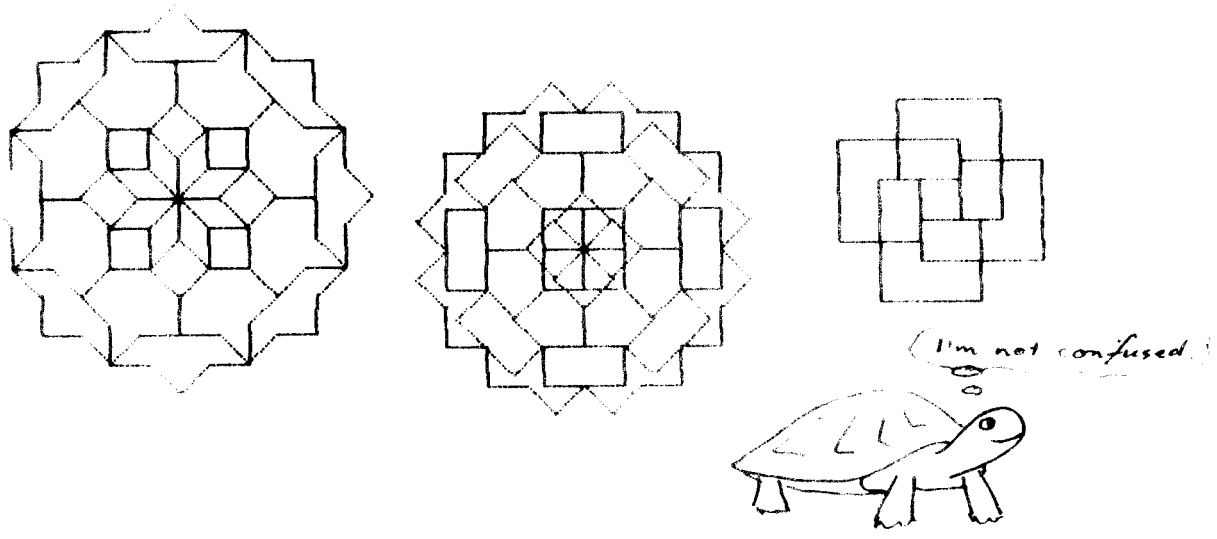
# POALL

## A Journal for Logo Users




## Conference Reports

**Volume 3 Number 4 November 1988**

This issue is largely devoted to items from two conferences, ACEC '88 in Perth, and the first state conference of the Computers in Education Group of South Australia, October 28..30th. We have brief reports of the ACEC papers, and the Logo paper presented (?) at the CEGSA conference. LEGO/Logo was much in evidence at both events.

As well, we have an article on the development of some list processing functions by Peter Mitchell, a review of *Turtle Confusion*, one of the most intriguing Logo books ever published, and a few other things.

By rights, there ought to be another issue by the end of the year. We'll need another 15 or so pages to make it.

Peter

**POTS**


2 Papert's Vision
8 ACEC '88
11 Turtle Confusion
13 Jumbled Letters
17 The MIDI Turtle
18 Dirty Tricks Department Revisited
19 Computing at Entropy House
20 The Back Page

## Papert's Vision: a promise fulfilled?

> 'In my vision, *the child programs the computer* and, in doing so, both aquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building.' (*Mindstorms*, p 5)

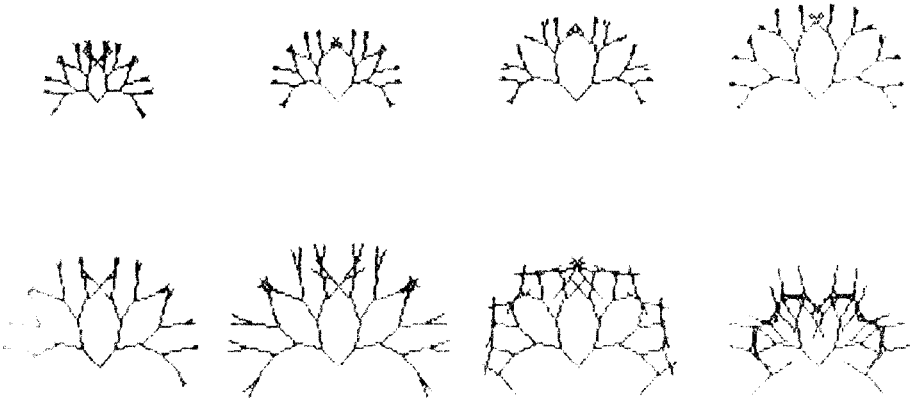
That is the promise of Logo, children learning, not by being 'taught' but by exploring the ideas for themselves, by dissecting and reassembling, in the form of Logo procedures, problems in science, mathematics, language, art, music, the whole gamut of intellectual activity. But has the promise been fulfilled? Can and do students learn these things with Logo? What is the current state of Logo and its results?

Let's start with brief looks at some explorations with Logo, beginning with science. Biology may seem to be an unusual topic for Logo, but two examples. The first is based on Richard Dawkins' program described in *The Blind Watchmaker*, a cogent explanation of evolution, 'the only known theory that could, in principle, solve the mystery of our existence.'

The program is built around a fairly simple Logo procedure (Carter, 1987):

```
TO Grow :genes
IF EMPTY? :genes [STOP]
LEFT FIRST BUTFIRST :genes
FORWARD FIRST :genes
Grow BUTFIRST BUTFIRST :genes
BACK FIRST :genes
RIGHT 2 * FIRST BUTFIRST :genes
FORWARD FIRST :genes
Grow BUTFIRST BUTFIRST :genes
BACK FIRST :genes
LEFT FIRST BUTFIRST :genes
END
```

As you may gather by the variable name, :genes, the procedure is given a list of values which can be selected and mutated by other procedures. As the generations proceed so the 'organisms' evolve. The picture below shows something of what can happen through eight generations, beginning at top left. Students can play with evolution at first hand. (If we are built by cells following genetic information, does that mean that we are simply information processing machines?)



