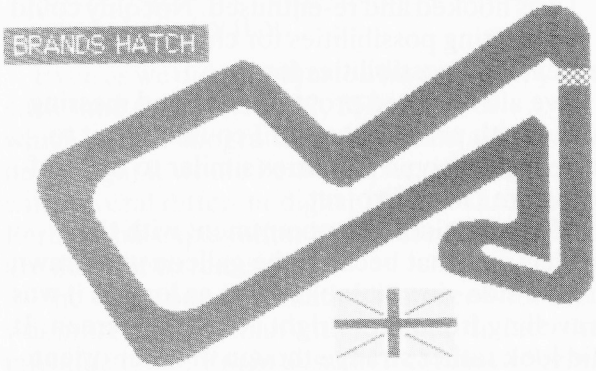


Figure 1 *The Brands Hatch circuit.*

As an idea of how 'into it' the children were, look at part of two children's procedure to guide their car around the Brands Hatch circuit. (Italics are explanations and not part of the actual procedure):

```
to race
ctx  (clears the text area of the screen)
brands  (loads in a picture file. See Fig. 1)
say [Drivers Ginever & Osborne on the grid.]
say [On your marks get set . . . ]
repeat 10 [ht type [brmmm. . / ] st fd 3 bk 3]
(makes the racing car judder as if on the
starting line raring to go)
pause 1
ctx say [The race is on!]
fd 100
arcr 7.5 95
rt 10
fd 50
arcr 5 180
lt 15
.
.
.
.
arcr 10 100
fd 10
rt 30
fd 100
say [Congratulations to the W I N N E R !]
```

You will notice one command – ARCR 7.5 95 – which tells the turtle to draw an arc to the right with a step size of 7.5 Logo units and through 95°. I cannot think of many reasons why primary children would need to use measurement of size and degree with such accuracy simply because they want to get it right.

They even had a commentary of the race as it proceeded, along the lines of: 'It's a long sweep to the left followed by a tight right-hander.' – all done simply by using the commands CTX to clear the text screen, and SAY followed by whatever screen comment was required, enclosed in square brackets.

Teachers as well as children started to demand more Logo files like the *Logo Grand Prix 3 & 4*.

Teachers of years 1 and 2 asked what their classes (current Years 3 and 4 children) could do using Logo. I turned the question back on them and asked what they thought they could do. One teacher responded by saying that he would draw some racing tracks suitable for first and second year juniors.

He used *PaintSpa* to draw a number of simpler racing tracks, a square, a rectangle, a triangle, an oval, and a figure-of-eight, that he thought were suitable for younger children. I wrote the start-up file for Logo, and designed a slightly larger car for the turtle to adopt, and *Logo Grand Prix 1 & 2* was born. Now all four years were Logoing, but still wanted more.

I am always conscious as a class teacher of the use of the computer as a social tool. I like children to work in pairs or small groups at the computer, to interact with each other and the machine. I also believe in competition as a stimulus for children. So I decided to write an enhanced version of the Grand Prix Logos, which involved three children, two drivers and a race referee. Hence the development of Two-Car Grand Prix.

It also got the children used to dealing with more than one turtle at a time, and having to address each of the two turtles in turn using the Logo commands TELL 1 or TELL 2. It is worth mentioning at this time that RM Logo supports up to eight turtles on screen at the same time in the original version, and up to 16 turtles in the Staffordshire version.

I later decided to cash in on the children's enthusiasm for Logo, and their enthusiasm for the then current Mutant Hero Turtles trend, by asking some fourth-year children to design a title

screen and four work screens for a *Mutant Hero Turtle Graphics* version of Logo.

This is the title screen:

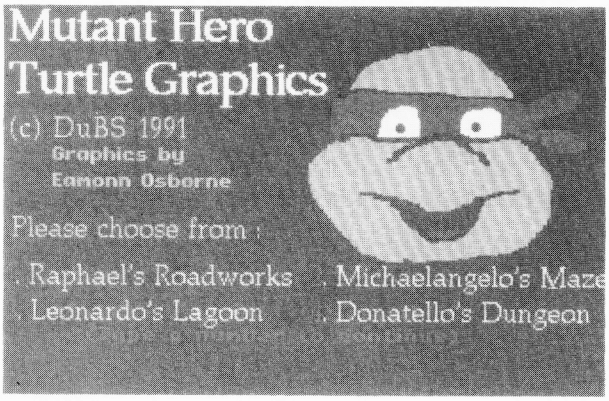


Figure 2 *The title screen.*

and this is one of the work screens for Michaelangelo's Maze, after a little bit of tidying up by myself.

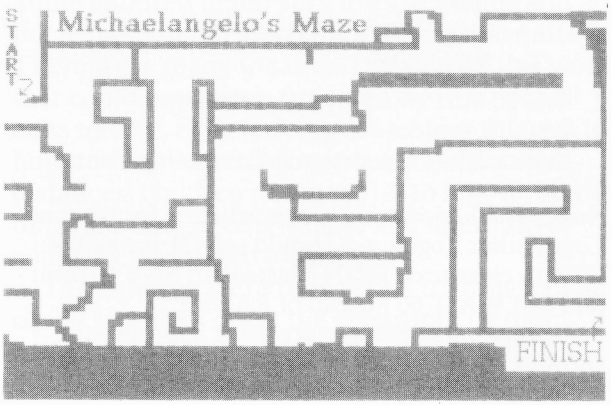


Figure 3 *Work screen for Michaelangelo's Maze.*

Further developments include a *Logo Litter Flying Squad*, which involves the children flying, in a helicopter, over the area around my school, and placing litter bins or bottle banks in positions predetermined by a **T**actical **E**nvironmental **A**ssistant **C**o-ordinating **H**elp in **E**ach **R**egion – or **TEACHER**.

It is the **TEACHER**'s role to mark circles on a briefing sheet similar to Figure 4, where they or the children feel that litter bins or bottle banks would be best placed. This means that each time a group of students uses the program it should be different.

This particular Logo program involves children using bearings and headings, and so it is probably best used by Year 6 children. Try it and decide for yourself.

The next development was for a West Midlands MAPE morning when a copy of pictures and the start-up file were given out

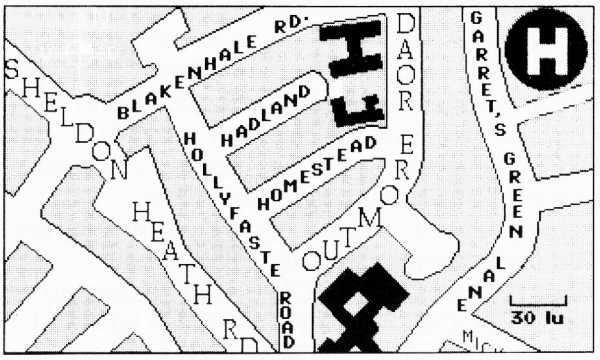


Figure 4 *Briefing sheet for* Logo Litter Flying Squad.

free to course members at a MAPE Christmas session.

The program is simply called *Christmas Logo*, and it involves the children guiding Santa's helicopter over rooftops to drop sacks of presents down chimneys. Again the program uses bearings, but after suggestions from teachers at my own school, who felt that many Year 5 children would not be able to cope with these, the facility to guide the helicopter simply by using NORTH, EAST, SOUTH and WEST was added. This now means that more children have access at a simpler level, but it can still be used again in Year 6, this time using bearings.

And so the rolling program continues, the most recent addition being a *SPACE Logo*. There are two versions, ORDINARY language, and SPACE language.

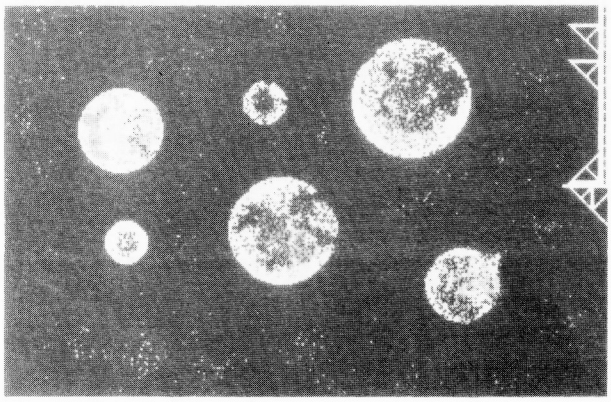


Figure 5 *SPACE Logo.*

The user has to guide a rocket from its launch pad on the right of the screen, around planets (which ones and how many times is decided by the teacher), and then back on to the launch pad (if the teacher so decides).

There is no physical check within Logo itself that the operations in this or any other program are actually carried out. I am not trying to do away with the teacher. In fact I am trying to involve her/him as much as possible, by getting

her/him involved in the planning and target-setting stages.

The children can guide the rocket with simple FORWARD, BACKWARDS, LEFT and RIGHT commands, or they can use ARCL and ARCR, which are more appropriate commands for a continually-moving space vehicle.

Using SPACE-language, the children may use even more appropriate language such as ORBITL instead of ARCL, ORBITR instead of ARCR, BOOST instead of FORWARD, and RETRO instead of BACKWARDS. They can also change the colour of the TRAIL left on screen. This version is probably my favourite. It is simple and realistic in terms of use and language used. It also looks more realistic, because some of the other turtles have been turned into sprites, and redefined as space rocks and debris which continually move across the screen, to give the whole program a more realistic feel.

I no longer have the problem of how to get children and teachers using Logo. This has been replaced by two other (much more pleasant) problems:

1. Restricting certain Logo programs to certain years to ensure continuity and progression.
2. Are there any more Logo microworlds that children and teachers in my own school can explore? Will I write some more?

Currently available microworlds (which ALL need *Gallery*, the Logo extension from SPA (Software Production Associates) are:

Grand Prix 1 & 2
Grand Prix 2 & 3
Two-Car Grand Prix 1 & 2
Two-Car Grand Prix 3 & 4
Mutant Hero Turtle Graphics
Litter Flying Squad
Christmas Logo
Space Logo

If you want to use any of the above-mentioned files, you will need a disc with a copy of your version of RM Logo, and *Gallery*, and then copy my files onto your disc.

Further details and copies of start-up files and pictures can be obtained from:

Mick Harwood,
Flat 3 Baxter Court, 96 School Road,
Moseley, Birmingham B13 9TP
Tel. 021 449 8224

There will be a small charge of £2.00 per title for the disc, limited documentation, and p&p.

References

1. *PaintSpa* and *Gallery* are both available from:
   Software Production Associates,
   P.O. Box 59,
   Leamington Spa CV31 3PF
   Tel. 0926 22959
2. The Island Project by Beryl Slade:
   ESG/IT Project 1989
   Staffordshire Educational Computing Centre.

Note: The Logo listing in this article is for Nimbus Logo; other Logo users should use CT instead of CTX to clear text, PRINT instead of SAY to print text and WAIT 60 instead of PAUSE 1.

A prepared screen picture using appropriate software may be loaded using LOADPIC "title.

This procedure will be needed:

```
TO ARCR :units :turn
REPEAT :turn [FD :units RT 1]
END
```

Plans and routes: working towards Logo with Year 1 children

Betty Lumley
St Helen's College, Hillingdon

The children who experienced this work had already spent one year in full-time education of rather a formal nature; their average age was 5 years 8 months. This project was pursued during one term for half an hour per week following a movement lesson. Although there was no feed-back during the week because this was not my class, the children were always keen to work in their 'Tuesday Book' and some continuity was maintained.

The reading scheme the children use is *1, 2, 3 and Away*, so they were all familiar with the map of 'The Village With Three Corners' on the back of some of the books. This proved a good lead-in

to the first task – to draw a map with a pond and roads connecting a church, a school and a house.

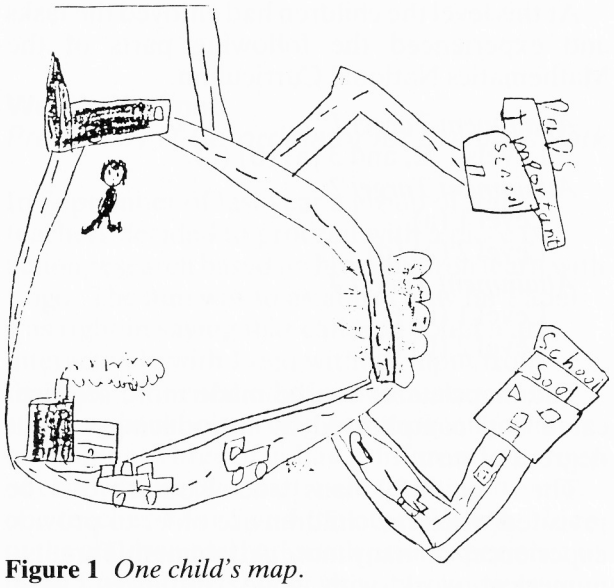


Figure 1 *One child's map.*

Some children made very intricate maps and all put in more detail than I had originally required. There were many ideas and aspects of this work that could have been followed up in a normal class setting, eg describing routes from church to house, discussion of directions, comparative distances, children's own routes to school, study of local estate agents' maps, and alternative routes between two locations.

The children were then asked to draw a plan of the teacher's desk, the classroom or their bedroom. We discussed the need to have a 'bird's eye view', and some children then noticed the discrepancies in the map of *The Village With Three Corners*, but decided that it was a picture rather than a plan.

Our next session was devoted to discussion of straight line routes, both around the school, and from their bedroom to the kitchen at home.

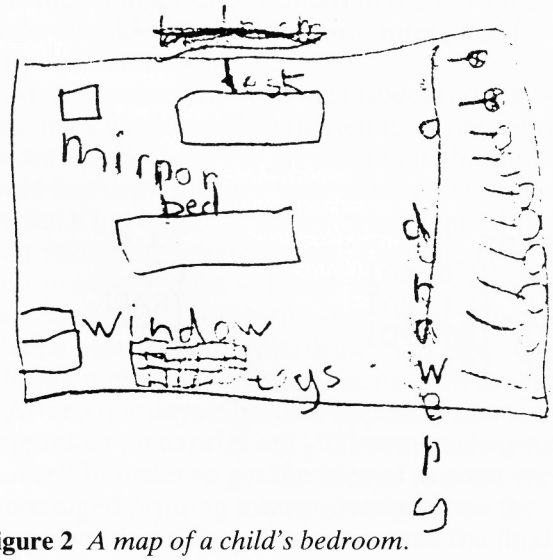


Figure 2 *A map of a child's bedroom.*

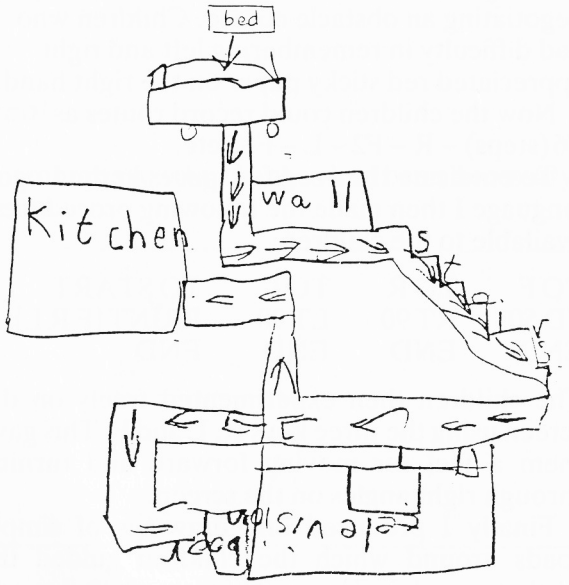


Figure 3 *The route from bedroom to kitchen at home.*

The children finally had to construct a route from A to B on squared paper. When working on the squared paper the children were encouraged to find the shortest/longest route from A to B counting the side of a square as one unit. They recorded the route and length with coloured pencils to avoid confusion.

