The Educational Programming Language Logo: Its Nature and Its Use in Australia

Anne Mcdougall, John S. Murnane, Sandra Wills

The Educational Programming Language Logo: Its Nature and Its Use in Australia

Anne McDougall[1], John Murnane[1] and Sandra Wills[2]
[1]Melbourne Graduate School of Education, The University of Melbourne, Australia
[2]Australian Graduate School of Education, Charles Sturt University, Australia
jmurnane@unimelb.edu.au, a.mcdougall@unimelb.edu.au,
swills@csu.edu.au

Abstract: Although Logo was expressly designed as a Mathematical language for use in Education its early versions were very logic-orientated. With the addition of Seymour Papert's 'turtle' the system became far more accessible to students and teachers. This paper explores some of the educational ideas behind its development and describes its first use in schools in Australia through reminiscences by two of the co-authors, Sandra Wills and Anne McDougall. The paper concludes with a reflection that educational research has not been able to *prove* the educational benefits of a ground-breaking approach that empowered students with computers. However, many rich case studies of successful implementation by passionate teachers abound in the literature to provide inspiration to teachers working with the new digital natives.

Key Words: Logo, turtles, Logo in Australia, Lego Logo, constructivism

1. Introduction

Early in 1973 one of the co-authors, Anne McDougall, commencing a newly created position as Research Fellow in Computer Assisted Instruction at the University of Melbourne, set out to read every research paper then published concerning the use of computers in educational contexts. She found accounts of innovative work with drill and practice programs for learning, computer managed instruction, and some preliminary computer simulation and modelling programs. She also found a set of papers from a group working at MIT in the USA (Papert, 1971a; Papert, 1971b; Papert & Solomon, 1971). These argued that most current work in educational computing involved the computers 'programming the children,' and by contrast suggested that students should themselves learn to program the computers to enable powerful learning through exploration of novel ideas, situations, concepts and problems. These papers described a programming language called Logo, developed specifically for this purpose. Reading them, and being delighted by their approach but reflecting on the current state of computer hardware and software, she wondered whether such an opportunity might be provided for students during her lifetime. Just three years later, thanks to some extraordinary initiatives by two teachers in Tasmania, her own students were using Logo.

This paper is concerned with the development and nature of Logo, and describes some innovative and influential Logo work done in Australia. We begin by examining

the educational concepts behind Logo—and its subsequent versions such as LogoWriter and MicroWorlds.

2. A Language for Learning

The creation of a programming language of any sort is a complex business and the province of a special elite in the world of programming. Yet the difficulty of the creation of a language for fields such as business or mathematics, for which there is an existing set of well-tried models, pales into insignificance compared to the task of creating one for educational purposes: a space where a choice must first be made between various pedagogies, all with their own built-in advantages and disadvantages, advocates and detractors, before even the form of the language is decided on. Nor will the educational aim be simple. Is it to be a language to introduce the forms and disciplines of programming itself, or is it to facilitate a more general development in problem solving and analytical and logical thinking? Are we teaching computing and its applications, or are we using programming for some wider educational purpose?

Looking back at the a-priori positions on the benefits that writing programs could bring to students, we find a remarkable unity of spirit among the pioneers of educational computing languages. Cynically, it could be argued that in the 1960s, apart from some rather inflexible Computer Assisted Instruction (CAI), and some (mostly non-interactive) simulations, writing a program was about the only educational thing students could do with a computer. But the pioneers, and particularly those with a hand in writing specialised educational languages, such as Kemeny and Kurtz who developed Basic, and Feurzeig and Papert, the developers of Logo, were all convinced that great educational advantages would come from programming a digital computer, although often for different educational and cognitive reasons. Weyer and Cannara (1975, p.3) put it this way: "If … any ideas which can be formalised may be studied concretely via a computer program, then, by learning programming in full generality, students can learn to construct laboratories to study any ideas they wish to think about."

The decade that produced the first educational computer languages is now 50 years in the past. With the singular exception of anything written by Seymour Papert, looking back through the papers and reports leaves a distinct impression of teachers striving towards goals imperfectly grasped, using computing equipment barely up to the task, and hampered by primitive translators, operating systems and input/output devices. Yet the *overall* feeling is of high optimism, more positive than one finds across the more recent literature. (Papert who had worked with Jean Piaget for many years, knew what he was doing from the outset, and Wally Feurzeig and Papert, working with Lisp at the Artificial Intelligence Laboratory at MIT in the USA, not only had an example of a language congruent with their educational ideas, but one in which their new language could be written.) The pioneers, from their own experiences, *knew* that programming a computer had educational benefits, and were going to set about proving it to the world.