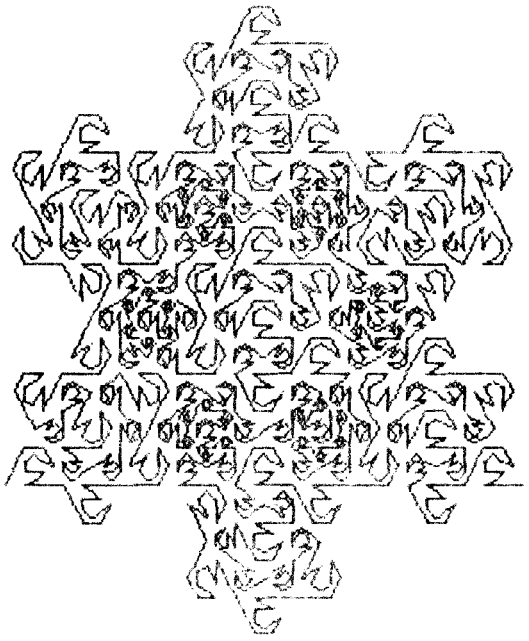
The second example may have particular meaning as we face the AIDS problem. What is a virus? How can we deal with them? Again, the Logo looks innocent enough:

```
TO Virus.0 :generation
PR SE [This is] WORD "Virus. :generation
POTS
DEFINE WORD "Virus. :generation + 1 TEXT WORD "Virus. :generation
ER WORD "Virus. :generation
RUN LIST WORD "Virus. :generation + 1 :generation + 1
END
```

What is really doing the work, the procedure itself, or the Logo interpreter acting on some data? There is a real issue here. What *is* it that is replicating? Surely the procedure, a few lines of code, cannot be making new copies of itself can it? To go a bit further, would it be possible to make a machine that could build new copies of itself, in other words, is self-replication possible for machines? (Some would argue that plants and animals, including ourselves, are (biological) machines, and that that would answer the question.) In the 1940s and 1950s John von Neumann argued that it was logically possible, and even proposed self-reproducing machines to explore the universe. He and Stan Ulam devised computer 'games' that generated self replicating patterns, games which evolved into John Conway's 'Life'. To explore that, I would recommend Brian Silverman's *Phantom Fish Tank*, a Turtle-less version of Logo for working with cellular automata.

What about an antibody to the virus?

Let's move to mathematics. In *Mindstorms*, Papert (1980) writes of 'mathophobia', the intellectual disease which prevents people learning mathematics (and other subjects). As a mathophobe myself, I have found in Logo a way to approach many things that were formerly on forbidden territory. Mathematics, at least in the form of logic, is inherent in everything in Logo. But one example here. What Mandelbrot has named fractals were once in the 'too hard basket' of mathematics, the 'monsters'. We can now readily play with them in Logo:



I leave it to you to write the procedure.

Perhaps a few words about Computing Studies, since Logo is an 'approved' language are in order. What is it we are trying to teach in Computing Studies? What has word processing to do with computing, or databases, or spreadsheets? Who cares about what goes on inside microprocessors? Why the emphasis on algorithm design and structured programming? Why teach Pascal, full of syntax and semicolons, in schools, when Logo is more powerful and easier to write and use? Where does Logo fit into this? One view is expressed by Brian Harvey in his *Computer Science Logo Style* series, where he describes a Pascal compiler written in Logo. My own view is in *Thinking Logo*, from which I take one example:

```
TO ANDgate :a :b
OP IF AND :a = 1 :b = 1 [1][0]
END

TO XORgate :a :b
OP IF OR (AND :a = 0 :b = 0)(AND :a = 1 :b = 1) [0][1]
END

TO HalfAdder :addend :augend
OP WORD Andgate :addend :augend XORgate :addend :augend
END
```

I suspect that Weizenbaum (1976) was right when he wrote of computer science graduates as being '... rather like people who have somehow become eloquent in some foreign language, but who, when they attempt to write something in that language, find they have literally nothing to say.'(p 278) We ought to be emphasising ideas, and their exploration and expression, not merely techniques and technicalities.

What are the 'deepest ideas from science, from mathematics...'? Are they simply things like Newton's laws of motion, Cartesian coordinates, and the like, or is there more? I would suggest that working with Logo involves immersion in the 'scientific method'. Perhaps more than science itself, as it is presently taught, Logo involves more of the hypothesising, testing, verifying and falsifying than takes place in the laboratory. A Logo procedure is simply an expression of an hypothesis, a possible way of solving a problem. If it works, fine; if it doesn't, the hypothesis must be revised or rejected. Popper, Lakatos and Kuhn are the people who wrote about this sort of thing.

Perhaps the boldest claim by Papert is the one about 'intellectual model building', and the transfer of learning and skills to other subjects. It is the assertion most discussed and disputed by researchers and writers, and any attempt to survey the field in this brief paper is doomed to failure. Essentially, however, there seem to be two camps, one critical, the other supportive. The chief critic would appear to be Roy Pea, of the Bank Street College of Education. Papert himself, and writers like Harvey, Lawler, Solomon and Weir represent the advocates.

In a paper presented at ACEC '88, Wing Au of the University of Newcastle presented the findings of his own research (Au and Leung, 1988), commenting 'It would appear that many researchers have failed to take into account the type of teacher intervention provided in a Logo environment.' I have always been uneasy with research based on a month or so of introductory Logo, followed by tests related to an entirely different subject. I doubt if Papert (following Piaget) ever believed that Logo was anything other than a long term proposition, needing years of use by a student to develop the real benefits. In contrast to others, Au and Leung carefully describe their intervention, ie. teaching, and their testing, and report beneficial effects of Logo. They also stress the important social aspect of learning with Logo, the need for students to be able to freely discuss what they are doing and share ideas, and the need for the teacher to take an active role.

For those interested in pursuing this aspect, the following papers may prove useful:

Au, W. and Leung, J. 'Transfer of Problem Solving in Logo Programming' in Alp, P. (Ed) *Golden Opportunities* ECAWA, 1986

Clements, D. and Gullo, D. 'Effects of Computer Programming on Young Children's Cognition' in *Journal of Educational Psychology* Vol 76, No 6, 1985

Fay, A. and Mayer, R. 'Children's Naive Conceptions and Confusions About Logo Graphics Commands' in *Journal of Educational Psychology* Vol 79, No 3, 1987

Mayer, R. and Fay, A. 'A Chain of Cognitive Changes With Learning to Program in Logo' in *Journal of Educational Psychology* Vol 79, No 3, 1987

McMillan, B. 'Logo and the Teaching of Logo: a Piagetian perspective' in Hancock, J. (Ed) *Tomorrow's Technology Today* CEGSA, 1987

The main papers by Pea are included in the book *Mirrors of Minds* listed below, and of course the list is far from exhaustive. What is the overall view? I think it tends to be positive, and certainly people like Sylvia Weir and Robert Lawler (see below) report positive benefits.

Logo has been described by some as the 'new Latin'. In contrast to Latin, Logo is a living language, with an active culture. It is more than just grammar, it has the power to make things happen: Turtles move, LEGO machines operate, language is processed, music plays, the whole cathedral of computing is active. We may not yet have classical Logo literature, but we do have lively, social learning environments. I don't remember that from LEGAMVS.