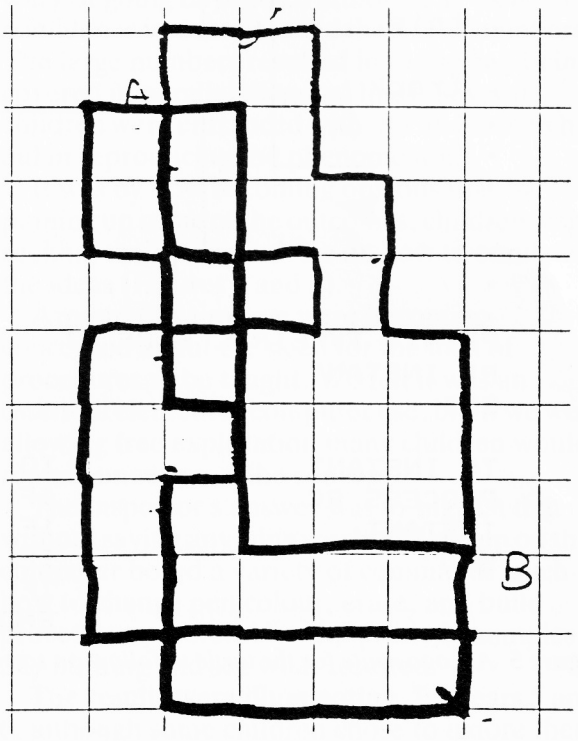


Figure 4 *Route from A to B on squared paper.*

At this point right/left turns (90°) were introduced. We spent a movement lesson following instructions, sometimes as a class, sometimes working with a friend, and some time was spent negotiating an obstacle course. Children who had difficulty in remembering left and right appreciated red sticky paper on the right hand.

Now the children could record routes as F6(steps) – R – F2 – L – F12 etc.

To eradicate the need for spaces in the Logo language I then made the following procedures available to the children:

```
TO F      TO R      TO L      TO START
FD 50     RT 90     LT 90     PRINT [F R L]
END       END       END       END
```

The children then experimented freely on the screen using the three keys R, L and F. This gave them a feel for moving forward and turning through right angles on the screen.

Finally I produced line drawings of simple roads around which the children guided the turtle to reach its house. The procedure provided a key to clear the screen and reproduce the 'map' if the child made a mistake and wanted to start again.

At this level the children had enjoyed the tasks and experienced the following parts of the Mathematics National Curriculum:

Attainment Target 1
Levels 1, 2, and 3 (a) (b) (c).
Attainment Target 2
Level 1 (a)
Level 2 (a) (d)
Attainment Target 4
Level 1 (b) (c)
Level 2 (b) (c)

The maps/mazes can be made more sophisticated and complicated by introducing various degrees of turn.

The topic of 'Plans and Routes' can be revisited when the children are older to provide experiences of many more Attainment Targets – numerical work with scale, distances, time, angles of turn, compass points etc.

```
Programs used :-

TO TRACK
.
. Program
.
. to
.
. produce
.
. this
.
. track
.
.
.
.
.
.
.
PRINT  TYPE[R L F OR C]
PD INSTANT
END

TO INSTANT
PROCESS RC
INSTANT
END

TO PROCESS :K
IF :K="F [FD 50]
IF :K="L [LT 90]
IF :K="R [RT 90]
IF :K="C [C]
END

TO C
CS
TRACK
END
```

Figure 5 *A Logo route for the turtle to follow on screen.*

A case of action research with Logo

Wendy Harknett
Previously IT Co-ordinator for Cranford Park School

In September of last year a group of four teachers decided to proceed with a piece of action research based on how children learn with Logo. The aim was to establish how far Papert was right in saying that children could learn interactively with Logo without input from teachers. The method of action research is standing back, observing and analysing what is happening. Three of us had a different age group of children, so two staff were teaching Year 6, one teacher Year 5 and the other Year 4. None of the children had had much prior experience of using Logo. Indeed only two of us knew of Papert and his philosophy. We had the advantage of guidance and help from the General Inspector of IT in our borough.

Initially the four of us discussed what to do, and we decided to set the children a task – something simple, such as a closed shape. We monitored this for a week before the inspector came to talk with us. After this meeting we realised that we had been totally wrong in our direction. We had dictated the content, and the children had no ownership of the task at all.

We then decided that we would leave the choice entirely with the children. Each person in the class, with a partner of their choice, would sit at the computer and draw anything they wanted, printing it out at the end and then choosing the next pair of children.

Various observations resulted, principally that the younger ones wanted to draw 'things', while the older ones were more interested in shapes and patterns.

After another talk with the inspector on these outcomes we decided to tighten it up regarding recording. Books were given so that children could first draw the idea they wanted to try, then predict what might be tricky or easy, and list their reasons for their choice. They then put their names on a list – 'I have an idea I want to try' – and when their turn came up they could select a partner and try to draw their idea. On printing it out they selected the next on the list.

After a few days it became apparent that some were reluctant to print out – 'That means my turn is over!' In order to get the idea of process we encouraged printing intermittently to see the design build up. They then evaluated the final outcome as to why changes had been necessary or not.

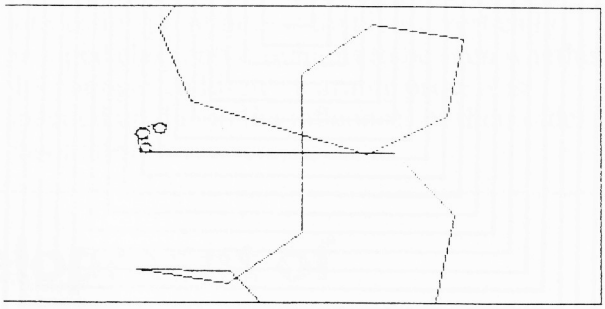


Figure 1 *Shapes and patterns*

Interestingly, in the case of the youngest children, one of them had got locked into turns of 200 and inadvertently produced a perfect prism in the style of a Toblerone box. After this many in the class decided they would also stay with one number for turns, with various outcomes.

In Year 5, one child became involved with using very large numbers, (some children were wary of going beyond 20 although as their confidence increased so did the size of numbers). The large numbers resulted in the screen being covered in parallel diagonal lines. Several children were entranced with this and sought his aid in reproducing the phenomenon.

It was by now becoming obvious that by pinning up some of the outcomes, children were seeking their peers' advice on how to emulate the ideas (Figures 2 and 3).

Around this time we were becoming concerned about the need for the idea of procedures to be taught. We felt it was an essential element in computer use, but if we were allowing free exploration many children would remain unaware of the concept.

The inspector's answer was to suggest that without saying anything we should pin up on the computer board a variety of commands, such as how to change pen colour, erase, and build procedures, preferably with printed examples. Say nothing and see what resulted.

The results were illuminating. In Years 5 and 6, although some children chose to ignore them altogether and work at their ideal picture, others

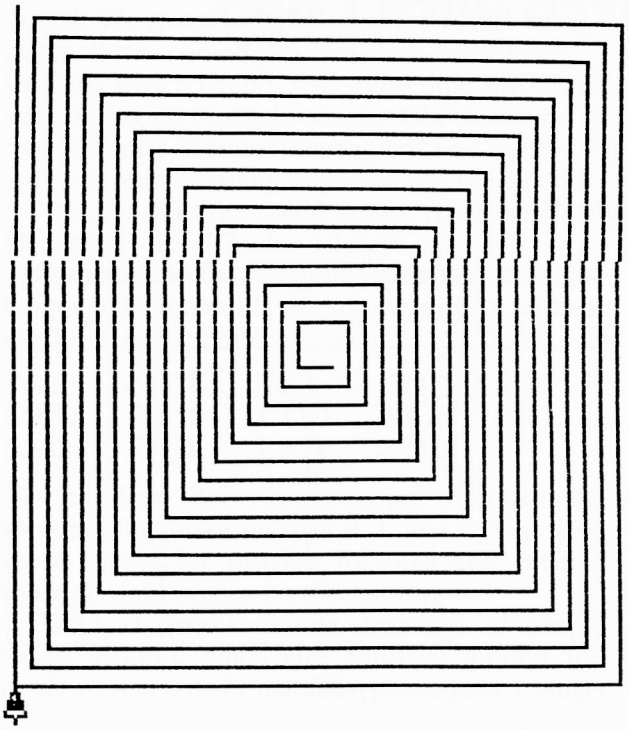


Figure 2 *Spiral with parallel lines*

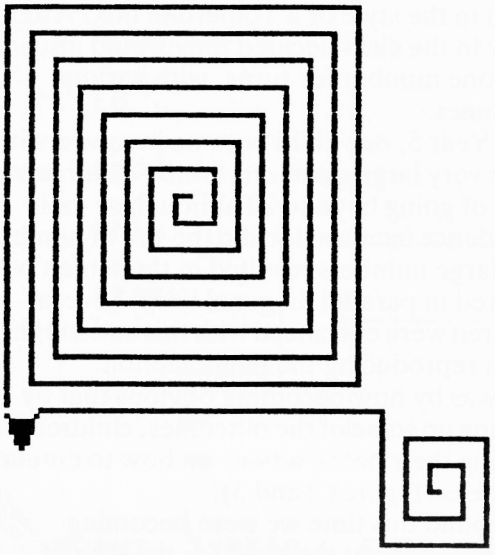


Figure 3 *An attempt to emulate the above*

were keen to try out these new additions. There then followed a rash of square-building and geometric shapes. They slowly began to build a procedure, turn the turtle and call up the procedure again. Gradually one or two began to realise that they could incorporate the turn into the procedure.

Year 4 were still very much in the realms of drawing outlines of letters and pictures. They weren't interested in the more abstract notions of shape.

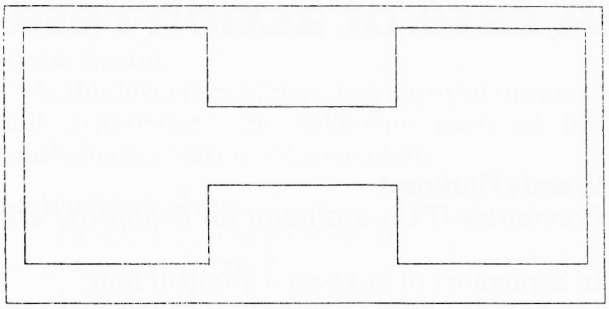


Figure 4 *A closed shape*

One of our difficulties was the difference in the programs of the BBC Master and RM Nimbus machines. The school was so organised that Years 3 and 4 had only Masters and Years 5 and 6 Nimbus. The turtle is less clear on the Master, being difficult to see the direction in which it is pointing, and a diagonal line is more likely to be stepped. This caused great frustration for the younger children. Nimbus has the added advantage of colour.

While the children adapted to the difficulties in time, the teachers found the different commands annoying: rubber/erase, pen up/lift, etc. We had to keep consulting manuals. The lists meant for children's use became essential for us as well.

This had taken us three months, because one of the main 'drawbacks' of Logo and this way of working means children must be given time to explore. With 30 children in the class this takes some time. Two of us were fortunate in having two machines in the room, which released the others for essential data handling and word-processing work. Those with only one machine had to evolve a system whereby it could only be used for Logo in the afternoons or some such arrangement. There came a great cry for two machines per room, but then space would be a problem.

By Christmas the work looked to be heading in a most enlightening direction. The children were definitely stretching themselves, given the teacher-strategic placing of material for those who wished to explore those options. Again this needed teachers to have a variety of material available. It was a combination of this subtle teacher input and children learning directly from their peers. One or two children had Logo derivatives at home and they were very keen to bring in samples of work from these, rekindling interest at home.

The next step, the inspector felt, once the children were comfortable with building procedures and had breached the pattern phase, was to start making pictures incorporating

procedures, eg a castle with turrets and windows sub-procedures.

How long it takes children to get to that stage in the cycle seems to depend on their intellectual maturity. We have yet to analyse this. It seems apparent that the younger children need more practice at each stage. The older ones, however, would revert to previous stages for fun, and possibly a sense of security.

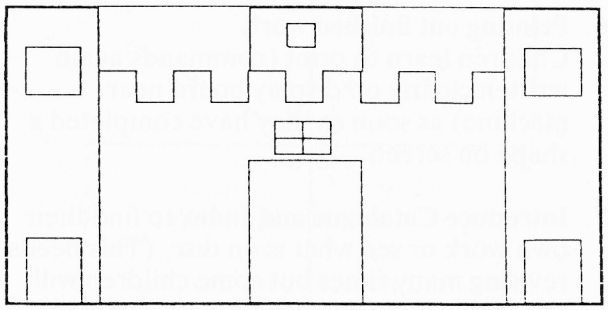


Figure 5 *Drawing of a castle with turrets and windows.*

The benefit of the recording books is that there is a record of their progress which we hope will be added to as they proceed through the junior school. They should be a valuable record upon entering the secondary phase.

I am currently starting afresh in a different school in Oxfordshire, and so far the children have followed similar learning patterns. The interesting point here is that it is a vertically grouped class and it remains to be seen whether the younger children's learning process is speeded up due to the influences of their older classmates.

Introduction and development of turtle graphics using Logo

Jane Carson
Deputy Head/Technology Co-ordinator, Roselands J.M.I School, Hoddesdon

We are a small primary school, well-supported by our PTA, having now the benefit of two computers per classroom. Thus we have looked at the introduction of turtle graphics and the progression in the use of Logo throughout our JMI.

Following the introduction in mid/top infants of the floor turtle and related directional games and activities, we then proceed in a fairly structured way to develop the use of the screen turtle. We often use class or group teaching sessions to introduce a task, which each 'pair' of children then works on when it is their computer turn. The pairs of children are listed on a rota, displayed near the computer and marked off by teacher/children each time they have a turn. Sometimes this session will be limited by time, and sometimes by the stage in the task the children have reached. In the older age groups the pairs each have their own discs for a year and save their work and re-load it when appropriate. Thus their disc management skills progress too, as work from all programs used will be on their own discs.

The tasks introduced are often reinforced by the use of work/task cards for basic shape work and quadrants with co-ordinates.

The order of work we have found to be of most use so far is:

1. **Practise moving the turtle** around the screen in 'free play' sessions with basic commands – FD BK RT LT PU PD CS ST HOME. A list of these commands, plus colour numbers, loading and saving procedures and 'exit', are all written up on a display board with the rota beside the machine. These offer easy reference for both beginner children and staff.

2. **Awareness of screen size** is encouraged. This is especially important when moving from BBC machines to RM Nimbus where linear move sizes are quite different. Sometimes FENCE WRAP and NOFENCE are introduced here.

3. **Making** the child's own initial in *direct drive*, or the initials for each 'pair', eg **TB**, or **L**. 'Turns' are now changed to 90°, 180° and 270°, and soon 45° is mentioned. Angles in degrees seem to be very easily accepted and used.

Figure 1 *Initial L.*

4. **Introducing procedures** – saving initials previously made by using a simple procedure.
   a. Make procedure – giving it a name.
   b. Saving the procedure under a filename, noting this must be different from procedure name.
   c. Coming out of Logo, re-loading and finding procedure again.
   Does the procedure still work?
   The time spent with this simple procedure seems extremely valuable as it is the basis for so much of the following work and the children derive great satisfaction from being able to re-load and see 'It is still there!'

5. **Reinforcement of simple procedure work:**
   a. Make a simple shape (eg square, triangle, rectangle). We often provide a box of named geometric shapes to help. A great deal of valuable discussion and vocabulary often emerges, as do many questions, usually including 'How do you make a circle?'

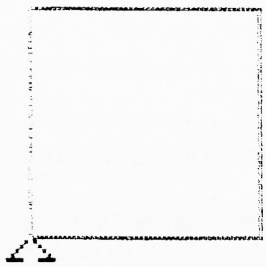


Figure 2 *Make a square.*

   This may be a point at which to draw group/class together and explain, as far as they can understand. Sometimes it is necessary to just say 'This may work: REPEAT 36[FD 20 RT 10] (These must multiply to make 360!) Some children will be too young or inexperienced to continue with this line of investigation. We have seen others, however, extend themselves just in order to produce something they really want to appear on the screen.
   b. *Make a simple picture.* In order to avert disappointment it helps to encourage simple linear artlines. They will still tend to become involved in quite amazing points of detail, without being asked to focus on this aspect! Here it is also often quite incredible to see the level of perseverance of some of the most unlikely characters.