3. A Language for Learning Mathematics

Feurzeig's Logo group began with *education* and worked *back* to the form of their language. Early Logo was very simple. Like Basic, it came with built-in editing and file manipulation commands. If these are ignored, the 1971 version consisted of just 26 'operations,' five of which accessed the calendar and clock, and 15 'commands.' Most operations were concerned with program logic or list manipulation. An essential part of the design was to produce a language of such simplicity that it forced users to write their own library of commonly used routines, such as multiplication and division. From such a library, complex programs could be built. "Ideally, by the end of the course, each student would have created his own extended version of Logo" (Brown & Rubinstein, 1974, p.10).

There is no mention of the degree of difficulty inherent in learning to program generally, and certainly not in learning Logo, in Feurzeig's papers. The entire emphasis in *The Final Report* (Feurzeig et al., 1969) is on the difficulty of learning Mathematics, and how Logo was developed to make that easier. The programming language followed as a result of a specific educational need. The designers of Logo intended it not only as a vehicle to express Mathematical ideas and make Mathematical concepts concrete, they saw it also as a meta-language in which to express Mathematical thought (Feurzeig et al., 1969, p.5). Here is the origin of Papert's often expressed need to teach 'thinking about thinking' (Papert, 1971a, p.2) and the decision to write a computer language whose primitives and predicates inherently contained and *expressed* the mechanisms of logic and Mathematics. "Do we give children the instruction 'think!' without even telling them how to think?" (Papert, 1971a, p.4). The mathematical purpose expressed by Feurzeig is actually at odds with Papert who is at pains to stress the general problem solving capability of the language (Papert, 1971a; Papert & Solomon, 1971). Lisp's origins in Artificial Intelligence were supposed to support this (Evans, 1992, p.14; Abelson et al., 1976, p.16), but no author we have read ever explained how it was to happen.

The 'Lo' in Logo suggested 'Logic,' and the earliest versions of the Logo system were written in Lisp, a list processing language. It is curious then that early papers (Feurzeig et al., 1969, p.1; Feurzeig et al., 1971, p.1) make it clear that Logo was "expressly designed" for the teaching of Mathematics. At that time there was no Turtle Geometry, and indeed no arithmetic functionality beyond addition and subtraction. Brown and Rubinstein (1974, p.3) flatly describe it as 'non-numeric'.

Many of the programs in the *Final Report on the Logo Project* (Feurzeig et al., 1969) seem forced, elementary and repetitive. Many from the primary level, ages 7 to 9, are examples of programs to reverse the letters in a word, print a set of consecutive numbers or simply print strings. The first lessons did not involve writing code at all. This did not happen until Lesson Seven (p. 67). In the secondary curriculum, many essential elementary functions such as divide and multiply were written by the teachers and given to the students to try to understand (p. 215), the inference being that students could not be expected to write these routines themselves. Johnson (2000, p.201) found "The position that the programming environments themselves, e.g., Logo microworlds, would become the school mathematics curriculum has clearly failed to gain the support of the educational system."

None of this suggests a language easily taken up by beginners and used for their own purposes. Part of the reason has to be the use, initially, of recursion for all loops, definite or indefinite. Recursion is, as Papert has said repeatedly, a powerful problem solving tool (Papert, 1980; Papert, 1971a; Papert & Solomon, 1971; Papert, 2002) and indeed it is. But then, so is calculus!

Papert's work with Jean Piaget resulted in a passionate belief in the idea of 'learning by doing,' something he later extended to what might be termed 'learning by *making things*.' Papert in particular has always insisted that Logo is designed to encourage experimentation, with students writing and testing their own creations. Given this emphasis, it is difficult to understand the reliance on recursion at the expense of a general iterative statement. We can find nothing in the early literature that asserts that students can be expected to discover a recursive solution to a problem on their own. All we can find are examples provided *to* students to explain, understand, and adapt. In his seminal book, *Mindstorms*, published in 1980, Papert states that "recursion stands out as the one idea that is particularly able to evoke an excited response." That might be so, but he devotes less than two pages to it, mentioning it once more in the Appendix in the context of 'circular logic' (p109). Brown and Rubinstein suggest that with suitable prior experience, students can write their own recursive routine to traverse a tree, but these writers give no clue to their success rate. They did find that "if a student couldn't figure out how to write a function, we could not slowly lead him down the path to discovery" (Brown & Rubinstein, 1974, p.43). Once acquainted with WHILE—DO in Basic or Pascal, or even the primitive Dartmouth-Basic GOTO, students have no trouble in writing their own indefinite loops (Murnane, 1991; Murnane, 1992). (See also McDougall, 1985.)