About Logo there has grown a number of myths: that it is a 'toy' for little children, that it is just another language like Pascal or BASIC, and so on. The truth is that it is a very effective system in the right hands, a powerful language in a powerful culture. Logo is not easy, it is intellectually demanding; it would have no point otherwise. You can learn the essentials of a wordprocessor or package like *Print Shop* in an hour or so; it will probably take a year for you to be sufficiently experienced and aware to be able to begin to use Logo effectively. But the rewards are much greater.

Logo itself is changing. If you've only ever seen the original Apple Logo or Commodore Logo, take a look at LogoWriter and LEGO TC Logo. These versions work in a much more natural environment, and in the case of LogoWriter come with a range of excellent teaching materials. LEGO Logo is a whole new field in itself.

Let me conclude with a brief list of some recent or significant books:

Abelson, H. and diSessa, A. *Turtle Geometry* MIT Press, 1981
This is not a Logo book, and the programming examples are not in Logo but 'Turtle Procedure Notation', but a mathematics book. Beginning with squares, Abelson and diSessa explore geometry all the way to relativity. Turtles are not trivial.

Clayson, J. *Visual Modeling with Logo* MIT Press, 1988
Clayson is not a programmer, but an artist, and Logo is merely the vehicle for his explorations of the visual. In quoting Constable, Clayson asks 'Painting is a science and should be pursued as an inquiry into the laws of nature. Why, then, may not landscape painting be considered as a branch of natural philosophy, of which pictures are but the experiments?'

Goldenberg, E. P. and Feurzeig, W. *Exploring Language with Logo* MIT Press, 1988
A grammar book, a very different one from all others. Goldenberg and Feurzeig explore language by generating words and phrases using Logo procedures which incorporate grammatical rules. The Logo procedures become working models of language, not static and abstract rules.

Harvey, B. *Computer Science Logo Style* (Three volumes) MIT Press, 1985..1987
The three volumes are subtitled, *Intermediate Programming, Projects, Styles, and Techniques* and *Advanced Topics*. The first is at about Year 8..Year 10 level, while Advanced Topics is aimed at tertiary level computer science students and demonstrates a clear and viable alternative to the software engineering approach to computing. Anyone using Logo in secondary schools should have at least the first volume.

Hurley, J. *Logo Physics* Holt, Rinehart and Winston, 1985
Turtle as a projectile, cosmology, bridges, rainbows: physics experiments as stimulating simulations.

Lawler, R. *Computer Experience and Cognitive Development: A Child's Learning in a Computer Culture* Ellis Horwood, 1985
Based on Lawler's PhD research, the book details the work with Miriam, his daughter, as she worked in a Logo environment. Reviewed by Prof B. Inhelder as 'The first highly convincing synthesis of cognition science and genetic psychology.'

Lawler, R. and Yazdani, M. *Artificial Intelligence and Education: Volume One Learning Environments and Tutoring Systems* Ablex Publishing, 1987
A collection of papers, including some of Lawler's own, on Logo, intelligent tutoring systems and the like.

Newell, B. *Turtle Confusion* Curriculum Development Centre, 1988
A book of Logo puzzles and related riddles. Perhaps the most unusual Logo book you have seen, but definitely one of the best.

Pea, R. and Sheingold, K. (Eds) *Mirrors of Minds: Patterns of Experience in Educational Computing* Ablex Publishing, 1987
A collection of papers, including a number critical of Logo.

Silverman, B. *The Phantom Fish Tank* LCSI, 1987
A version of Logo for exploring cellular automata, beginning with Conway's 'Life'. Cellular automata are finding uses in many areas of research, and this package is an excellent introduction. (For Apple ][ machines only.)

Solomon, C. *Computer Environments for Children* MIT Press, 1986
Cynthia Solomon was involved in Logo almost from its beginnings. In this book she compares four computer environments for schools, those of Suppes, Davis, Dwyer and Papert, concluding that Logo is the most appropriate.

**Weir, S.** ***Cultivating Minds: A Logo Casebook*** Harper and Row, 1987

**Sylvia Weir has also had involvement with Logo from its early days. She describes a number of instances where Logo was instrumental in enabling children with disabilities to learn, communicate and take their place in the community. Her work with autistic children is particularly interesting. Here, clearly, is Papert's vision realised.**

**There is one final thing that must be said: Do not attempt to develop your use of Logo in isolation. To do so, as in many fields, is to run the danger of stagnation. Besides resources like the books listed above, there exist local journals and user groups which are in being to communicate and share ideas. If that sounds like a 'plug' for *POALL* and BiKiLog I make no apology. Logo is not just a programming language, it is a culture, and cultures are people.**

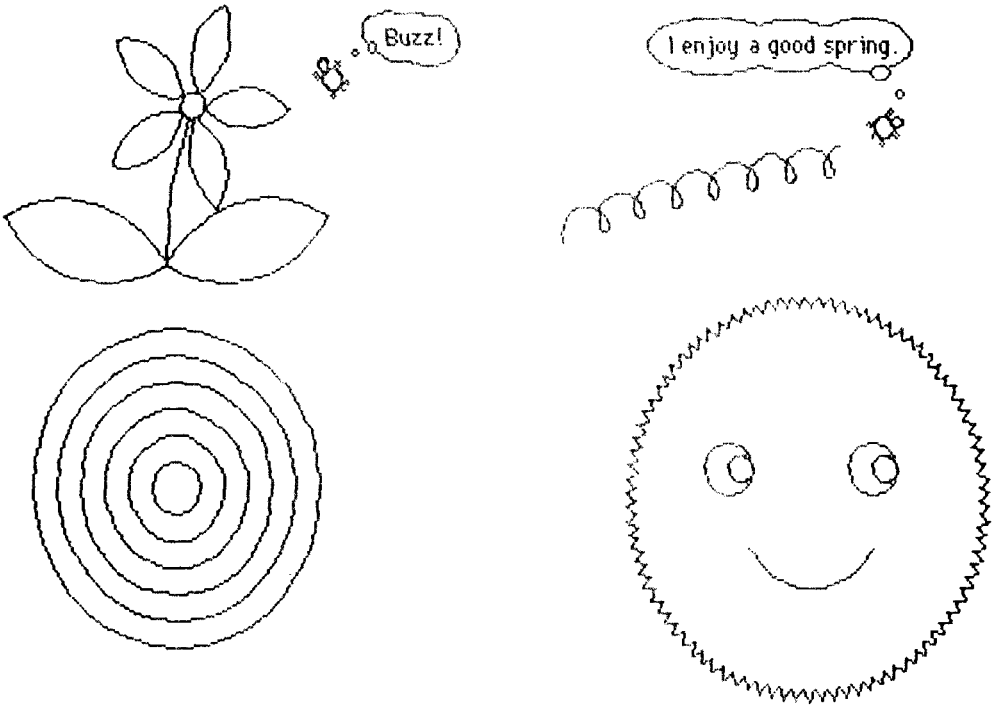
## References (not listed above)