   In all these early tasks we encourage the noting down by the one of the pair who isn't actively 'at the keyboard' of all the direct drive moves. These can then be typed into procedures easily.
   c. *Use of the Editor* will automatically be introduced to make procedure, correct and extend work.

6. **Printing out finished work**
   Children learn to print (commands again written clearly on display board near machine) as soon as they have completed a shape on screen.

7. **Introduce Catalogue and Index** to find their own work or see what is on disc. (This needs revising many times but some children will retain this early on and it can be useful in helping those who 'can't find' their work when the teacher already has 10 other pupils waiting!)

8. **Use of REPEAT**
   If original simple shapes or initials are loaded back into the computer they can be repeated to give pleasing, attractive patterns, eg REPEAT 12[SQUARE RT 20].

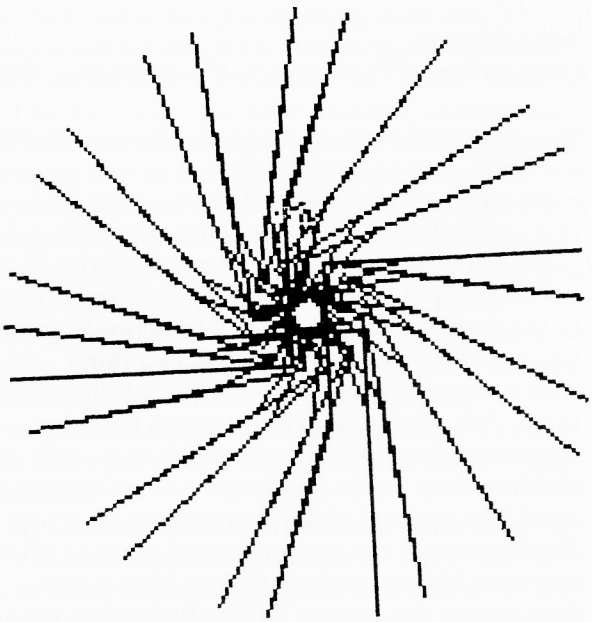


Figure 3 *Repeat of L.*

Gradually the children also see when the turtle draws over itself, or if the number of repeats is not enough, or too many. Some may even consider relationships between number of repeats and angle of turn. These repeated patterns may also be put into procedure and saved.

Now the beginning of 'using procedures' has really begun.

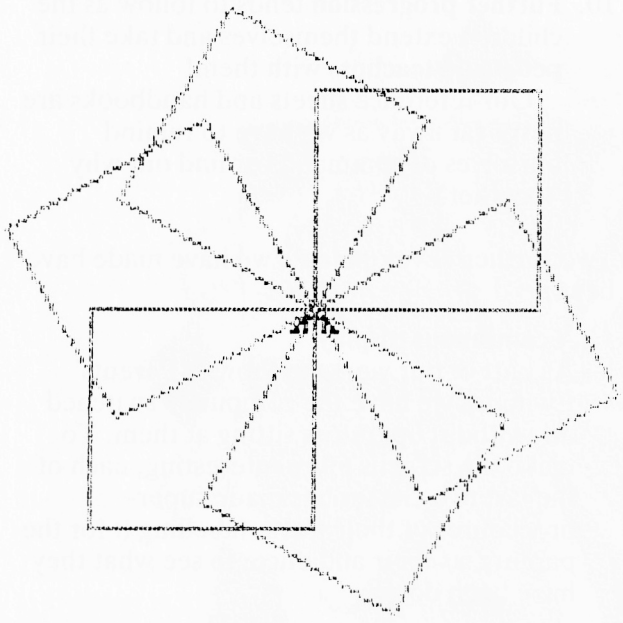


Figure 4 *Repeat of square.*

9. **Colour and Fill**
   Again, we often use our basic shape, re-load it and fill in different ways. This can then be used with repeat. Background, border and pen colours are all shown. Use of rubber and re-setting pen-colour is usually introduced here.

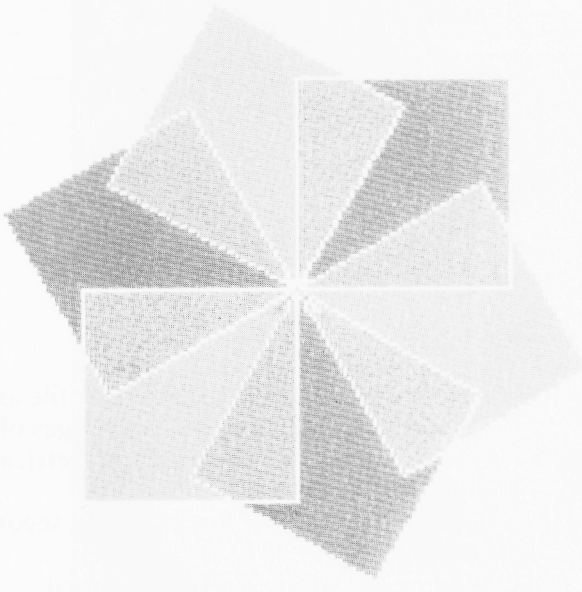
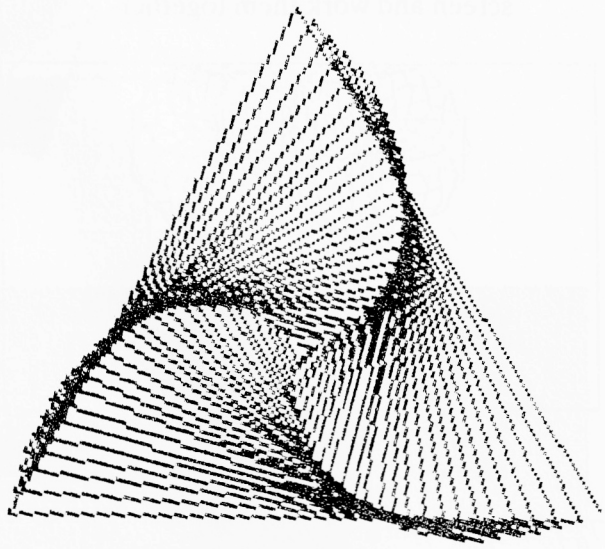


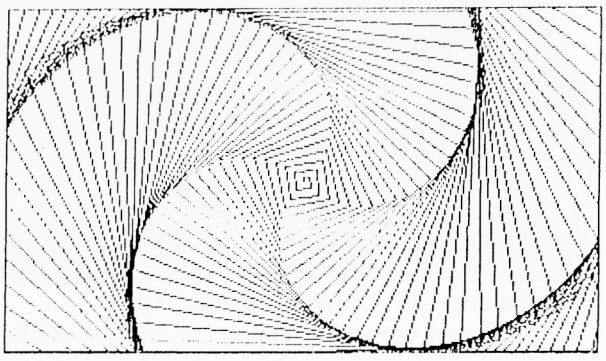
Figure 5 *Colour and Fill with Repeat.*

10. **Reinforcement of work learned so far.**
    The work so far may take at least a year, maybe two, to develop and can now be varied according to the interests and abilities of the children.
    More advanced shapes and pictures may all be made but the basic principles of direct drive → procedure → save → re-load will be continued.

We then extend our Logo work when we feel these basic principles are firmly established. Extensions will include:

1. **Using Superprocedures**
   Using one procedure within another often increasing basic understanding and overview by pupils as they plan their task.
2. **Using Setscrunch.**
3. **Using variables** of size, length, angles.
4. Developing work with variables to produce **spirals** (see Figures 6 and 7).





Figures 6 & 7 *Examples of spirals.*

5. **Quadrants – using co-ordinates**
Logo is a valuable tool when explaining minus numbers to children and in using the quadrant of the screen we have introduced co-ordinates and related work quite successfully.

6. **Multiple-Turtles**
RM Logo offers the user up to eight turtles. These are easily introduced when co-ordinates have been used and offer a wide scope for much of the previous basic work to be developed. If you can draw a spiral with one, can you do the same with eight?
We aim to develop this work with two main tasks:
(i) Make each turtle do the same things, in sequence, in different places on the screen.
(ii) Make each turtle go to an area of the screen and work them together.

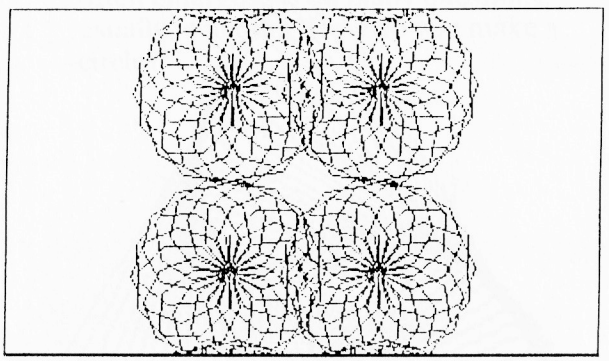


Figure 8 *Four turtles together.*

7. **Labels for work**
In writing one word or sentence with a screen picture, the use of text then sometimes becomes a natural development.

8. **Text and turtles**
Adventure games can be written by the children showing great levels of persistence and skills in planning as each 'alternative route' must be mapped and written in separate procedures. These offer a great challenge to the swift, agile minds of the keen Logo pupils!

9. **Control Logo**
When procedures and basic commands are fully understood and discs are managed with confidence, the children are able to transfer skills and use control programs and equipment with ease. They can extend their designs and models without 'thinking' about making the program to work it as the language of Logo has become another tool they can use in a technological environment.

10. **Further progression** tends to follow as the children extend themselves and take their peers and teachers with them!
Our reference sheets and handbooks are never far away as we have to remind ourselves of commands or find out why 'together'.

Two further developments we have made have been:

1. **A continuous display**
As part of our work on show at Parents' Evenings we have the computers switched on, without operators sitting at them. To make the screens more interesting, each of the last two classes has made super-procedures of their work, labelling it for the parents, as their audience, to see what they have been doing.

2. **Logo → craft and cover designs**
We have printed out simple designs made with Logo and used them as a basis for press-printing onto paper to then use as folder covers for topic work.
a. Simple design made and printed in black and white.

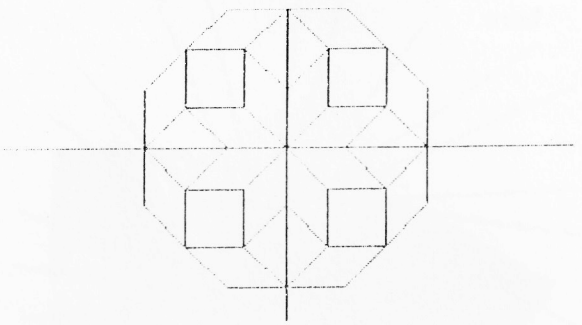


Figure 9 *The basic design.*

b. Design traced onto a polystyrene tile and lines extended from pattern to edges of tile. (These are likely to be symmetrical and join to each other.)
c. Tile printed and over-printed in a second colour.
d. Pattern ink-sprayed through stencil shapes.
e. Cover made for folder from patterned, printed paper.
The basic shape on Logo is then also extended, coloured, repeated, put into quadrants or developed as the pupil wishes and finally displayed with cover to show a variety of ways of developing a simple 'turtle' picture.

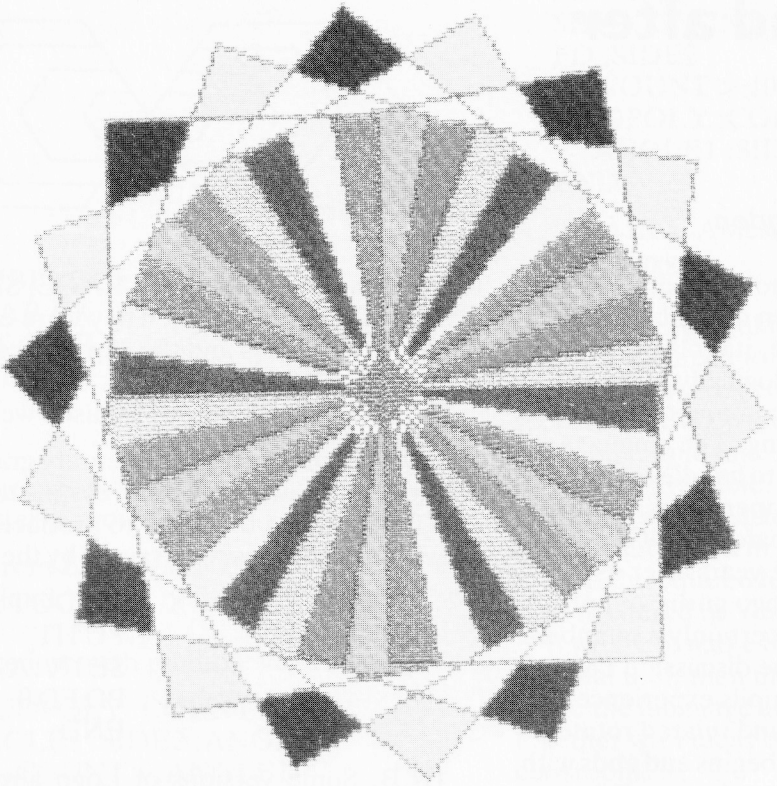


Figure 10 *The developed pattern.*

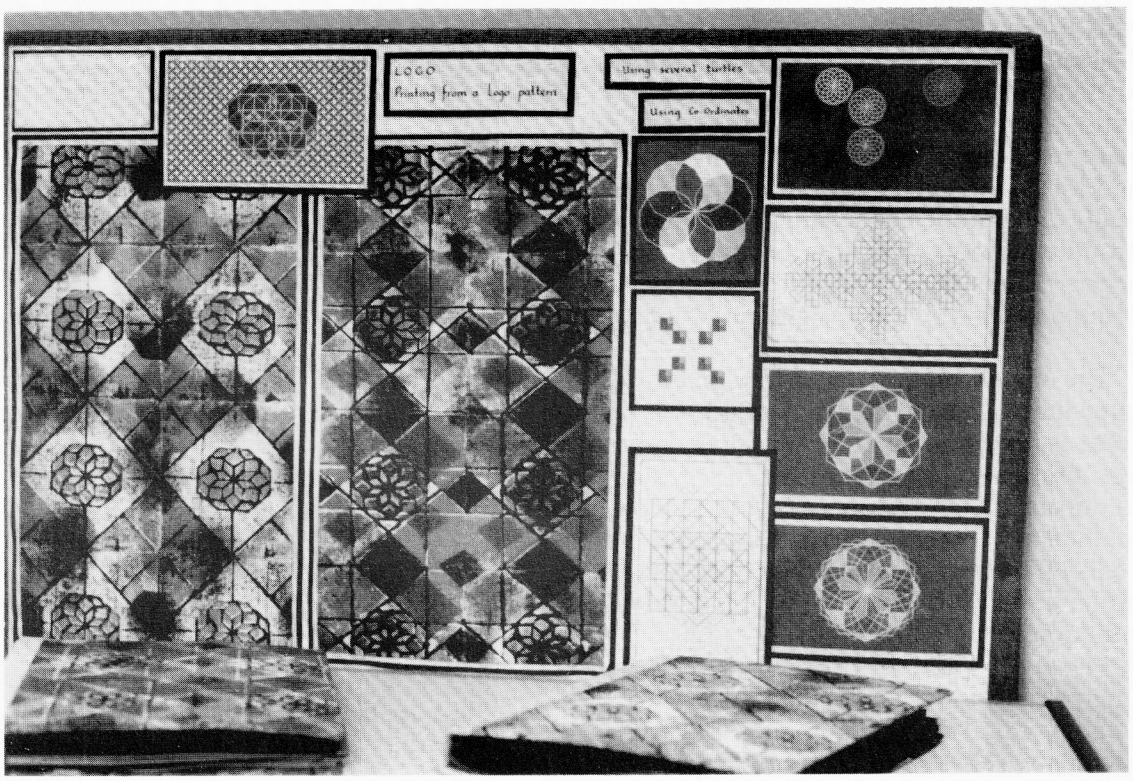


Figure 11 *Newspaper photograph of display of folders and designs.*

TO BOX and after

Betty Lumley
St Helen's College, Hillingdon

Since *Mindstorms* and Seymour Papert there has been considerable discussion of the benefits of using Logo. The experience, it is claimed, encourages pupils to think logically, pose and solve problems and feel in command of the micro-world they are creating. The language itself is versatile, being able to handle text, turtle graphics and mathematical operations. Finally it has been mentioned in despatches (NC).