4. Enter the Turtle

Even allowing for difficulty in conceptualising recursion, Logo is not English, and in its early versions Logo struggled to make progress. The cure for many of these problems was provided by Seymour Papert. Logo is often associated specifically with Papert, and particularly with his Turtle Graphics. He joined the project in January 1969 as a consultant (Feurzeig et al., 1969, p.1) and his invention of the Turtle and its commands transformed the language.

A Turtle is a small robot which, when connected to a computer, can move and turn on the floor. At a stroke this eliminated the gap between entering a program and observing its outcome, since the Turtle could execute a command as soon as it was entered. It also solved the problem of students understanding what the command did. While they might need to be taught the meaning of "TEST IS COUNT /SENTENCE/ 1 (Feurzeig et al., 1971, p.45) they could easily appreciate what FORWARD 100 meant because it accorded with their own body actions and their natural language. Turtle Geometry provides an immediate and meaningful environment for the beginner.

Along with Turtle commands came definite iteration: REPEAT :N, relieving the programmer of the need to write all loops recursively. A recursive loop can only be executed by writing a procedure and then executing it. In keeping with the idea of

observing actions as the commands were entered, you could now type REPEAT 4 [FORWARD 100 RIGHT 90] and *watch* a square being drawn. Note also the close correspondence to English syntax. Once the Turtle migrated from the floor to the screen, Logo became accessible and viable in any classroom.

5. Logo Comes to Australia

In 1975 Scott Brownell, a teacher in Australia's island state of Tasmania, saw Logo at MIT on a study tour and in a visionary move brought a magnetic tape copy of Logo from Boston to Hobart to run on a PDP-11 mini-computer at the Tasmanian Education Department's Computer Centre (Richardson 1997). With a Commonwealth Schools Commission Innovation Grant, he was able to recruit a Tasmanian teacher, Sandra Wills, to extend into primary schools the work on computer awareness and computer science already underway in secondary schools in the state, using Logo as the vehicle. He secured a rare and expensive robot Turtle from the General Turtle Co. in the USA. This was at a time when the telephone system was used to connect every senior high school in Tasmania with a terminal to the PDP-11. Over the next few years, Sandra would load the turtle into her car and travel all over the island visiting the Education Department's 300 schools. At each school the children would connect the Turtle and a Tektronix graphics terminal to a home-made modem to dial up the PDP-11 in Hobart.

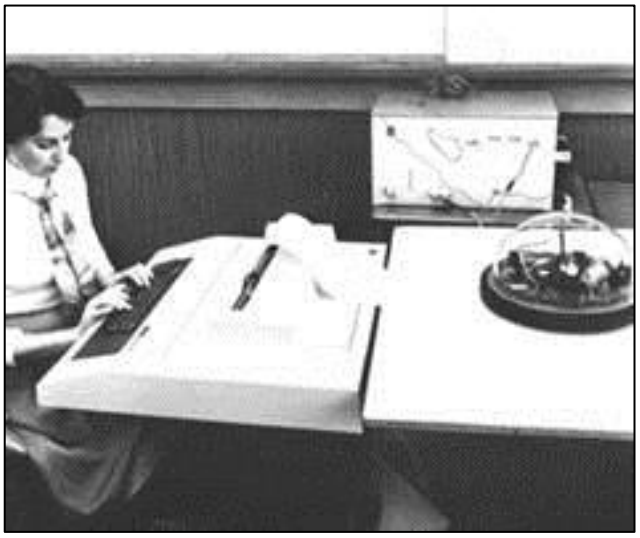

Figure 1: General Turtle Inc. robot (interface box in background) with Sandra Wills at the controls of the PDP-11 terminal, 1976.

The Logo project was instrumental in spreading computing, not only into primary schools, but also middle high schools because it changed people's perceptions of computing. At a time when the general population had never seen a computer nor knew what it might do for them, the Turtle and the English–like Logo language,

particularly Papert's geometric Turtle commands, dispelled perceptions that computing was only for the Mathematical geniuses.

John Gilbert, originally from Hatfield in the UK but then working at the Centre in Hobart, wrote a version of Logo in which a virtual turtle left lines of asterisks on the screen to approximate the path the floor turtle was taking. (Wills, 1980; Wills, 1981). In 1976 the Tasmanian group sent Anne McDougall this version of the language to introduce Logo to her students in the Faculty of Education at the University of Melbourne.