**Carter, P. J.** ***Thinking Logo*** 1987

**Dawkins, R.** ***The Blind Watchmaker*** Longman, 1986

**Papert, S.** ***Mindstorms: Children, Computers and Powerful Ideas*** Basic Books, 1980

**Weizenbaum, J.** ***Computer Power and Human Reason*** W. H. Freeman, 1976



# Australian Computers in Education Conference 1988

ACEC '88 was held at Observation City, Perth September 25-28th. Below, are brief accounts of the Logo papers and poster sessions.

## Jeff James 'Beyond Turtles and Houses'

The presentation diverged somewhat, deliberately, from the paper and concentrated on drawing and recursion at different levels. The paper deals with the use of Logo at Primary, Secondary and Tertiary levels, and makes the point that Logo is a powerful general purpose language, not the 'toy' it is sometimes taken to be.

At the Primary level, the stress is on programming principles: structure, precision in use of the language, and mathematical principles. At the Secondary level, emphasis is still on structure, together with the use of variables and the passing of parameters, while for tertiary students there are aspects like topology, statistics, numerical methods, differential calculus, and recursion. The latter was much in evidence in the presentation with demonstrations, on OHP and Logotron of the Tower of Hanoi, as well as some interesting trees.

Another point made by James is the need for students to have goals set for them. Three sets of procedures, a bicycle picture, a graph drawing system and a conversational program are appended to the paper.

## Pam Gibbons 'Proceduraility, Modularity and Problem Structure: Introducing Logo through Music'

Pam Gibbons came across the problem that most Logo using teachers face at some stage: students typing very long lines at toplevel, and avoiding the use of procedures. Gibbons examines some recent literature on the problem, concluding that 'structured problem solving is neither instinctive nor inevitable.'

Her solution is to use music as the medium to introduce Logo to Year 7 students, in the following sequence:

1. Play with the TOOT primitive, with arbitrary numbers (pitches).
2. Play with a sequence of TOOTs.
3. Use real note frequencies.
4. Try to produce a tune in immediate mode. (Given the limited length of an input line, the tune will be very short.)
5. Define procedures for notes:

```
TO C
TOOT 523 30
END
```

6. Use the note procedures in immediate mode.
7. Define procedures for phrases and songs.
8. Produce medleys, rounds 'mudleys' (ie. phrases from various tunes), perhaps with variables and recursion.

The progression was demonstrated, ending with Frère Jaques played as a round on two Apples.

Gibbons sees a number of advantages in the method:

1. There is an intuitive breaking up of tunes into phrases and notes.
2. Only one primitive, TOOT, is involved.
3. Immediate mode is very limiting.
4. Long procedures are almost impossible to debug, encouraging short ones.
5. Ideas will almost invariably be present.
6. Cooperation is encouraged.
7. Doodling and happy accidents are minimised, planning emphasised.
8. Music has obvious hierarchical structure.

In closing, Gibbons emphasises the role of the teacher, particularly considering that early experiences may well set the pattern for learners' continued approaches to problem solving.

## Paul Dench 'Logo Shells'

Paul is a member of the Computers in Education Project Team of the Ministry of Education. He has developed 'shell' programs to allow children to develop their ideas within a prepared framework. Because the children's procedures must fit the shell, they must be written in a structured manner, and meet the rules for interfacing with the shell. Such a system imposes obvious constraints but on the other hand supplies the 'house keeping' procedures ready made.

Paul demonstrated two completed projects, a hamburger maker and a train builder. The first presents a menu for the hamburger and its bun: meat, cheese, lettuce and tomato. As each item is chosen it is drawn, and after all are in place the bun is completed, complete with randomly sprinkled sesame seeds. The train builder allowed a choice of colour for the locomotive (steam, but with a whistle from a diesel), and a range of rolling stock and cargo. Both drew good pictures and had been written by Year 6 and 7 students.

Paul also briefly demonstrated the beginnings of a sentence building program for use by Aboriginal students. The grammar of the Wajarri language is rather different from English, for instance, word order is largely irrelevant, and the program has to add the word endings distinguishing subjects from objects. Although the graphics were not yet included, the program reminded me somewhat of *Story Machine*.

## Wing Au 'Logo Programming and Metacognitive Training'

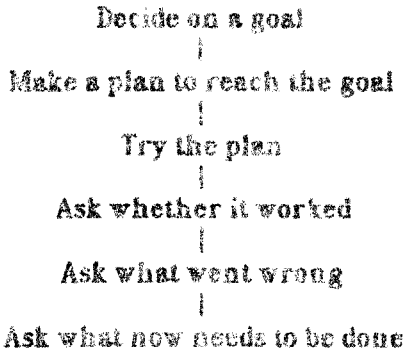
This paper investigates the important issue of transfer of training from the Logo environment to other subjects. This has always been the contentious aspect of Logo, so much so that for some time there have been two camps, one, following Papert, believing that transfer exists, the other, exemplified by Pea *et al*, asserting that it does not, at least in any measurable form. That the argument exists should not be surprising. Rarely do researchers adequately describe their intervention (teaching methods), evaluation, and the test instruments used.

Au is more careful, describing how the subjects (children in Hong Kong schools) were in three groups, one using a Process Approach, the second a Content Approach, and the third what amounted to little more than a Logo typing course.

The Process Oriented Approach involved three aspects:

1. Structured worksheets: a graded set of exercises and instructions, generally following the conventional graphics approach to Logo.
2. Teacher questioning
3. A socially interactive and reflective environment: With students working in pairs the social aspect of Logo, regarded by some as one of the most important, was enhanced; children learn, not so much by the actual programming, but by the interaction and discussion.

Overall, children were forced to think about their own thinking, aided by guides like this (adapted from a paper by Belmont and Butterfield):




Three measuring instruments were used by Au and Leung:

1. Raven Matrices: A system of non-verbal pattern matching exercises.
2. Tower of Hanoi: Au demonstrated this with a model, and charts of the scoring system. The students practised with a 2 disc tower and were then tested on 3, 4 and 5 disc towers.
3. Rule Learning Task: This test involved a set of cards with particular patterns of spots. The children had to determine the rules governing the cards.