Now however, some nine years after its introduction, it is time for the Logo gurus and pro-Logo lobby (of which I am certainly a member) to take stock. Despite all the discussion I am of the opinion that very few pupils experience much more than the BOX and related rotated patterns. Progression often begins and ends with the same treatment given to polygons. What a waste!

Every year the London Borough of Hillingdon runs a series of eight Saturday morning Master Classes for the most mathematically able pupils of Year 9. One session is devoted to the mathematical applications of computers. Three years ago I was running this session using Logo and was surprised by how little the students really understood of the language. They could produce stereotyped patterns but had no appreciation of link procedures or knowledge of commands that give an insight into what is going on.

The Maths Support Team had previously introduced Logo to the primary schools and run courses for secondary teachers. Last February I again returned to Logo during the Master Class to pursue my interest in using Logo as a tool for mathematical investigation. Now, some seven years after its original introduction to schools, nothing seems to have changed; if anything the situation seems worse.

The group I met were selected to do advanced Logo as a result of having had experience of Logo, yet only 20 per cent knew how to create a procedure. Most worked from the keyboard. The idea of a procedure was seized upon as being very efficient. Still 25 per cent of the pupils went through the stage of inventing the rotating patterns which they were convinced is what Logo is about.

The rest of the pupils were delighted to be able to use commands like OP, TRACE, SETX, SETY, SETH, IF and [ ]. They were excited by the use of variables and the experience of recursion.

The three investigations we used were:

1. A way of producing rudimentary graphs using the two following procedures; one to produce a function and the other to produce a dot whose position is constrained by the function.

```
TO FUNCTION :X
OP :X+5
END
```

```
TO DOT :A :B
PU HT
SETX :A SETY :B
PD FD 0
END
```

(N.B. Some versions of Logo already have DOT as a primitive.)

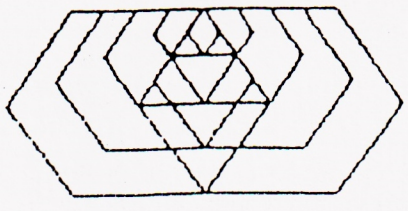
This led one group to write a procedure to plot automatically rather than keep typing

```
DOT 10 FUNCTION 10
DOT 20 FUNCTION 20   to make two dots
```

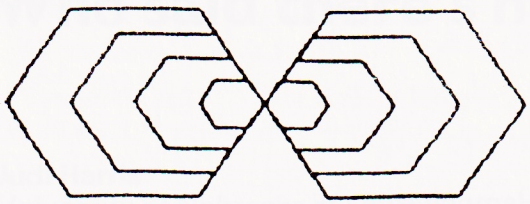
```
TO GRAF :A :B
DOT :A FUNCTION :B
IF :B > 60 [STOP]
GRAF :A+5 :B +5
END
```

This meant that the FUNCTION procedure had to be changed each time a different graph was wanted. This would have been the next 'problem' to solve had there been more time.

2. Recursion using TRACE and the following procedures.



```
TO PROCEDURE :VAR
IF :VAR > 100 [STOP]
SUBPROCEDURE1 :VAR
PROCEDURE :VAR + 30
SUBPROCEDURE1 :VAR
END
```



```
TO SUBPROCEDURE1 :VAR
FD :VAR BK :VAR
PRINT :VAR
RT 90 FD 30
LT 90
END
```

Some groups worked on trying to reproduce the patterns provided.

(Idea from *Logo Maths Project* – Hoyles and Sutherland.)

3. A procedure with multiple variables.

```
TO DUOPOLY :COUNT :SIDE1
  :ANGLE1 :SIDE2 :ANGLE2
SETH :COUNT * :ANGLE1
FD :SIDE1
SETH :COUNT * :ANGLE2
FD :SIDE2
IF :COUNT > 10 [STOP]
DUOPOLY :COUNT + 1 :SIDE1
  :ANGLE1 :SIDE2 :ANGLE2
END
```

(Idea from Ablesson & Di Sessa's *Turtle Geometry*.)

In the past the thinking was to let children discover the need for new commands, to pose and set their own problems, but this does not seem to have been too successful. This approach relies on ample time and resources and neither have been available. Perhaps a more structured approach would ensure progression and get over the present problem that Year 9 children are often only required to reproduce the Logo work they mastered in Year 6.

If we are to make the most of Logo as an educational tool, then more needs to be done to make the language accessible to teachers, together with ideas on how to use it in the classroom.

One small leap for turtle kind?

(An adapted tongue-in-cheek look at a turtle development)

Les Watson
Cheltenham & Gloucester College of HE, Cheltenham

Once upon a time, two Logotics, Bert and Craisy, were playing with their mice when Bert had an idea:

Bert: Hey, Craisy, I've just had a great idea!
Craisy: Oh yeah, Bert, What's that?
Bert: 4D Logo.
Craisy: Don't be daft, Bert. We don't know what to do with 3D Logo yet.
Bert: No, listen. This is 3D Logo but on a network. The network is the fourth dimension. If we used a Nimbus network, we could have 8 turtles in 3 dimensions for each station.
Craisy: So?
Bert: Well, one aspect of this fourth dimension is that each user can send their turtles to the next workstation.
Craisy: Yes, I like it. if they all sent them to the end machine on a 16 station network, we could have 128 3D turtles on screen. That'd really blow your mind!
Bert: But listen. If you had a modem on the network, when a turtle went off the bottom of the screen, it could go along the telephone lines and pop up on any screen connected to the telephone system.
Craisy: It's getting better! All 128 turtles from one network could migrate to another network. This is too much!
Bert: There's more. We could send them off the top of the screen via satellite down onto the top of anybody else's screen anywhere in the world. Global migration of turtles!
Craisy: I think we've created the first macroworld. Soon everybody will know about turtle graphics!
Bert: Hang on though, why restrict it to computers? Any one of the turtles going backwards into the screen could travel into the mains supply and along the cables. Unsuspecting people anywhere in the world could open the oven, or take the lid off the kettle to find them full of turtles!
Craisy: Wait a minute though, Bert. What's the educational value of this?
Bert: Who cares? From the underground cables, they could escape through sewer pipes. Television companies could make stories of them, toy manufacturers could make plastic models of them . . .
Craisy: And I suppose they'd be named after Italian rennaisance painters? No. It'll never catch on.

Say it in Logo

Chris Robinson
BLUG and Deputy Head, Horndean Middle School, Hampshire

There is more to Logo than just drawing a square and, if you're lucky, rotating it to make a pretty pattern – that can be accomplished with any turtle geometry program.

Children using Logo 'talk' to the computer, the turtle depicted on the screen, the floor turtle controlled by it or the control interface attached. They use real words, typed on its keyboard, and the computer obeys their commands or can talk back, using text displayed on the screen.

Logo tries to be helpful and friendly. If it doesn't understand what is asked of it, it says so. Hence if you type 'HELP', it will reply, 'I DON'T KNOW HOW TO HELP'.

If you type 'Hello', wouldn't it be nicer to have the computer reply, 'Hello. Pleased to meet you', rather than 'I DON'T KNOW HOW TO HELLO'!

All we need to do is teach the computer how to respond when we talk to it. We need to teach it how to recognise some common English words and respond to them. So how do we teach it to respond to HELLO?

```
TO HELLO
PRINT [Hello. Pleased to meet you.]
END
```

In 1984, I was carrying out research work in two different schools 200 miles apart. A class from each school used the language ideas to tell each other about themselves.

```
TO STARTUP
TS
PRINT [Who do you want to know about?]
PRINT [ ]
END
```

(TS clears the Screen for Text. PRINT [ ] gives a blank line).

The children then had to tell the computer about themselves:

```
TO PETER_E
PR [Peter Edwards is ten years old.]
PR [ ] PR [He goes to Ink Pen Primary.]
PR [ ] PR [His interests are fishing, football
  and cycling.]
etc.
```

Using POTS or PR OPPS the computer listed the names of the procedures, which were mainly, of course, the children's names.

The following procedure was provided to make this easier:

```
TO CLASS
PR [I can tell you about:]
DETAIL OPPS
END

TO DETAIL :list
IF FIRST :list = STARTUP [MAKE "list
  BF :list]
IF FIRST :list = "NAMES [MAKE "list
  BF :list]
IF FIRST :list = "DETAIL [MAKE "list
  BF :list]
IF :list = [ ] [STOP]
PR FIRST :list
DETAIL BF :list
END
```

The command, CLASS, could then be added as a final line in the STARTUP procedure.

Those with a deeper knowledge of list processing can write procedures to search the texts of these entries for particular entries if desired.

Who said there's no knocking Logo?

Judi Harris
University of Nebraska at Omaha

If your students saw **KNOCK, KNOCK.** displayed on a computer's screen, what do you suspect the majority of them would type in reply? Probably something like: **WHO'S THERE?** Right? Interactive 'knock-knock jokes' are as much a part of Western popular culture as jump rope rhymes and rap songs. Their predictable pattern make them excellent candidates for Logo and language arts explorations. Programming the computer to initiate and respond to 'knock-knock' jokes is a simple, yet powerful way to help students of any age to become comfortable with interactive text procedure writing in Logo.

Knocking around

Let's examine the structure of a sample 'knock-knock' interchange. It occurs in five (or six) steps.

1. Someone initiates the interaction by saying: 'knock, knock'.
2. Someone responds: 'who's there?'
3. The first person says a name, such as 'SEYMOUR'.
4. The other person, using the name supplied, queries: 'Seymour who?'
5. The conversation initiator then presents the punchline: 'You'd Seymour if you opened your eyes!'
6. Either (or both) of the participants laugh and/or groan.

Steps 1, 3 and 5 are performed by one participant in the interchange; steps 2 and 4 are performed by the other. Let's first assume that the computer user initiates the interchange in a coded-in-Logo version of this joke form. The user would type **KNOCK, KNOCK.**

Logo looks at this entry as a series of two potential commands: **KNOCK,** and **KNOCK.** (Any series of characters separated from any other series of characters by a space is 'understood' by Logo to be a command, unless special punctuation, such as " or : immediately precedes the character series). Since we have not yet 'taught' Logo to understand the **KNOCK,** or **KNOCK.** commands, it will tell us, **I don't know how to KNOCK.**

Shall we teach it how to respond to **KNOCK, KNOCK.**? As in step two (above), the first thing that the computer must do is to say **WHO'S THERE?**.

We can tell it to do this quite simply. Since the user will be typing two commands, **KNOCK,** and **KNOCK.**, the first can be coded to print the expected question.

```
TO KNOCK,
PRINT [WHO'S THERE?]
END
```

Once this procedure is defined in the computer's memory, we can invoke it by typing **KNOCK,** in the Command Center, and pressing the <Return> key. The computer will print the expected question as a reply: **WHO'S THERE?**

If the user forgets to type the comma that is the last character of the **KNOCK,** command – and instead types **KNOCK** – the computer will politely tell us:

```
I don't know how to KNOCK
```

We could help future users of our knock-knock program by providing a 'user-friendly' procedure, such as:

```
TO KNOCK
KNOCK,
END
```

. . . so that typing either **KNOCK,** *or* **KNOCK** would elicit the desired response.

Not-so-hard knocks

Our user now sees a question (**WHO'S THERE?**) to which they will probably type an answer. We must make the computer ready to *receive* their answer, then respond appropriately. Remember that the user has typed *two* commands: **KNOCK,** (or **KNOCK**) and **KNOCK.** So far, we have defined only the **KNOCK,** (or **KNOCK**) procedure. The second procedure must:

1. accept the answer to the **WHO'S THERE?** question,
2. repeat that answer, followed by **WHO?**,
3. accept the answer to the **WHO?** question,
4. and respond appropriately.

Steps 1 and 2 can be combined in one line of code by using PRINT, SENTENCE and READLIST primitives. READLIST tells the computer to wait for whatever the user types in before pressing the <Return> key. We can concatenate the user's answer with the word **WHO?** using the SENTENCE command. The PRINT command tells the computer to display the results of the concatenation for the user to see, All together, directions for these three actions look like this:

```
PRINT SENTENCE READLIST "WHO?
```

At this point in the interaction, the user has the pleasure of providing the punchline. If the computer is not prepared to 'listen', then a confusing error message will be displayed, such as: **I don't know how to YOU'D** (from the sample knock-knock joke at the beginning of the article).

Obviously, the READLIST primitive will be helpful here, also. READLIST is a type of primitive called a *reporter*; it must report input (in this case, the users answer to the **WHO?** question) to a command. We will create this command, calling it LISTEN:

```
TO LISTEN :LIST
END
```

That isn't a typo! All LISTEN does is to accept the input from the user (through READLIST), then END. No overt action is taken.

The polite listener to knock-knock jokes laughs after the punchline is delivered, whether naturally inclined to do so or not. Our computer can do this with a simple:

```
PRINT [HA, HA, HA!]
```

All together, steps 1 through 4 above can be coded in the **KNOCK.** procedure as follows.

```
TO KNOCK.
PRINT SENTENCE READLIST "WHO?
LISTEN READLIST
PRINT [HA, HA, HA!]
END
```

These will be invoked only *after* the code in **KNOCK,** (or **KNOCK**) is executed, since the user typed: **KNOCK, KNOCK.** – and Logo, being a list processing language, executes commands presented on the same line sequentially from left to right.

For users that prefer greater emphasis expressed in their interactions, the **KNOCK!** procedure can be offered as an alternative to **KNOCK.** With this tool, users can begin the interactions with **KNOCK, KNOCK!**

```
TO KNOCK!
KNOCK.
END
```

Knocking 'n ease

As developed with the procedures listed above, the computer operates as a friendly audience for computer users' knock-knock jokes. How might we code procedures so that those roles were *switched*? What if students programmed the *computer* to be the punchline-deliverer?