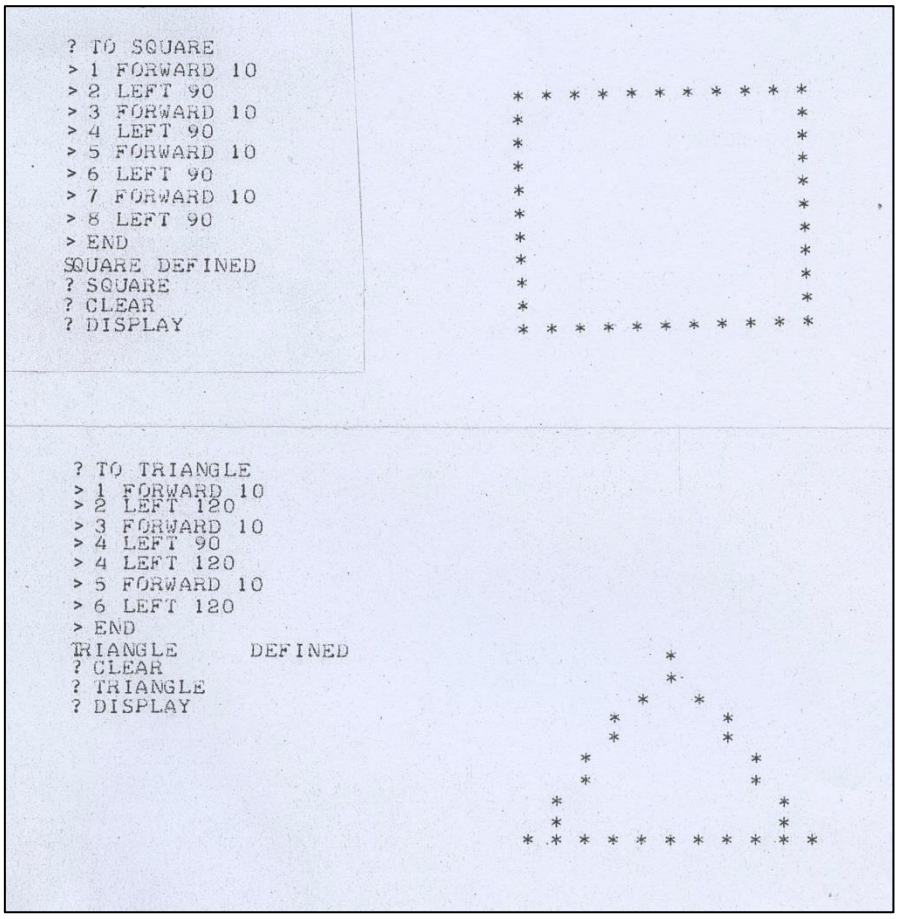

Figure 2: Virtual turtle tracks

Once personal computers were invented in the late 70s early 80s, Richard Miller of the University of Wollongong in New South Wales wrote the first version of Logo to run on the Apple][, specifically to drive the Turtles in the Tasmanian project. Wills, in collaboration with Miller and Allan Branch, a Tasmanian engineer, oversaw the design and manufacture of a smaller and relatively inexpensive floor turtle, the Tassie Turtle, which achieved a degree of accuracy and precision that had eluded similar

research and development efforts in Edinburgh and elsewhere, and could be run from a 5.25 inch floppy disk on the Apple][(Richardson, 1997).

Figure 3: Primary school class in Hobart working with the robot turtle and Sandra Wills, 1977

A number of schools and teacher training institutions across the country were now using Richard Miller Logo and the Tassie Turtle to explore what might be done with Logo. In 1981 Seymour Papert visited Australia. Arriving in Melbourne to speak at a conference, he was moved to tears when he found himself in a room surrounded by a swarm of buzzing, beeping robot Turtles (Richardson 1997).

Papert's visit built on the pioneering work of the Tasmanians to inspire a generation of Australian Logo workers. The next few years saw the publication by Australian authors of several widely used Logo books (McDougall et al., 1982; Nevile & Dowling, 1983; Carter, 1987; Newell, 1988a; Newell, 1988b). *Learning Logo on the Apple II*, by Anne McDougall and Tony and Pauline Adams, was

translated into German, French and Chinese. In Victoria, Logo was included as one of the recommended programming languages for senior secondary school Computer Science courses. A Special Interest Group of the Computer Education Group of Victoria, OzLogo, was established, and a quarterly newsletter, POALL, was produced and edited by Peter Carter at the University of Adelaide in South Australia.