Au and Leung found that members of the Process Oriented Group, children who had used Logo in a socially active environment and had been encouraged to think about problem solving processes, scored better in the Tower of Hanoi test than students in the other groups. Differences between the groups in the Raven Matrices and Rule Learning tasks were not significant. Au and Leung conclude that problem solving skills do transfer from Logo to other tasks, but that the Logo needs to be used in a process-oriented environment that focuses on metacognitive training.

Unfortunately the paper was poorly edited for the proceedings, and several lines are missing in various places.

## Renato Schibeci 'Logo in Pre-Service and In-Service Teacher Education'

The paper is an account of Logo courses at Murdoch University, and describes the attitudes of the students before, during and after their exposure to Logo. Overall, they were favourable at the end, feeling that they had learned something of computers in general, some principles of computer programming, and more importantly, something of their own and others' learning styles.

Schibeci believes that perhaps the most important thing the students learned was what it is like to be a novice, and through that to be sympathetic to their own students' difficulties.

## Peter Carter 'LEGO and Logo, the bits between'

LEGO and Logo can be understood and used at various levels, from the TALKTO of TC Logo for primary age children to bit twiddling for senior secondary. You read most of it in *POALL* Vol 3 No 1.

Copies of the Proceedings are available from:

Educational Computing Association of Western Australia
POB 10
Mount Lawley 6050

# Turtle Confusion: Logo Puzzles and Riddles

Barry Newell, Curriculum Development Centre, Canberra

A Review:

Most Logo books fall into a range, from the simple 'recipe' books at one end to Brian Harvey's *Computer Science Logo Style* series at the other. *Turtle Confusion* lies well outside that continuum. It contains not one Logo procedure, and virtually no explanation of Logo systems at all, yet perhaps more than any other Logo book since *Mindstorms* it reaches to the very heart of the Logo experience.

The name is not just a play on words (Why is it that Logophiles are fond of puns?), it highlights the purpose of the book: to stimulate, even provoke, interest, discussion and discovery. Perhaps Newell's own words express it best:

> 'The material in this booklet is intended to be somewhat confusing at first sight. In real life we cannot avoid being confused every now and then. A feeling of confusion is an indication that our understanding of a given situation is inadequate...in other words, confusion signals a chance to learn.
>
> Adults should recognise the danger of shielding young people from feelings of confusion. ... We owe it to our students to allow them to feel confused sometimes; to allow them to learn how to cope with, and even benefit from, confusion; to allow them to develop responses, other than panic, to situations where they are initially 'out of their depth'." (p 43)

I cannot recall ever reading the Piagetian term 'equilibration' in any Logo book, but that is, after all, what 'debugging' is; unexpectedly finding weak spots in one's understanding and rebuilding the cognitive structures with new information. Readers of *Turtle Confusion* will be doing lots of that. Newell has set out to provide ample opportunities for equilibration; learning by examining, discussing, thinking about and the all important debugging of solutions to the puzzles and riddles.

There are no answers. As the Turtle says on page 38, 'there are no answers in the Book of Life', so readers are forced into adopting the methods of science and mathematics, devising hypotheses (Logo procedures) and testing them, and then never being certain. Therein lies a difficulty of course, and this book will not be well received by people who want everything cut and dried, or who believe that all computing in high schools should be based on data processing. The book has greatly frustrated some of my students who have clearly been 'spoon fed' in the past. It is perhaps a reflection on the state of science and mathematics education that a book like *Turtle Confusion* is needed.

Most of the book is in the form of a dialogue between the Turtle and the author, EBN, reminiscent of the dialogues between Achilles and the Tortoise in Hofstadter's *Godel, Escher, Bach*. The dialogues are lively and varied and are the ideal way to present the riddles which are an important part of the book, relating as they do to each other and to the puzzles. The first can serve as a sample:

> 'There was a young student named Myrtle,
> Who tried to converse with the Turtle,
> She later said, "Guys,
> The scales fall from your eyes,
> When you clear the homology hurdle."'

In my experience, young children clear the hurdle without realising it, but secondary students, for whom the book is intended, together with their parents and teachers, are often like Bucephalus, afraid of their own shadows and unwilling to take the leap. To help with the riddles, Newell recommends that students have available a

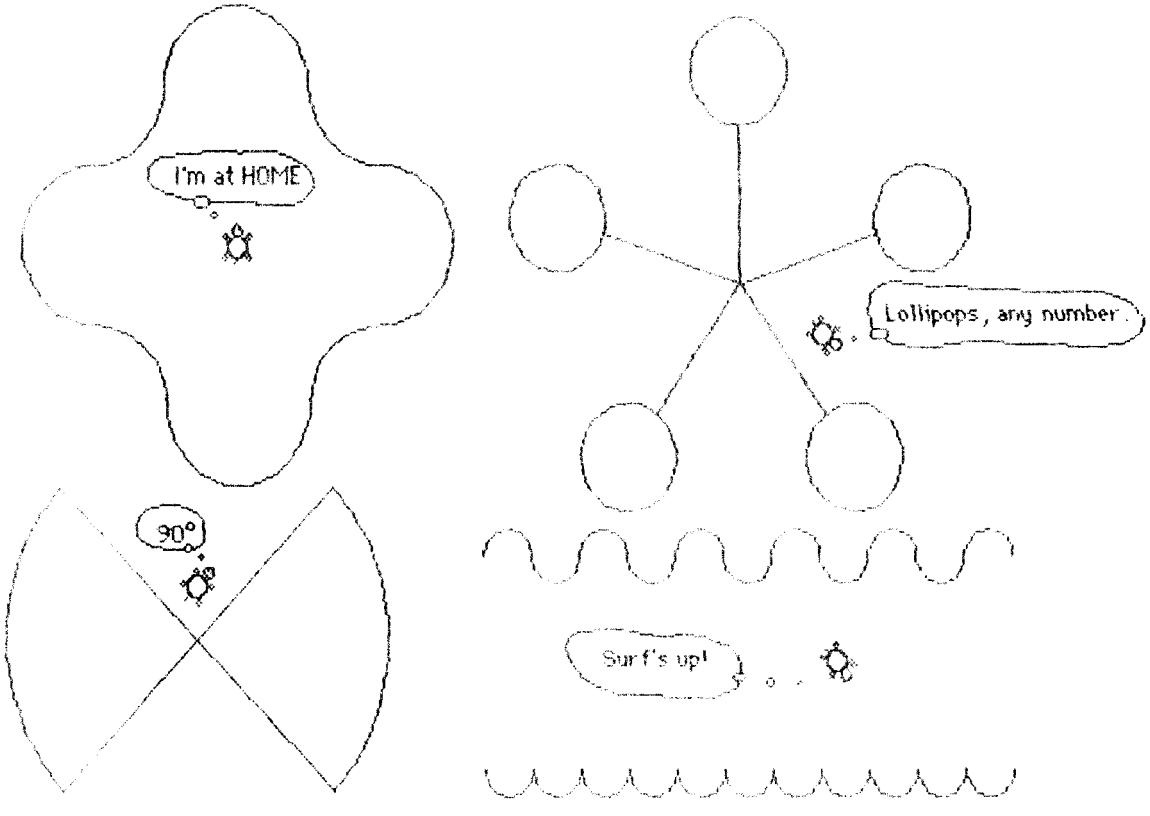
dictionary, atlas, encyclopaedia and 'a modest collection of classical literature.' One of the riddles is enciphered. I used a couple of Logo procedures to help with the deciphering so that I can now read it (But that doesn't mean that I understand it yet).