First, it would have to initiate the interaction. Using a STARTUP procedure in *LogoWriter* makes procedure contents execute automatically when the page is selected from a diskette table of contents. This STARTUP procedure hides the turtle, clears the Command Centre, then begins the interaction.

```
TO STARTUP
HT CC (TS for "ordinary" Logo)
PRINT [KNOCK, KNOCK!]
END
```

The user, of course, should type: **WHO'S THERE?** Logo 'sees' this as two sequential commands, **WHO'S** and **THERE?**. Since the computer now will be supplying the name of who *is* there and the related punchline, there must be a supply of names from which the computer can randomly choose (with accompanying punchlines) stored in the computer's memory. We can store them in a procedure called **WHO.IS.THERE**:

```
TO WHO.IS.THERE
OUTPUT [BET JEWEL ALDA DISHES
  AMMONIA GUS HOWARD AUTO
  ARFUR N.E.]
END
```

WHO.IS.THERE is a reporter of sorts, since the list of 10 'who's' is OUTPUT only; no processing of these names is called for in this procedure. First, we'd like the computer to pick a name from the 10 at random, then print it in response to the user's **WHO'S THERE?** question. We can use a familiar tool procedure, PICK, to make the selection from the output from **WHO.IS.THERE**, and, since PICK is also a reporter, we can tell the computer to PRINT the random choice in the **THERE?** procedure.

```
TO PICK :LIST
OUTPUT ITEM 1 + RANDOM COUNT
  :LIST :LIST
END
```

```
TO THERE?
PRINT PICK WHO.IS.THERE
END
```

Typing **THERE?** might yield: **ALDA** one time, **HOWARD** another time, or **JEWEL** at another time. Remember, though, that the user will have typed ***WHO'S*** **THERE?**, not just **THERE?**, so we must code something for the new command **WHO'S** to do so too. Since **THERE?** has performed all of the activity needed at this point in the dialogue, **WHO'S** can be used to print a blank line.

```
TO WHO'S
PRINT [ ]
END
```

Knock and the answer shall be given

So far in this scenario, the computer has prompted: **KNOCK, KNOCK.** – and the user has replied: **WHO'S THERE?** The computer then chose a name from the collection in the **WHO.IS.THERE.** procedure and printed it. Perhaps it printed: **JEWEL.** The user will then (probably) type: **Jewel who?**

Logo 'sees' this as two commands: **JEWEL** and **WHO? JEWEL** must be coded as a procedure that supplies a retort specific to that name. In other words it would be disappointing to the user if they typed **JEWEL** and the computer replied, **NORMA LEE I DON'T GO AROUND KNOCKING ON DOORS, BUT YOU SEE I HAVE THIS WONDERFUL SET OF ENCYCLOPEDIAS TO SHOW YOU.**

Needless to say, it wouldn't be very funny, either. The punchline that corresponds to the name JEWEL could be stored in a procedure called JEWEL. The procedure **TO JEWEL** could tell the computer to PRINT that punchline.

```
TO JEWEL
PRINT [JEWEL REMEMBER ME WHEN
  YOU SEE MY FACE.]
END
```

Retorts appropriate to the other names that the computer might select at random could be similarly coded.

```
TO BET
PRINT [BET YOU DON'T KNOW WHO'S
  KNOCKING ON YOUR DOOR!]
END

TO ALDA
PRINT [ALDA TIME YOU KNEW WHO
  IT WAS.]
END

TO DISHES
PRINT [DISHES YOUR FRIEND; OPEN
  THE DOOR.]
END

TO AMMONIA
PRINT [AMMONIA GONNA TELL YOU
  ONCE]
END

TO GUS
PRINT [THAT'S WHAT YOU'RE
  SUPPOSED TO DO.]
END

TO HOWARD
PRINT [HOWARD I KNOW?]
END

TO AUTO
PRINT [AUTO KNOW, BUT I'VE
  FORGOTTEN.]
END

TO ARFUR
PRINT [ARFUR GOT.]
END

TO N.E.
PRINT [N.E. BODY YOU LIKE SO LONG
  AS YOU LET ME IN.]
END
```

By invoking any of the 10 'name procedures' the interaction is concluded (except of course, for the hoped-for laughter), so the second half of the user's question, **WHO?** could, like the **WHO'S** procedure above, just print a blank line.

```
TO WHO?
PRINT [ ]
END
```

Your students may want to repeat the interaction to explore the other knock-knock sequences coded on the page. To make these repeat performances a bit easier to request, an AGAIN procedure can be defined.

```
TO AGAIN
STARTUP
END
```

Opportunity's knocking

Once your students are comfortable with the second knock-knock interaction procedure structure, why not encourage them to make a Logo-encoded knock-knock collection? Let's say that someone finds this beauty:

```
KNOCK,KNOCK
Who's there?
KANGA
Kanga who?
NO, KANGAROO
```

It is easy to add it to the collection. First, the punchline must be coded into a new procedure with the name as its title:

```
TO KANGA
PRINT [NO, KANGAROO.]
END
```

Then, the name must be added to the collection from which the computer makes its random choice in the **THERE?** procedure. As you probably remember, that collection is in the **WHO.IS.THERE** procedure.

```
TO WHO.IS.THERE
OUTPUT [BET JEWEL ALDA DISHES
  AMMONIA GUS HOWARD AUTO
  ARFUR N.E. KANGA]
END
```

As you can see, we've added the new name to the list that is output. This list can grow to as many elements as your students care to add and your computer's memory can store . . . in both cases, probably quite a large number!

```
KNOCK, KNOCK.
Who's there?
HACIENDA
Hacienda who?
HACIENDA DA ARTICLE
```

Reference

Ward Lock Ltd (1985), *1,000 knock knock jokes for kids,* New York: Ballantine Books.

This article has been reproduced from 'The Computing Teacher' with permission from the International Society for Technology in Education, 1787 Agate St, Eugene, Oregon.

To boldly go . . .

Chris Robinson
BLUG and Deputy Head, Horndean Middle School, Hampshire

'I am sitting at my word processor seeking inspiration for an article on writing branching stories in Logo when I realise it is sunny outside. Do I WRITE any old rubbish or do I go for a RIDE on my bike?'

If I ask my Logo computer, it will tell me 'I DON'T KNOW HOW TO WRITE' – but I can teach it. So I'll take a ride and think up what the computer can write.

In all adventure (branching) stories, it is important to set the scene first. Where can we set the story? A fairy tale castle with dragons lurking in dungeons and a princess to rescue from a turret? How about round these roads? Now, where do I want to go? I fancy seeing the sea. Shall I go SOUTH down Old Street or WEST along Knight's Bank?

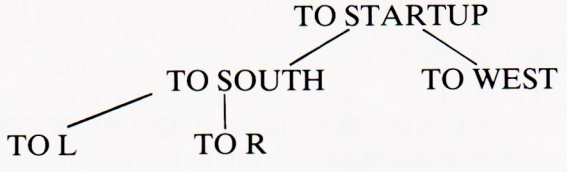
Writing adventure stories in Logo is easy. All I have to do is teach the computer what it doesn't already know. Thus:

```
TO SOUTH
PR [ ] PR [You cycle down Old Street.]
PR [ ] PR [At the end you see the sea.]
PR [ ] PR [Do you turn LEFT or RIGHT?]
END
```

No! That's no good. The computer already knows LEFT and RIGHT to be instructions to turn the turtle. I'll have to change the last line:

```
PR [ ] PR [Type L to turn Left or R to turn
  Right.]
```

Now I'll have to teach it what would happen if I were to type L or R. And I mustn't forget to teach it TO WEST either. Let's see what I've got so far. It would be a good idea to write these procedures out on pieces of paper and spread them in front of me:



That's better. Now I can make sure I don't use the same procedure name more than once, too. Of course, I could have added a third option in the SOUTH procedure. I might have decided, as it's a hot day, to stop at the end of Old Street and TO SWIM. Still, it will be easy to edit it and

add that option later. I must make it clear to the readers/players what options are available to them by highlighting the words in CAPITALS.

It's a shame they won't be able to see the sea too. Perhaps I could write turtle commands to draw it. I could add the line SEA to call up the picture first so the text is printed below it. Or I could load in a digitised screen picture or one created with a painting program. . . .

(This is beginning to sound a bit like, 'Make a representation of a real or imaginery place.' That's 1 Ge 2b but I'm sure this can cover more statements of attainment if I want to look.)

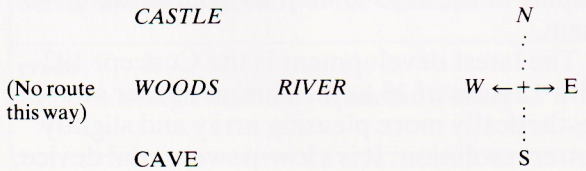
When writing adventure stories, or any amount of plain text, with ordinary Logo (as opposed to *LogoWriter*), thought has to be given to the page layout, counting characters so words don't wrap around from one line to the next and, for clarity's sake, I personally prefer to double space text on screen. The following procedures were written to overcome these problems.

```
TO PT :text
PP :text 1
END

TO PP :text :pos
IF :text = [ ] [STOP]
IF :pos + (COUNT FIRST :TEXT) > 39
  [NEWLINE PP :text 1] [TYPE FIRST :text
  TYPE CHAR 32 PP BF :text :pos +
  (COUNT FIRST :text)+1]
END

TO NEWLINE
PR [ ] PR [ ]
END
```

An alternative way to go adventuring

The limited memory space in a BBC B computer will limit the production of an adventure story to a few restricted locations only, particularly if a graphics mode is used. An alternative is for the details for each location to be loaded from disc over-writing the previous location's details in the computer's memory. The following map and procedures illustrates a way of doing this. (This technique, of course, may be used with any Logo computer!)

Plan for a simple adventure story with four locations



For this example, the disc will eventually hold location files entitled WOODS, RIVER, CASTLE, CAVE.

In each location file will be stored procedures named as follows TO STARTUP, TO PICTURE, TO NORTH, TO SOUTH, TO EAST, TO WEST, even if they are 'dummy' procedures. The NORTH, SOUTH, EAST and WEST procedures must ouput the name of the location file to be loaded if the reader decides to choose the route. The 'management' procedure TO TRAVEL must also be present, although you may prefer to use an alternative such as TO GO, TO WALK, etc if your version of Logo will permit this (ie they may be present as primitives which you cannot change.)

```
TO TRAVEL :destination
IF :destination = " [PR [You can't go that
  way.] STOP]
LOAD :destination
STARTUP
END
```

As an example, the WOODS file may contain the following procedures:

```
TO STARTUP
PICTURE
PT [You are in a clearing in the woods. To the
  East there is a river. A path runs North and
  South. Which direction do you wish to
  travel?]
END

TO PICTURE
(Turtle drawing commands to draw scene, or
  LOADPIC "TREES)
END

TO EAST
OP "RIVER
END

TO WEST
OP "
END

TO NORTH
OP "CASTLE
END

TO SOUTH
OP "CAVE
END
```

The players must type TRAVEL NORTH (or whichever direction takes their fancy).

This article doesn't purport to be a tutorial on list processing but when we print or

manipulate a list of words that is what we are doing. However, one more aspect of list processing may be helpful to develop our adventure stories further.

Frequently, it is necessary to be carrying a particular article to enable something to happen. Let's put these objects in a box:

```
MAKE "BOX [ ]
```

As you will see, and discover if you type PR :BOX, the box is empty. To put something in the box, we need the following construct:

```
MAKE "BOX LPUT "TORCH :BOX
```

Or we can use:

```
TO PICKUP
PR [You pick up the torch.]
MAKE "BOX LPUT "TORCH :BOX
END
```

To see if we are carrying the torch, we can type PR :BOX or we can use:

```
IF MEMBER? "TORCH :BOX [ PR [ Type
  ENTER to go into the cave. ] ]
```

which, of course, could be an extra line in one of the location procedures.

Magic paper: Logo and the overlay keyboard

M P Doyle
Chairman, BLUG, and teacher of pupils with special needs in Bradford

I have in my hands an ordinary sheet of paper. On it, some words. I put the paper on the tablet. I press it. Rien ce pas! But, I now say the magic word: ckeyson! When I press a word on the paper, piff-paff-puff, it jumps into the computer. That's magic.

The Logo user interface

The school computer insists that logo users 'type a letter' to tell it what to do. This is ill suited to the classroom; neither children nor their teachers have time to develp effective keyboard skills. Moreover, primary school children are only beginning literate; and an alpha-numeric keyboard will overstretch their langauge arts skills. This is a problem in search of a solution.

LCSI, when implementing *LogoWriter*, devised a screen interface and working environment that took a sheet of paper and a schoolbook as a metaphor. This 'classroom intuitive' interface proved effective; and is intuitive to pupil and teacher alike. The need now is to extend this familiar notion to the input device.

A potentially suitable input device had been developed in the UK in 1981, initially for handicapped children. It is now widely used in British schools, mainly on the original 8-bit Acorn/BBC microcomputer. It is known generically as an overlay keyboard, but more commonly as the Concept® Keyboard. Recent developments in electronics have made possible a low-power version, which connects to any serial port.

The overlay keyboard

The overlay keyboard is a flat, touch sensitive keyboard with keys arranged in a rectangular grid. The keys are numbered from 0. It returns a single 8-bit byte to the computer.

The original Concept® keyboard was A4 in size with keys arranged as eight rows of 16. An A3 version followed.

Software is used to assign 'messages to the computer' to the keys. The content of the messages is conveyed to the user by overlaying the keyboard with a sheet of paper, which displays representations of them.

Traditionally, special Concept keyboard editing programs have been used to prepare files of key definitions, which are then used with a special interface program. Some software incorporated Concept keyboard support. All editing programs use a graphic display of the keys to help users in defining them.

The latest development is the Concept 192 with 12 rows of 16 keys. This gives user an aesthetically more pleasing array and slightly better resolution. It is a low-power serial device, cf a mouse.

The Logo primitive

If we are to use this keyboard from within Logo, we need to work with it at language, not graphic, level. In other words, we must be able to talk to it as we would, for example, talk to a Lego TC Logo interface.

The earliest Concept primitive recorded was devised by Jessop, a UK Turtle manufacturer, who incorporated the primitive CONCEPT in turtle driver extension for *Logotron* (ACT/SOLI) BBC Logo. CONCEPT worked like readchar; it waited for a key to be pressed then returned its number.

Recently, LCSI and the author have devised a more comprehensive set of primitives for *LogoWriter*. The core primitive is based on the 'set-' prefix. It sets a key, or list of keys, to a particular word:

setckey [list of key numbers] "word

The messages are stored within Logo, somewhat like words associated with a name. The potential unlocked by this one primitive is immense. The full primitive set is shown in Table 1.

setckey *setckey [1 2 3] word "|hello world| char 13* Assigns a word to concept keyboard keys.
cleackeys Clears all key definitions.
ckey *show ckey 5* Reports the key definition.
ckey? *show ckey? 6* Reports "true if key is defined, else "false.
ckeyson Switches on the Concept keyboard.
ckeysoff Switches off the Concept keyboard.
setcport *setcport 2* Sets the serial (COM) port to be used.
setcbaud *setcbaud 4800* Sets the baud rate.
csend *csend word char 0 char 46* Sends control characters to the Concept.
crecc Reports a single character from the Concept.

Table 1 *LogoWriter/Concept Primitives.*

The overlay

It is tedious to type F O R W A R D hence the Logo abbreviation FD. This unfortunately introduces a element of obscurity. Replacing the QWERTY keyboard with the Concept keyboard enables the same effect to be achieved with enhanced clarity.

The legend on the overlay covering the key matrix may be any graphic device which conveys the meaning of the primitive: an arrow, a drawing of a turtle moving forward, a definition of the primitive showing the inputs required, etc. Many resonances with screen-based graphic user interfaces are immediately apparent; but paper is far more 'open', in terms of the skills required to create and modify it.

It is versatile, informative and leaves the screen clear.

On page 36 (Figure 1), is an overlay for use in early turtle graphics work. It was designed to assist both teacher and pupil.

Specific features to note are:

1. A back and front. Notes for the teacher on the back, a keyboard for the pupil on the front.
2. Each key area has not only the primitive or procedure name in full, but also an explanation of what it does. There is space for the child or teacher to add a drawing or icon as an *aide memoire*. Any input may be indicated and the contraction of the primitive shown in bold type.
3. Primitives and procedures can be grouped according to type. Here, the four commands of turtle geometry are clearly grouped and labelled.

This overlay obviates the need for 'one-key' Logo.

Paper user interface: It will be seen that the combination of one Logo primitive with a very simple keyboard allows us to bring all our established skills with graphics to bear on the design of the interface between the hidden, fluid world of the computer program and the language required to control it. By directly interfacing our older medium with the new one we build a bridge between the two. We know that the 'icons' on the overlay pass messages because we entered the messages ourselves – using setckey.

In use

The overlay primitives in *LogoWriter* make possible a number of activities that were formerly over-complex. A sample of these is

Figure 1 *An overlay for use in early turtle graphics work.*

illustrated below; but we should first comment on a use of the overlay keyboard unique to *LogoWriter*: its capability as a dynamic notepad.

Notes on the fly!

Consider a group of children drawing a house.

First, they want to draw a square. Once they have written the procedure and given it an appropriate name ('square') they may ask a group of keys to remember this name:

```
setkey [1 2 3] "square
```

and then write 'square' (or draw a square) on the overlay. Now, whenever they want to enter 'square' they just press the word (or drawing). There is no pressure to invent obscure abbreviations such as 'sq'.

Next, as they build up the house procedures (triangle, window, etc) they may add these.

They may also add the commands which are needed to join these shapes together to draw a house: the so-called 'interface' commands.

Soon, the overlay might become dotted with: penup, pendown, cleargraphics, left 90, right 120, etc. Superprocedures, as they are developed, may also be added to the overlay.

The overlay has become the children's project record.

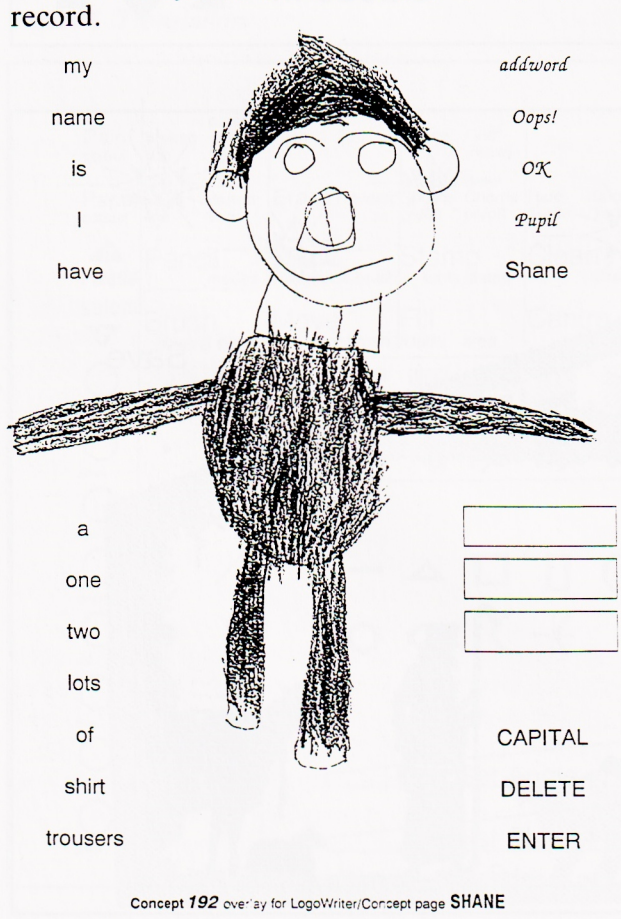


Figure 2 *LogoWriter is a beginning word processor.*

User informer

The other major use of the overlay follows on from the above. It is an interface to a Logo project, or to educational software written in Logo. These overlays inform the user of the vocabulary developed (as user-procedures) and the relationships between vocabulary items, ie to illustrate the structure of the project.

Examples and illustrations

Figures 2–6 give some indication of the potential of this simple yet elegant user-interface.

Words may be input by pressing that word written on the overlay (recognition) or by pressing an illustration (referent). The child's own drawing of himself (Figure 2) is used. Procedures, eg to capitalise the next letter entered, are also accessed from the overlay.

Figure 3 (page 38) shows an interface for a text-relevation toolkit. The structure of the program is clearly represented on the overlay. Characters not easily available at the keyboard have been provided; as are facilities to load and save text. The arrow keys are dynamically re-defined according to whether the text cursor or turtle is exploring the text.

The primitive setckey assigns a message to an area of the Concept keyboard. This is the basis of a program, *Touch Explorer*, which is very widely used in the UK (see example, Figure 4). The graphic may be a map of a village; the messages information on historic events at particular locations.

Using dynamic re-definition of keys using multiple procedure, different levels may be accessed, e.g. different historical periods.

The original *Touch Explorer* is constrained within the parameters set by its programmer. Written in Logo, it is open to elaboration or simplifications by any Logo-literate teacher.

For example, *Touch Explorer* itself does not provide an active cloze procedure mechanism and, as it is coded in BASIC, alteration is problematic. Adding the facility in Logo involved two short procedures.

Figure 5 shows a few simple procedures and an overlay. Children can write Logo in a language other than their mother tongue.

The turtle shapes and turtle graphic primitives (Figure 6) provide ample scope for writing a mathematically oriented graphics package. The overlay is used to provide a 'mouse', selection bars, and access to procedures whilst leaving the screen clear. Without an overlay a project such as this would have been impossible.

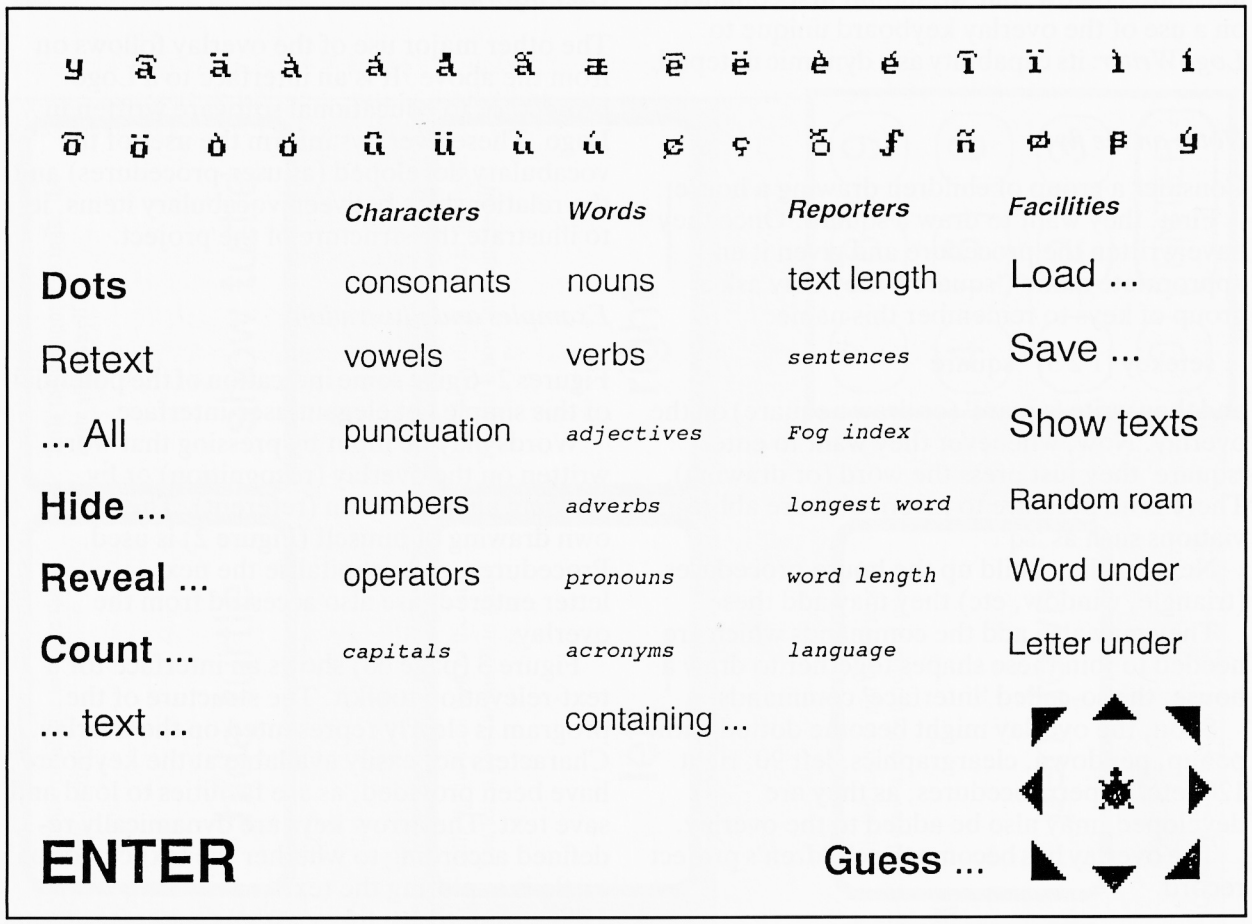


Figure 3 *Beyond word processing*

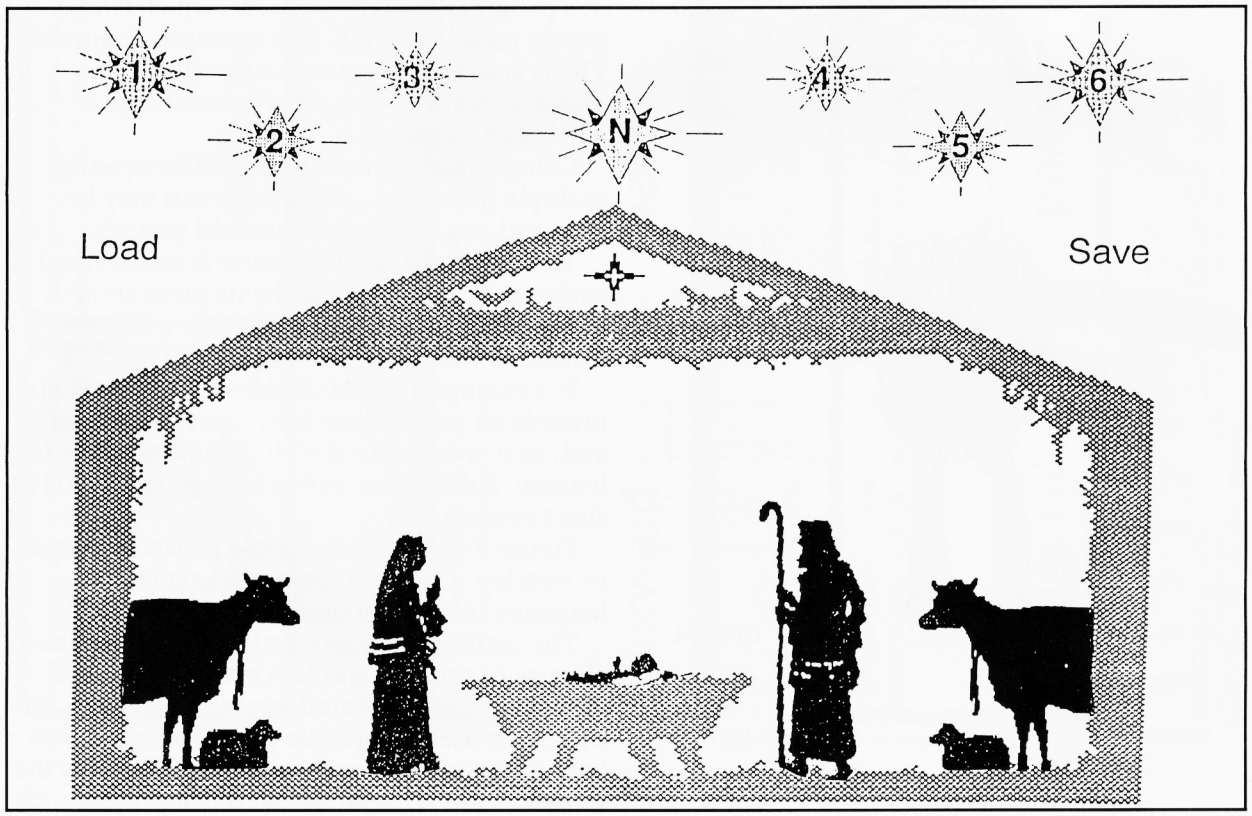


Figure 4 *Touch Exploration*

forward (fd)	avance (av)	adelante (ad)	vooruit (vt)
back (bk)	recule (re)	atras (at)	achteruit (at)
left (lt)	gauche (ga)	izquierda (iz)	links (li)
right (rt)	droite (dr)	derecha (de)	rechts (re)
slowturtle	tortuelente	tortugalenta	traag
fastturtle	tortuerapide	tortugarapida	vlug
(penup) pu	(levecrayon) lc	(sinpluma) sp	penop (po)
(pendown) pd	(baissecrayon) bc	(conpluma) cp	penneer (pn)
(hideturtle) ht	(cachetortue) ct	(escondetortuga) et	(schildpadweg) sw
(showturtle) st	(montretortue) mt	(muestratortuga) mt	(toonschildpad) ts
home	origine	origen	midden
setshape	fixeforme	fijafigura	kiestvorm
setcolour	fixecouleur	fijacolor	kiestkleur
stamp	estampe	estampa	stempel
label	etiquette	rotolo	sticker
clean	nettoie	limpia	schoon

Figure 5 *Translation*

Summary

The original Concept Keyboard was made for a physically handicapped child to program using his nose. As an interface to computer software it has been little explored so far. The Logo/ Overlay combination will enable educators to develop new and powerful tools. *LogoWriter/* Concept is a most important development.

Acknowledgements: Chris Thamm of LCSI who coded an elegant Concept Keyboard extension to Logo. The Concept Keyboard company for producing a new keyboard for the author and Logo: The Concept 192.

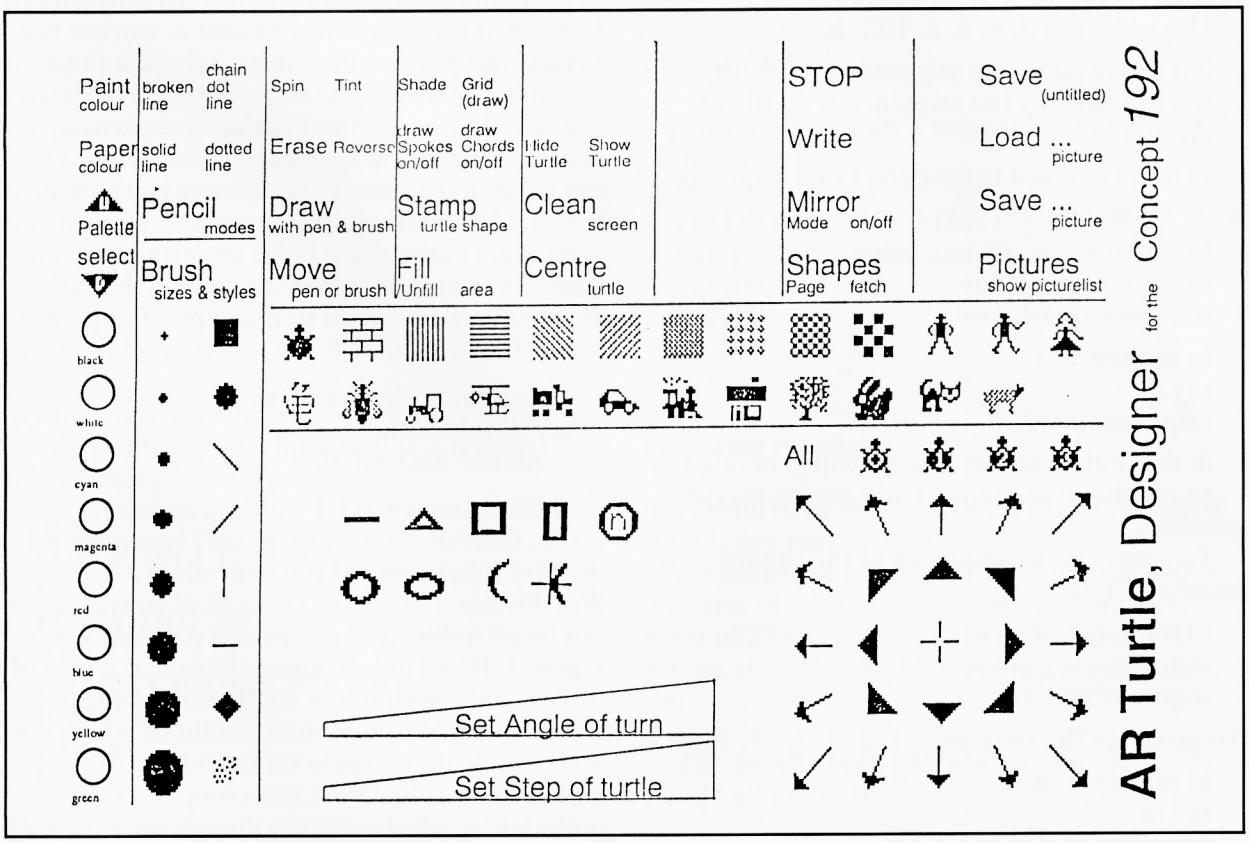


Figure 6 *Model application*

The generation of sequences

Peter Butt
City of London School

The generation of sequences can use one of the most powerful concepts in modern programming, mathematics and education. This concept is: build upon what one already has.