The forty puzzles are all fairly straightforward. I did them all in one evening, hardly the way the book was meant to be used. Newell emphasises group work and discussion. The puzzles begin with a simple (?) square and end with patterns with multiple axes of symmetry. They relate to each other in often subtle ways, and in many cases a procedure for one can be used as the basis for another, and there are clues to their solutions in the text and riddles. Most of them can be done in a number of ways, and, interestingly, none of them requires recursion. Several require careful planning and use of subprocedures. Newell points out that there's nothing wrong, at least at first, with 'brute force' solutions, but that the neat and elegant should be the eventual aim.

Newell is an astonomer at Mount Stromlo, fluent in FORTRAN and active with Canberra schools through the ACT Schools Authority. His book reflects a commitment to change science and mathematics education to the point where students understand, not because they read about things or watch predictable experiments, but because they are active researchers themselves. That has always been the real purpose of Logo, and Newell's book is a valuable tool to help develop the vital skills of problem solving and research.

As the Turtle says (p vii), 'You can regard the whole booklet as one big problem.' He might have added: 'a thoroughly fascinating and rewarding one'.

*Turtle Confusion: Logo Puzzles and Riddles*
Newell, Barry
Curriculum Development Centre, PO Box 34, Aus Capital Territory 2606
ISBN 0 642 53271 0



# Jumbled Letters

Peter Mitchell, Wirreanda High School

People often talk about 'playing turtle' in order to explain how turtle graphics procedures work. A similiar approach, naturally enough, is very useful in explaining how recursive procedures and operations work. In working on such material, I was side-tracked into thinking of a name, to call such an activity. Playing 'procedures', playing 'Logo', playing 'Words and Lists'. As the problems often involve words or lists (or both) I had a thought of making a word from the letters in the two words Words and List. And so came the thought of writing a procedure, Jumbled, that would actually jumble the letters of an input word. Working on the task, was interesting as it led to the use of procedures (operations) that are very similiar to Logo primitives, and yet I am not aware of any similiar procedures being mentioned elsewhere.

The procedures themselves turned out to be very interesting. Did they lead to a name for the playing. No! But not to worry, as the task was very useful.

## Analysis

My initial reaction to this task was to think of two alternative strategies that I could use in completing this program. One strategy involved looking at each letter in the word, and giving to it, a new position, at random, in the new word to be formed. The second method is essentially the opposite strategy. Get a letter at random from the word, and place it in at the end of the new word that is to be formed. So one method, starts with the letters of the known word. The other looks at it from the positions of the new word. While I was initially attracted to the first strategy, I settled on the second because it reminded me of the some other programs I had written before.

## Design

The primary strategy involved is to select a letter, at random, from the word, and place it onto the new word, and to continue until all letters have been exhausted. In selecting a letter from the word, and placing it onto the new word, the letter will need to be deleted from the known word. Using this strategy I started to think about the commands at my disposal, namely:

| | | | | |
|---|---|---|---|---|
| **FIRST** | **BUTFIRST** | **ITEM** | **LAST** | **BUTLAST** |
| **RANDOM** | **WORD** | **FPUT** | **LPUT** | |

I thought about having a random number, that would be determined by the length (or count) of the known word, taking the item of that number in the word, and putting it into the new word. But there was also the task of deleting it from the list. As there would be two steps required, I decided it would need to be stored. I thought about a command, `ButItem`, so that `ButItem 4 "ABCDEF` would delete D, leaving **ABCEF**

So the primary lines required would be:

- **MAKE** the random number 1 + **RANDOM** of the **COUNT** of the known word
- **WORD** together the letter of knownword at the random number position and the rest of the word, Jumbled.
- The rest of the word Jumbled, would involve ButItem using the random number as an input and the known word.
- The word that has been formed to be output, so that Jumbled is to be a function, to be used, for example: `PRINT JumbledF "mitchell`

I decided I really needed some tools. Using the term ButItem was of course deliberate, and similar to BUTFIRST and BUTLAST. And I was also thinking about a command FirstN, that would be like FIRST, but also use the notion of ITEM. The idea was that FirstN 4 "ABCDEF would result in ABCD, because they are the first 4 letters. And if I was going to create a FirstN, a LastN, a ButFirstN, and a ButLastN also seemed natural enough. (There actually isn't a need for all four). ButItem would consist of the FirstN and the LastN, but without the nth letter.

FirstN will therefore be a function with two inputs, one a number, for the nth position, and the other a word. The number input I'll call :n and the word input, :object. The first letter will need to be peeled off :object and WORDed with the rest of the appropriate letters from :object, in a similiar manner, up until the nth value. Having peeled off one letter, the nth position for the next generation of the word is one less. For illustration, consider FirstN 4 "ABCDEF (I would say this as the First Four of ABCDEFG). I first of all peel off the A, place it in the new word, and what I then require is the FirstN 3 "BCDEF. Peeling off the B will mean FirstN 2 "CDEF will be needed, and similarly FirstN 1 "DEF will be required

Of course, before invoking this process I need to consider a stopping condition. That will be when :n is zero, under the recursive action of the above process. When that occurs I will want nothing to be output to the WORD being formed. So in many ways, similiar to other procedures that use lists or words, with OUTPUT, the following function performs the required task:

```
TO FirstN :n :object
IF :n = 0 [OP "  ] [OP WORD FIRST :object FirstN :n-1 BF :object]
END
```

Creating LastN is naturally rather similiar. See if you can change the order of the preceding function yourself. See if you can also create a ButFirstN and a ButLastN, so they would perform the following:

```
PRINT FirstN 3 "ABCDEFG        displays  ABC
PRINT LastN 2 "ABCDEFG         displays  FG
PRINT ButFirstN 4 "ABCDEFG     displays  EFG
PRINT ButLastN 1 "ABCDEFG      displays  ABCDEF
```

FirstN and LastN with appropriate inputs, can really be used for ButLastN and ButFirstN, avoiding having all four functions. For example, if :n has the value 3 stored, and :object has "ABCDEFG stored; then PRINT ButFirstN :n :object displays DEFG, but PRINT LastN (COUNT :object)-:n :object also displays DEFG because, the COUNT of the object is seven, minus :n, which is three, which in turn gives four, and the last four letters are DEFG.