But first, what is required so that pupils can generate a given sequence?

A sequence requires a starting value and a relationship between the successive values.

The relationship specifies how one builds up a sequence from the previous value. For example, the sequence of even numbers, 2, 4, 6, 8,.. has a starting value of 2, and the relationship is n + 2, where n is the value of the previous number.

As is obvious, the 4 is obtained from 2 + 2, the 6 is obtained from 4 + 2. This can lead on to the concept that a sequence can be considered as 'a sequence of sequences', each starting with a different starting value but having the same relationship between the successive starting values.

The sequence 2, 4, 6, 8, 10,.. is:

2 is followed by the sequence 4, 6, 8, 10,..
4 is followed by the sequence 6, 8, 10, 12,..
etc.

This is the concept behind the Logo sequence generating template of:

```
to sequence 'starting_value
say :starting_value
sequence relationship
```

```
to sequence 'n
say :n
sequence :n+2
```

with the initial call sequence 2 will generate the even numbers; sequence 1 will generate odd numbers.

To generate the sequence of the square numbers: 1, 2, 4, 8,..

```
to sequence 'n
say :n*n
sequence :n+1
```

To generate the sequence: 1, 2, 4, 7, 11, 16,..

```
to sequence 'n 'd
say :n
sequence :n+:d :d+1
```

The initial call being: sequence 1 1.

The previous two sequences both start with the numbers 1, 2, 4. To find other sequences which start with these three numbers can be issued as a challenge. Mathematicians know of more than 150 such sequences and 30 sequences starting 1, 2, 4, 8.

The rule for one of the most often quoted sequence examples is: 'produce a sequence of numbers in which the third or any subsequent number is the sum of the previous two numbers'.

This can be generated by:

```
to sequence 'first 'second
say :first
sequence :second :first+:second
```

If the initial call is: sequence 1 1 then the Fibonacci sequence 1, 1, 2, 3, 5, 8, 13,.. is the result. The Fibonacci sequence at school level seems to be one of those things that one 'does' and that maths teachers go on about but is never really used except when considering the number of the ancestors of bees! A worker bee has a single parent, the queen, whereas the queen has two parents, a queen and a worker bee, etc. However, the Fibonacci sequence does crop up in the strangest of places as can be seen from many University maths text books. One worthwhile investigation that computers easily allow to be undertaken at school is to find out what happens to the ratio of the successive terms of this type of sequence, ie 1/1, 1/2, 2/3, 3/5, 5/8

```
to sequence 'first 'second
say :first/:second
sequence :second :first+:second
```

Compare: sequence 1 1 which generates 1, 0.5, 0.666667, 0.6,.. with (say) sequence 8 3 then try other pairs of starting values. Worthwhile, in that the concept of convergence can be introduced or reinforced painlessly, see Figure 1. Being able to appreciate the concept of convergence is vital to a whole range of topics from trial and improvement methods which are used to calculate square roots and the trig functions on calculators and computers, to probability, which answers the question, how often in the long term, will an event happen?

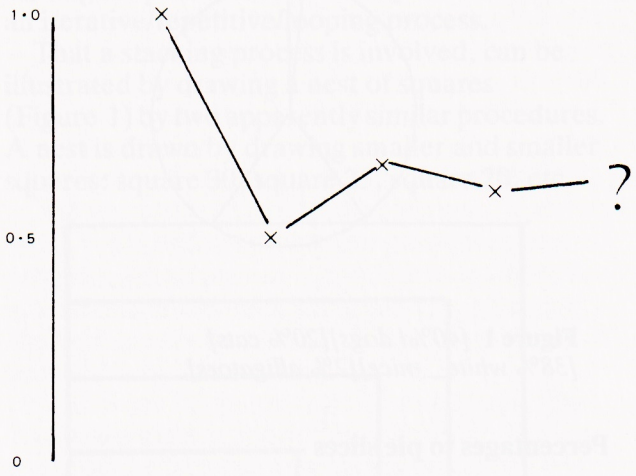


Figure 1 *Diagram showing the convergence of the ratios of the successive terms of the Fibonacci sequence.*

N.B. The procedures in this article are written for Nimbus users. Other Logo users will need to use PRINT instead of SAY and teach the computer:

```
TO SEQUENCE :starting__value
```

and complete the procedure with END

Pie charts

BLUG

In the summer of 1989, the British Logo User Group produced a special supplement to support the new Maths National Curriculum. The following data-plotting procedures from that document are reproduced here for those who missed them the first time around.

The circular outline

To draw a circle whose centre is the current Turtle position and whose radius is 50 units, use this startup procedure:

```
TO SET__UP
MAKE "PI 3.141592654
CS
FORWARD 200
RIGHT 90
REPEAT 100 [FD :PI * 4 RT 3.6]
LEFT 90
BK 200
END
```

This will also draw a radius upward from the centre; this will serve as a starting point for the pie chart.

The segments

A procedure to draw a segment containing N% of the circle is:

```
TO DRAW__SEGMENT :percentage
RIGHT 3.6 * :percentage
FD 200
BK 200
END
```

Preparing raw data

It is useful to be able to convert from raw data to cumulative percentages.

We suggest a representation for large sets in the form of a list of number-identifier pairs. A set of 20 dogs, 10 cats, 19 white mice and an alligator would be represented by the list:

```
[[20 dogs][10 cats][19 white__mice]
  [1 alligators]]
```

To find the total number of objects in the set, use this procedure:

```
TO TOTAL_OF :set
IF EMPTY? :set [OUTPUT 0]
OUTPUT (FIRST FIRST :set) +
  TOTAL_OF BUTFIRST :set
END
```

Expressing data as percentages

You can express the number of dogs, cats etc as percentages of the total:

```
TO PERCENTAGES :set :symbol
OUTPUT PERCENTAGES_1 :set
  TOTAL_OF :set :symbol
END

TO PERCENTAGES_1 :set :total :symbol
IF EMPTY? :set [OUTPUT [ ]]
OUTPUT FPUT LIST WORD((100 *
  (FIRST FIRST :set))/:total :symbol LAST
  FIRST :set PERCENTAGES_1
  BUTFIRST :set :total :symbol
END
```

This will convert from raw data to percentages. For instance:

```
PERCENTAGES [[20 dogs][10 cats][19
white_ mice][1 alligators]]"%
```

will return the result:

```
[[40%/ dogs][20% cats][38%
  white_mice][2% alligators]]
```

Figure 1 *[40%/ dogs][20% cats] [38% white_mice][2% alligators]*

Percentages to pie slices

Writing a procedure to express these percentages as a pie chart is thus straightforward:

```
TO DRAW_CHART :set
SET_UP
DRAW_SEGMENTS PERCENTAGES
  :set "
PRINT PERCENTAGES :set "%
END

TO DRAW_SEGMENTS :pc_set
IF EMPTY? :pc_set [STOP]
DRAW_SEGMENTS FIRST FIRST
  :pc_set
DRAW_SEGMENTS BUTFIRST :pc_set
END
```

Building upon what one already has – recursion

Peter Butt
City of London School

The concept of 'building upon what one already has' is simple and powerful. Unfortunately, in computing, this concept has traditionally been introduced far too late, often as the last topic instead of one of the first, in a programming course. As such, many students, whether children or adults, are unable to comprehend that recursion is actually possible. The reason would appear to be that they encounter difficulty trying to build upon the less powerful models of the Logo 'REPEAT' or BASIC 'FOR-loop' constructs. Recursion is a 'powerful idea', and as such it is worth the time and effort required to become familiar with it. Recursion can even become a way of looking at our lives for surely we try to learn from (build upon) our previous experiences.

As a definition, a recursive procedure uses itself as a sub-procedure.

The concept to accept is that recursion or 'building upon what one already has' is a stacking, followed by an unstacking, process. And essentially, this what actually happens to the machine code generated at run-time within the machine. Each timne a procedure is called, the salient facts are added to the code within the

computer's memory, to be removed again after leaving the procedure. Certainly recursion is not an iterative/repetitive/looping process.

That a stacking process is involved, can be illustrated by drawing a nest of squares (Figure 1) by two apparently similar procedures. A nest is drawn by drawing smaller and smaller squares: square 30, square 25, square 20, etc.

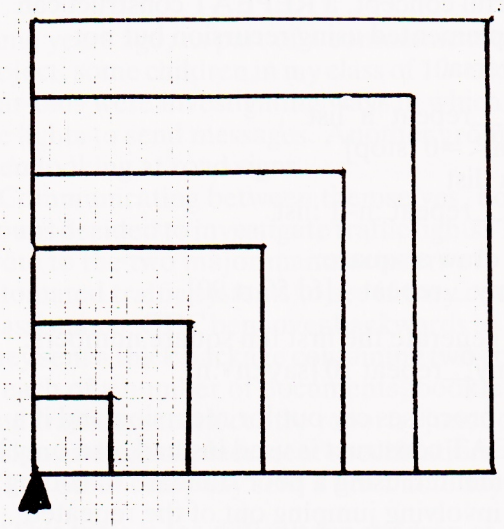


Figure 1 *The stacking process.*

Generalising and using the variable 'S for the length of the side of the square, simplifies to procedure 1, which can be called by (say) nest 30.

```
Procedure 1  nest 'S
             square :S
             nest :S–5
```

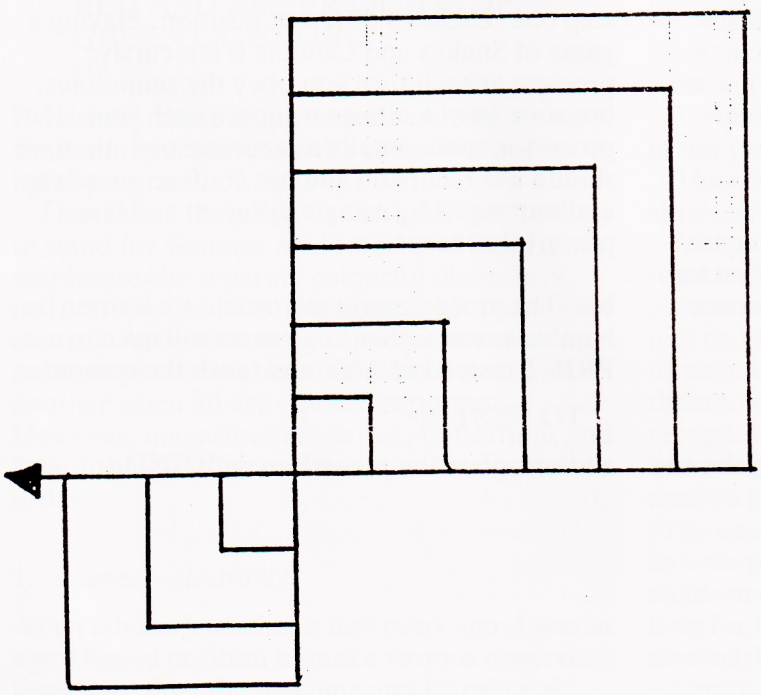


Figure 2 *Squares with negative lengths.*

This procedure will overshoot, for squares having negative lengths will also be drawn (Figure 2). That this happens, could be a bonus teaching point when introducing directed numbers.

To stop the negative squares being drawn, a conditional statement has to be introduced, to test when the next square to be drawn would have sides less than (say) 5 units, procedure 2.

```
Procedure 2  nest 'S
             if :S<5 [stop]
             square :S
             nest :S–5
```

Using procedure 2, Figure 3 shows nest 30 after the drawing of the first three squares, with sides 30, 25, 20.

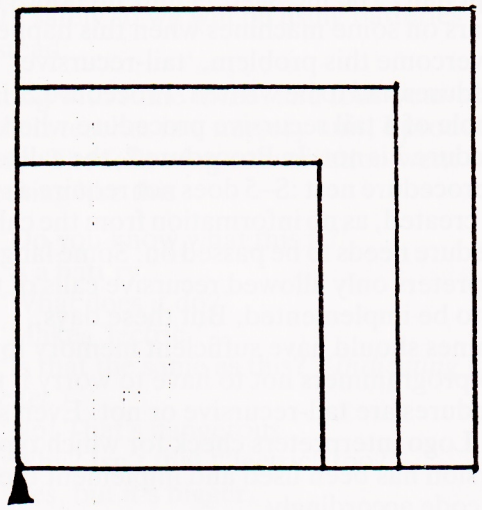


Figure 3 *Squares with sides 30, 25, 20.*

An alternative nest procedure is given by procedure 3.

```
Procedure 3  nest 'S
             if :S<5 [stop]
             nest :S–5
             square :S
```

With an initial call of nest 30, the first three squares to be drawn by procedure 3, are the smallest ones 5, 10, 15, drawn during the unstacking process (Figure 4).

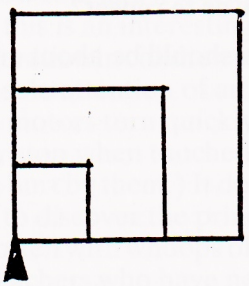


Figure 4 *Squares with sides 5, 10 15.*

To really demonstrate that a stacking and unstacking process does occur, combine 2 and 3 as in procedure 4.

```
Procedure 4  nest 'S
             if :S<5 [setpencolour pc–1 stop]
             square :S
             nest :S–5
             square :S
```

Unfortunately, at this stage, the red-herring of 'tail-recursion' may still need to be mentioned. A few years ago when machines with only a small amount of memory were available, using recursion to any reasonable depth/height of stacking caused all the computer's memory to be filled up and hence the process came to an abrupt halt. The message, 'Panic no nodes left' appears on some machines when this happens. To overcome this problem, 'tail-recursive' procedures had to be written. Procedure 2, is an example of a tail recursive procedure whereas Procedure 3 is not. In Procedure 2, the call to the sub-procedure nest :S–5 does not require a stack to be created, as no information from the calling procedure needs to be passed on. Some language interpreters only allowed recursive calls of this type to be implemented. But these days, machines should have sufficient memory for Logo programmers not to have to worry if their procedures are tail-recursive or not. Even so, good Logo interpreters check for which type of recursion has been used and implement the run time code accordingly.

Another example to demonstrate the stacking and unstacking involved with a recursive process is to count-down.

```
count down 'n
if :n < 0 [stop]
say :n
count_down :n-1
say :n
```

count_down 4 should result in the sequence 4, 3, 2, 1, 0, 1, 2, 3, 4. Removing the first say :n statement, gives a procedure that reverses a set of numbers.

To recurse or to repeat?

Education should be about encouraging the use of powerful concepts. Recursive procedures contain a call to themselves within their own procedure body. The call to themselves requires the making of a 'generalisation' and the encouragement of the making of generalisations is what much of a school education is all about.

To emphasise that recursion is the more powerful concept, a REPEAT construct can be implemented using recursion but not vice-versa.

```
my_repeat 'n 'list
if :n<=0 [stop]
run :list
my_repeat :n–1 :list
```

To draw a square:

```
my_repeat 4 [fd 50 rt 90]
```

To generate the first ten square numbers:

```
my_repeat 10 [say :n*:n]
```

Some processes cry out for recursion and if a REPEAT construct is used instead, can only be implemented using a poor standard of coding, often involving jumping out of the repeated code. A classic example is when searching text for the number of occurrences of a given word. Before the search starts, we do not know how many occurrences of the word there are, so the repeat variable cannot be set.