## Pseudocode

*Define the function Jumbled with an input :object*
*set up a LOCAL variable "rand"*
*test IF the :object is empty,*
*if it is then OUTPUT nothing*
*otherwise,*
*MAKE the rand" the RANDOM of the COUNT of the :obj*
*OUTPUT the WORD formed by the letter at ITEM position :rand" of :object, and the rest of the word jumbled, which involves recurring to Jumbled with input values ButItem :n and the :object*

*Define the function ButItem with inputs :n and :object*
*OUTPUT the WORD formed by the First (:n-1) letters of the :object and the Last*
*of the letters of :object, determined by the input value (COUNT :object)*
*-:n :object*

*Define the function FirstN with inputs :n and :object*
*test IF :n is 0,*
*if it is then OUTPUT nothing*
*otherwise,*
*OUTPUT the WORD formed by the FIRST letter of the :object*
*first (:n-1) letters of the :object without its first letter*
*(i.e. BUTFIRST :object)*

Similarly

*Define the function LastN with inputs :n and :object*
*test IF :n is 0,*
*if it is then OUTPUT nothing*
*otherwise*
*OUTPUT the WORD formed by the last (:n-1) letters of the :object*
*without its last letter, (BUTLAST :object), and the LAST*
*letter of the :object*

Code

```
TO Jumbled :object
; [An operation to jumble an input word]
; [P. Mitchell  October 1988]
LOCAL "rand#
IF EMPTYP :object [OUTPUT " ]
MAKE "rand# 1 + RANDOM COUNT :object
OUTPUT WORD ITEM :rand# :object Jumbled ButItem :rand# :object
END

TO ButItem :n :object
  [buts (deletes) the nth item from the :object]
  [using the two sub-operations FirstN and LastN]
OUTPUT WORD FirstN :n-1 :object LastN (COUNT :object)-:n :object
END

TO FirstN :n :object
  [return the first n letters of the :object]
IF :n = 0 [OUTPUT " ] [OUTPUT WORD FIRST :object FirstN :n-1 BUTFIRST :object]
END

TO LastN :n :object
  [return the last n letters of the :object]
IF :n = 0 [OUTPUT " ] [OUTPUT WORD LastN :n-1 BUTLAST :object LAST :object]
END
```

If your Logo is without a comment primitive, you'll also need this one:

```
TO ; :comment
END
```

Validation

```
PRINT Jumbled "ABCDEFG validates.
```

The following procedure also validates:

```
TO Demo
PRINT Jumbled "aitchell
END
```

Beside ordinary words, try a word with a number of the same letter(s). For example LOOOOOGOOOO

## Evaluation

It is interesting to reflect having solved the problem, and having completed the program, on the initial thoughts and strategy. When first dealing with the task, I was intent on starting with each letter of the word, giving it a random number, and hence position, in the new word. Now I find it difficult to even understand that approach, as the strategy that was applied is so easy and straight forward. And it also is interesting, because in all of the Logo reading I have seen, I haven't come across procedures like `FirstN` and `LastN`.

On the other hand, a procedure (or function), called `Pick`, is very popular, in Logo material. The procedure has been used by many people, and crediting the first to use it is questionable. An early user (and author?) was Hal Abelson. The procedure picks an item from an object, at random:

```
TO Pick :object
OUTPUT ITEM (1 + RANDOM COUNT :object) :object
END
```

Initially I contemplated using this function, but because I had to both pick and then delete the item, I needed a slightly more involved strategy.

What about a operation (procedure (function)) called `PutItem` (or `ItemPut`)?

The tools developed here could certainly be used in a variety of ways, such as creating a larger program about jumbled words, to be used by a teacher, with students. Another use of these tools, would be in a conversation type program. I'm sure you can probably think of other uses of jumbled words

## Exercise

Create a program that jumbles a word, informs the user of the number of letters, the first correct letter, and the jumbled letters; and then allows the user to enter guesses at the correct word, until the user either correctly enters the word, or enters the letter S by itself.

## Editor's Note:

`FirstN` and `Jumbled` exist as `piece` and `shuffle` in Dr Logo (see Vol 2 No 4, p 14) and `PutItem` is `SETITEM` in AcornSoft Logo for the BBC. `PutItem`/`SETITEM` isn't hard to write, and it's set as an exercise, as `Insert` (with `Delete` and `Replace`), in *Thinking Logo* (p 54) with the comment 'could be useful tools at some stage'

The procedures Peter has described are available for downloading in the BiKiLog section of NEXUS.

Send in the jumbled word guessing game you create for publication in the next issue of *POALL*

## The MIDI Turtle

The advent of MIDI systems opens up a field of Logo exploration that was never altogether satisfying before. Most Logo micros have some sort of sound generator built in, but the Apple, for instance, sounds distinctly primitive. With MIDI, the whole range of the synthesiser is available.