Having implied that I view life as a recursive process, I will finish by stating that the only repeating processes that I know are ones that just waste time, such as idly bouncing a ball. Climbing stairs is a recursive process for at each step one reaches a different position. Playing a game of Snakes and Ladders is a recursive process; at each turn you obey the same rules, but start from a different square each time. If a process is conceptually a recursive one, then we should use recursion and not confuse ourselves and our pupils by using a different and less powerful concept.

NB The procedures in this article are written for Nimbus users. Other Logo users will need to use PRINT instead of SAY and teach the computer

```
TO NEST :S
```

and complete the procedure with END.

SALAD technology

Chris Robinson
BLUG and Deputy Head, Horndean Middle School, Hampshire

Some years ago, as part of a communications project, some children in my class of 10- and 11-year-olds were investigating ways in which we use lights to send messages. Another group had been looking at road signs.

Communicating between themselves, both groups decided to investigate traffic lights. They wrote to the two major manufacturers of automated traffic systems to see if they could be of assistance. GEC bent over backwards to help. They sent a thick package containing two copies of each of a number of documents, booklets, road traffic research centre reports, junction design and signal placement specifications etc.

The other company sent a brief note saying:

> **'This topic is too difficult for children of this age!'**

Meanwhile, lesser processors on street corners were relentlessly carrying out the instructions:

```
WHEN CARS DETECTED, DO THIS –
  DELAY FOR INTERVAL 1
  AMBER LIGHT ON
  DELAY FOR INTERVAL 2
  RED AND AMBER LIGHTS OFF
  GREEN LIGHT ON
  etc
```

Eventually those children were able to use very similar instructions to write their own traffic lights program in Logo.

I have used the acronym 'SALAD' technology to stand for '**S**cience **A**nd **L**ogo **A**nd **D**esign', to emphasise the separate colourful disciplines integrated within the nutritious field of computer control. An unfortunate consequence of using an acronym means one discipline is put before another when all are equally important. However, magazines are serial publications and I must thus address the disciplines in a serial order.

1. *Science – electricity*

As an advisory teacher a few years ago, I was in a privileged position to make various observations regarding the teaching and learning of essential concepts like 'electricity'.

A class of 13-year-olds were being introduced to electricity. I do not remember the exact words of the teacher as he stood before the blackboard but they went something like this:

> 'Electricity is made in power stations and is delivered by cables to our houses. This mains electricity has too great a voltage to be played with safely so we will be using batteries instead. . . .'

Sitting with some three- and four-year-olds and their teacher in a nursery unit, I produced a battery. The following conversation went something like this :

> 'Do you know what this is?'
> ''s a batt'ry.'
> 'What does it do?'
> 'Makes 'lectric.'
> 'Is that the same as this?' *(Indicating wall socket.)*
> 'No! That's dangerous.'
> 'Why? Isn't that "'lectric" too?'
> 'Yes, but it's bigger.'

They went on to tell me mains 'lectric was made somewhere else and came to homes and schools through pipes (cables). (I ascertained from their teacher that they had never discussed electricity before.)

I wonder what they will be learning at school in ten years' time?

How open minded do we encourage our children to be?

When I introduce 'electricity' to children (or teachers) for the first time (at whatever age), I always use the same discovery approach: give one child a battery and another a 'device' like a torch bulb, LED or preferably a motor and leave them to discover how to make it work. For reception children this is an interesting exercise in developing cooperation in children previously used to having the sole attention of an adult. (The small electric motors turn quickly but with so little power they stop when touched and children will not be hurt by them.) It doesn't take long for the pupils to discover the principles of an electric circuit (often with whoops of pleasure – especially from teachers who have never done this before).

Now comes the paradox. If I ask reception children if they can make the motor go the other way, they look at me as if I'm stupid and reverse the wires. Ask 'new-to-electricity' eleven-year-olds if they can make it go the other way and they find it difficult. Is it a fear of being 'wrong' or do they think it's a trick question?.

School curricula should contain a progression of 'discoveries' about electric circuits from simple serial circuits to complex parallel ones containing different switches, sensing devices etc.

In my school we light model rooms in the early years with a single bulb and home-made switch. Later we light shoebox theatres with more lights. We even make 'electronic maps' as will be found in town centres whereby buttons pressed cause LEDs to illuminate that place of interest on a town plan. We also design and make burglar alarms for our model houses featuring more home-made switches. (Later we also use 'Micro-electronics For All' kits with their AND and OR gates).

2. *Design technology*

When children are faced with designing a system, artefact or environment to help meet a perceived human need, they can only bring to it those experiences that have already helped shape the thought processes.

A class asked to design a house or room for a paraplegic came up with some very good ideas like automatic opening doors but had no idea how one could be made.

There should be a progression from 'junk modelling' via guided experience of resistant and non-resistant materials and tools to work them and with them.

As this knowledge is acquired, more realistic designs can be produced and a wider experience will permit greater comparision with what else may be achievable making evaluation more productive. Children will see 'back to the drawing board' as being a worthwhile step in the learning process rather than a defeat.

3. *Programming*

Computer control requires more of the computer than just a sophisticated switch; it must respond to a sequence of commands with or without input from remote external devices.

To provide a list of instructions, some kind of program needs to be written. There are many 'Logo-like' interpreter programs that will permit children to enter a (limited) set of commands to be interpreted by that program, to be passed on to the computer's own internal interpreter, to perform the (relatively simple) switching operations microchips manage well.

Apart from learning that control language, the children may also have to learn many other computer pseudo (perhaps also 'Logo-like') languages to draw shapes, write stories, handle data etc. Why not use one common language with one set of constructs and syntax to learn and which permits the definition of new words as determined by the children? Such a language is control Logo.

As with science and technology, there should be an identified progression that will enable pupils to attain a level of fluency in the programming language equivalent to their developed scientific and technological skills.

For example:

1. Early years children learn turtle language through games directing them and their peers around obstacles.
2. The next move is to use floor turtles or programmable toys like Pip or Roamer.
3. Turtle commands are further developed with the screen turtle.
4. A progression of mathematical skills going hand in hand with Logo language development follows. (In my school this goes from 'free play', through regular polygons, bricks and walls. There is work on a scale using global variables and passed parameters. There is also work on coordinates and bearings.)
5. Next there can be an expansion of the offered Logo commands and possibilities through languages uses: electronic magazines, personal databases, adventure stories, etc.
6. Further progression to operate torch bulbs or LEDs directly and from programs to emulate lighthouses or other flashing lights for example.
7. Monitoring and recording of external inputs from remote switches or analog signals from light or temperature sensors etc follows. Building upon previous knowledge, it can be possible to have turtle-drawn graphs of analog inputs.
8. Combining inputs and outputs, controlled by the Logo language, complete systems may be built – including traffic lights that respond to the approach of toy cars!

And yet I have overheard teachers saying 'Oh, we've done Logo.'!

Logo: where next

M P Doyle
Chairman, BLUG, and teacher of pupils with special needs in Bradford

Logo in the UK has suffered from the degenerative effects of isolation. A promising start at Edinburgh University and an innovative implementation on the RM Nimbus is dying in infancy because it has been abandoned by its mother (the Artificial Intelligence Applications Institute at Edinburgh University which no longer works on Logo), because its father (Research Machines) will not pay maintenance. No one else can help poor Nimbus Logo – it only runs on RM computers and nowhere else in the world uses them. Logotron Logo was imported from France; its originators fared little better and no longer exist. The UK Logo world has also suffered on the human level. Paul Crowe, the brains behind LSL Logo, took his own life.

Across the Channel

In Europe and America, during the same period, there has been a variety of innovative approaches to Logo, and some first class research. Some of this is catalogued in the first issue of the new European Logo newsletter/journal, *EUROLOGOS*. I will list a few of the high points here.

Logo and 'Apprentisages'

Research into Logo and learning has been carried out in numerous European countries. One example is Belgium, in Ghent and Liège. At Ghent, Martin Valke and Gilberte Schuyten have developed the relationship of Logo with constructivist eductational thought. Whilst at Liège, Brigitte Denis's detailed study of Logo users over a 10-year period has pointed to the very practical problems in both teaching and learning to use the computer language. To me her most interesting observation was that children did not read error messages on the screen. Does this mean that they do not perceive the computer as a machine to converse with, or were the error messages intrinsically unhelpful? Other groups have tackled different aspects. For instance, both the Geneva and Leuven groups have delved deeply into the cognitive effects of Logo, and have thereby cleared up some of the early controversies in this area. The major UK contribution, from Celia Hoyles and Richard Noss in London, has been to the development of mathematical uses of Logo – based largely on turtle geometry.

Language development

Logo development has taken place in several European countries.

The Netherlands Logo Centre under Harry Pinxteren developed a new implementation of Logo based on Scheme. Herbert Loethe, the author of the first German Logo, has added turtle graphics to Scheme to provide a more powerful mathematical environment for secondary school and university students.

Several workers, including Christian Wagenknecht in (East) Germany, have experimented with object-oriented programming within Logo. Others, including Enrico Fischetti and Antonio Gisolfi in Salerno, have devised full object-oriented implementations of Logo.

Developments in the application of Logo have taken place, patchily, throughout Europe from the Nordic countries to the Mediterranean, from Eire to the former Soviet Union.

Extension to Logo

The extension of Logo to new domains – new topics of conversation – have also been seen. In Portugal, Veloso has devised an extension to IBM Logo, Logo.Geometria, which makes work with Euclidean geometry more easy. In the UK we have Dave Pratt's quasi-Newtonian extension. And there have also been a number of developments which provide utility extensions, e.g. for drawing graphs, and interface-related additions, for example to talk to a mouse or windows. The most recent in this vein is WINLOGO, a new Logo implementation from Spain which uses a windowing system.

But the jewel in the crown of European Logo is, without doubt, Boris Sendov's '**Plane Geometry System**', from Bulgaria. This introduces the primitives **point, line** and **object** (a geometrical object). The screen is a view onto

infinite straight lines, bisections, triangles, circles and tangents, which can be rotated and otherwise transformed by using computer language. This is Euclidean geometry done in procedural notation. For one who has suffered Pythagoras at school it was a revelation. I defy anyone, having seen this, to think of Logo-mathematics in terms of turtles ever again.

Transatlantic events

The major centre for Logo development has been North America. The MIT Logo team no longer exists as a separate entity; it is now one aspect of the Media Lab. Here Seymour Papert, known to some as 'the father of Logo' (its 'mother' was probably Cynthia Solomon), continues to work with Logo as Lego Professor of Education. He is also Chairman of Logo Computer Systems Inc, based in Montréal (Québec). LCSI continue to be the most innovative of Logo developers.

In the mid-1980s, on the inspiration of Brian Silverman, they extended Logo vocabulary to provide a word processor you could talk to. This was *LogoWriter*. Logo vocabulary now included **cut, copy, paste, search, found?** and so on. *LogoWriter* was also unique in that it had a user interface based on the metaphor of a sheet of paper and a scrapbook (cf a desktop and file manager).

LogoWriter, in turn, formed the basis of Lego TC Logo. This is the implementation of Control Logo sold by Lego throughout the world (except the UK). Lego TC Logo differs from the Control Logo we are used to in the UK because it treats inputs and outputs as 'turtles'. What do I mean by this? Here is an example:

In Julian Pixton's *Control Logo* for Longtron Logo, to switch on a light connected to output number 1, we write: **turnon 1**; in Lego TC Logo we write: **talkto 1 on**.

That is, we tell the computer which port to talk to, then what to do to it. This just like **tell 1 forward 50** when we are using several turtles.

The next development was to extend Logo vocabulary to telecommunications. This product was called *LogoExpress*. Logo vocabulary now included **terminal, send, receive?** and so on.

Most recently LCSI have added database primitives. (Logo has always had property lists.) These include **record, field, getdata, putdata**, and so on.

On the input side, there is the extension to the Concept keyboard, adding **setckey, ckey, ckey?, clearckeys**, etc.

From application to object

Thus we now have a range of vocabulary which relates to actions we can carry out using a computer. The concrete turtle now has new computer-concrete companions. We can talk to (or rather correspond with) a word processor, interface box, modem and e-mail system, database, etc. The next step is, perhaps, for Logo to 'know' about these applications as applications. Let me explain: *LogoEnsemble* is somewhat like an integrated software package. (*Claris Works* for the Apple Mac might be a good example). Within this integrated environment you can swap from turtle geometry to procedure definition to report writing to data processing and back. What *LogoEnsemble* does not know about are things like graphs and word processors.

In object-oriented programming, the language does know about entities such as graphs and text boxes (remember *BOXER* from MIT?). These objects are a bit like the turtle; they have specific characteristics yet inherit common characteristics from the computer environment. For instance, in *LogoWriter* both the turtle colour and text colour are changed using the same underlying mechanism, whilst their positioning on the screen uses totally different mechanisms.

In such an Object-Logo (OLOGO) these applications-objects would be 'hatched' like multiple turtles were in Acornsoft Logo. Let us say that we create a graph by hatching one. What might this entail? Well, there would need to be a screen display, perhaps of a piece of graph paper. Then we would need vocabulary with which to talk to the graph: **setxaxis, setyaxis, setzaxis**, for instance. The graph-object might, perhaps, know about data acquisition so that data from a sensor could be plotted as it was input. We might hatch multiple graphs, so our Logo might look something like this: **hatch "graph "tea plot "tea input 1, hatch "graph "lolly plot "lolly input 2**, which might be a way of plotting one graph of a cup of tea cooling down and another, in parallel, plotting an ice lolly melting.

How object-oriented Logo will develop is, as yet, uncertain. However, as we saw above, this is a major focus of Logo research; and some of the ground rules have already been established. Object-Logo from Paradigm in the USA is the first, rather academic, commercial application. I would expect to see more school-oriented developments within the year.

And the computer

Logo will also need to develop vocabulary to communicate with new additions to computer technology. This includes new storage media, input/output channels, and data types.

Taking the first and the last together, one example might be an extension to CD-ROM. Logo already has primitives to change disc drives, handle directories, load text or picture files. Extension to CD-ROM would involve not merely an extension to deal with CD-ROM hardware, but Logo would also have to know about pictures other than its own graphics. (Here I mean that **Logo** would need to know about such things, not merely to have access to a particular machine's operating system). Additional data types would include scanned images, video and digitised sound. There would also need to be some hypertext tools to aid the search for information and its linking together.

On the input/output side, we already have an extension to the overlay keyboard and to single switches for the handicapped. Future additions might include speech input and voice output.

Learning

If we are to take advantage of these new developments in Logo we need to take two important steps.

1. We need to accept that language is the most powerful of human capabilities and that working with a computer through language is to work with it in its native modality.
2. We need to accept that there is a learning overhead involved.

New languages associated with technology we barely comprehend are not the easiest of subjects to master, particularly for middle aged teachers. But, as teachers will tell you, some hard things are well worth mastering. Logo is not going to go away. . . .

Finally, please learn the following by heart:

Turtle Geometry is a new, computational, relative geometry. It has its own notation: Turtle Procedure Notation (TPN). Compare this with algebraic coordinate notation:

A Circle

	Algebraic equation
to circle repeat forever forward 1 right 1	$x^2 + y^2 = r^2$

Turtle graphics is something entirely different. The word **turtle** is no longer a precise descriptor for a branch of mathematics; it is a metaphor. (Confusing, isn't it?) A turtle is now an object that can roam the screen as you or I roam the world; sometimes using a map, sometimes consulting a compass and sometimes just wandering along turning left and right.

Logo is a computer language with a vocabulary and grammar. It is a good language for doing turtle graphics in. It was used to develop turtle geometry, but there are other, more powerful languages which advanced students might find better for this work.

MICRO-SCOPE

Published by Castlefield (Publishers) Ltd,
Newton Close, Park Farm Industrial Estate,
Wellingborough NN8 6UW

NEWMAN COLLEGE with MAPE