We'll need some MIDI driver procedures, and these are written for the Passport interface in slot 2 of an Apple. Other systems will be similar. This is not the place for a MIDI tutorial, but, in brief, a MIDI command consists of a status byte followed by one or two data bytes. For instance, to turn a note on we send the code for note on, followed by the number of the note (Middle C is 60) and the code for the note velocity. (Our Casio MT-600 simply uses the default value of 64 for the on velocity and 0 for off.) The codes are .DEPOSITed in turn into the interface's memory location. The **Reset** procedure is to initialise the interface, **Preset** to change the tone of the instrument, from piano to organ or whatever:

```
TO Reset
.DEPOSIT 49320 19
.DEPOSIT 49320 17
END

TO On :note
.DEPOSIT 49321 144
.DEPOSIT 49321 :note
.DEPOSIT 49321 64
END
```

```
TO Preset :tone
.DEPOSIT 49321 193
.DEPOSIT 49321 :tone
END

TO Off :note
.DEPOSIT 49321 128
.DEPOSIT 49321 :note
.DEPOSIT 49321 0
END
```

```
TO Play :note :duration
On :note
WAIT :duration
Off :duration
END
```

As described in Vol 2 No 4 of *POALL*, Colin Fox and Ursula Gomilschak developed a system analagous to a Turtle moving. In their scheme, a Turtle moves 'forward' by singing a note, and turns right and left by raising or lowering the pitch. The pitch value (MIDI number) is stored as a free variable. A few more procedures:

```
TO Up :semitones
MAKE "pitch :pitch + :semitones
END

TO SetPitch :note
MAKE "pitch :note
END
```

```
TO Down :semitones
MAKE "pitch :pitch - :semitones
END

TO Rest :duration
WAIT :duration
END
```

```
TO Sing :duration
Play :pitch :duration
END
```

Music now becomes a list of instructions to the singing Turtle...

```
MAKE "tune [Sing 20 Up 3 Sing 10 Down 2 Sing 20 Up 4 Sing 10 Down 2
      Sing 10 Up 2 Sing 10 Up 2 Sing 10 Up 1 Sing 30]
RUN :tune
```

and you can try changing the preset tones of the instrument along the way, remembering to **SetPitch** before you start. This Turtle is fond of serial music (its 'Total Trip' is not 360° but 12 semitones), and you can write procedures to turn :**tune**

into an inversion or retrograde as a list processing exercise. You can use procedures instead of the list system, but they are more difficult to manipulate.

Another possibility is musical fractals:

```
TO MusicTree :duration :up :down
IF :duration < 10 [STOP]
Sing :duration Down :down
MusicTree :duration / 2 :up :down
UP :up + :down
MusicTree :duration / 2 :up :down
Down :down Sing :duration
END
```

Try it with 30 3 3 as inputs. All good fun, and a usefully different way of exploring music.

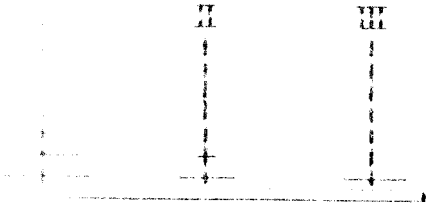
## Dirty Tricks Department Revisited

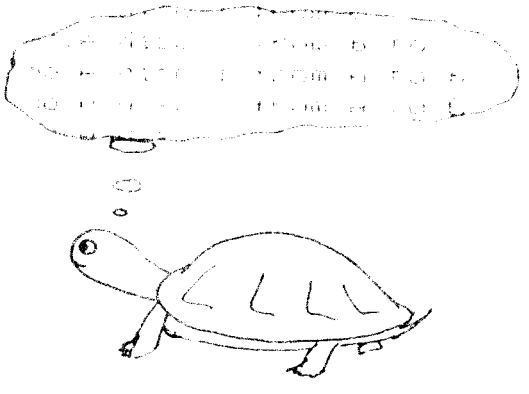
If you are plagued by students who are more interested in reading your demonstration procedures than devising their own, here's something to help keep them hidden:

```
TO xPOPS
PR [And the same to you.]
END

TO xPOALL
PR [A journal for Logo users.]
END

TO xFix
COPYDEF "xPO "PO
COPYDEF "xPOTS "POTS
COPYDEF "xPOPS "POPS
COPYDEF "xPOALL "POALL
COPYDEF "xED "ED
COPYDEF "xED "EDIT
BURYALL
END
```

```
TO xPOTS
PR [We're not POTty.]
END

TO xED :procedure
PR [ED's away today.]
END

TO xPO :procedure
PR [Go away.]
END
```

As shown here, they're for Apple Logo // and/or LCSI Logo II, but they will work in any Logo that allows for primitive redefinition and burying. Of course, normal editing is disabled; the best way out of that is through the use of EDITFILE.



## Computing at Enmotch Homes

It's time manufacturers and suppliers were realistic about the needs and finances of schools. One school recently bought 16 new Macs and wanted a Logo to use on them. The first problem was 'which one?' There are at least four: LCSI/Microsoft MacLogo, Coral Object Logo, Terrapin Logo and ExperLogo. The last named is rather Lisp-like, functional, Terrapin Logo doesn't seem to be readily available, Object Logo is an extremely powerful system, but seems more suited to tertiary use than for schools, leaving MacLogo. After many 'phone calls, 15 copies were tracked down, and ordered. Could there be a site licence arrangement, as with MS Works? Certainly not. Each copy would cost the full price. Cries of despair. A couple of days later, the price was reduced: $170 a copy, and 'sorry, one of the copies you ordered has been sold.' More requests for negotiations for a site licence were refused, and legal action was threatened if the school tried any sort of unilateral arrangement. It's cost that school most of its software budget for the next year just to buy Logo. That's not only ridiculous, it's an almost certain way to invite illicit copying. The school concerned wanted to be fair and honest, yet it has been penalised by the archaic and ignorant practices of some software houses.

LEGO Australia is currently without an educational representative in South Australia, with Di Jackson accepting a research position elsewhere. For orders and info, ring LEGO in Sydney on 008 251 031. Control Logo for the BBC is now available, and requires the Logotron chip or image. The package is well presented, in LEGO's new metallic grey and diagonal stripe livery and is the first in the new 'dacta' line. What's that mean? Apparently only the Danes know. The materials are comprehensive, but have an altogether different flavour from the Apple materials, reflecting their UK origin. It seems like a case of 'here's how the technology works, learn it so that you can live with it' rather than the 'what can you discover about this and make with it?' approach from MIT. The primitives have a different flavour too; instead of `TalkTO [A 3 5]` On and the like, it's `TURNON [A 3 5]`. The system puts a graphic of the interface on the screen, which can be turned off with `NODISPLAY`. The documentation includes some useful ideas for using nonLEGO materials, wood, cardboard, etc. School price (site licence) is $310.

Theme for the next ACEC conference is 'Backup the Future' and the logo (ie. the other sort) is based on an image from a well known manufacturer of disks, photocopiers and things. Are they sponsors, one wonders?

Boxer was mentioned at both ACEC '88 and CEGSA conference, and was given a good write up in *The Australian* on November 1st, complete with portrait of Liddy Nevile.

Australian
Computers in Education
Conference -
Canberra 1-4 October 1989

Conference Secretary:

ACEC '89
P.O. Box 311,
Manuka, A.C.T. 2603