Contingents of Australians attended, and some presented at three Logo conferences, Logo84, Logo85 and Logo86, held at MIT. In the ensuing years Logo leaders from the USA, including Hal Abelson, Andy diSessa and Brian Harvey, also visited Australia, to speak at conferences and to work with locals in several of the states. Gary Stager from the USA did extensive work in teacher professional development in Australia and Seymour Papert presented a memorable closing keynote address at the 1990 World Conference on Computers in Education in Sydney.

A conference, *Logo in Australia: Ten Years On*, was held in Melbourne in 1985 to mark ten years of Logo work in Australia. By then Logo use was being investigated in many parts of the country, at all levels of education from kindergarten, where a single-key version of the language developed by Pauline Adams was used (Adams, P. 1985) through primary (Wills 1985) and secondary (Squires & Sellman 1985; Clarke 1985) schools, to teacher pre-service and in-service education (Jones 1985; Nevile 1985). Selected papers from this conference were subsequently published (McDougall 1996).

Figure 4: Seymour Papert and Sandra Wills with Tassie Turtle robot at Computer Education Group of Victoria Conference, 1981

Consistent with developments in computer hardware, new and enhanced versions of Logo appeared from development centres in the USA. These included LogoWriter, TurtleMath, GeoLogo, StarLogo, Lego TC Logo, Object Logo and MicroWorlds, and investigation and adoption of these continued in a variety of school settings across this country. Associated with the adoption of laptop computers in some schools, in

1989 the Australian Council for Educational Research and Liddy Nevile set up the Sunrise Project—laptop computers in two pilot schools, Coombabah Primary School on the Gold Coast in Queensland and Methodist Ladies' College in Melbourne (Richardson 1997). Children in Years 5–7 were each given a laptop computer with LogoWriter. The entire curriculum was conducted and expressed by the children as LogoWriter projects. Many schools emulated this approach, and LogoWriter and then MicroWorlds were used in laptop initiatives all over the country.

A second conference, *Learning in Logo Microworlds* was held in Melbourne in 1996, marking twenty-one years, the 'coming of age' of Logo in Australia. The Proceedings were published (McDougall & Dowling 1997) along with another book of Logo-related research papers, *Logo in Australia: Selected Readings* (Oakley 1996). The across-curriculum use of LogoWriter or MicroWorlds was associated with increasing use of thematic and project work approaches to curriculum planning, more collaborative, group and discussion work, and greater flexibility of timetabling (Chapman 1997; Best 1997; Betts 1997; McDougall & Betts 1997; Costa 1997; Kerwin 1997). As well as these cross-curricular applications, Logo use for enhancement of understanding of concepts in specific subject areas continued, for example in Science (Duncan 1997; Hopkins 1997), Mathematics (Yelland & Masters 1997), and as a stimulus for reflection about natural language (Dowling 1985). Student competence in programming (Oakley and McDougall 1997) and issues in the teaching of programming (Betts 1997) remained matters for discussion.

6. Research

In Australia, as in other countries, many research studies were undertaken investigating aspects of the use of Logo by students of all ages (see for example Gibbons 1997, p.10) and in a wide range of settings (for example Hopkins & McDougall, 2003). We will outline here several more distinctive topics studied in this country.

Almost from the first, Logo has gone hand-in hand with Robotics, a natural extension of Papert's invention of the Turtle. Early experimenters often built their own interfaces to run computer-controlled models constructed from Fisher Technic and similar kits. One of the authors, John Murnane, built on ten years experience with Lego TC Logo and internally constructed interfaces for the Macintosh Classic by Jon Pierce at Melbourne State College. He was particularly interested in investigating recursion, as programming a robot in Logo not only requires it but provides a concrete model for students to explore directly (Murnane & McDougall, 2006). He also used Lego kits to investigate linguistic differences between using a written programming environment and a graphical one (Murnane 2010). Debora Lipson built on this work, concentrating on the mechanical design and construction of the robots, and providing new insights into teaching the Mathematics of gearing (Lipson, 2008). Interestingly, Lipson found confusing differences in the notation used by Mathematicians and Engineers to describe gearing ratios. Meanwhile, Peter Carter, working at Plympton High School in South Australia, was developing models in Lego/Logo to enable students to explore some of the issues in robotic locomotion (Carter, 1990).

The use of recursion in Logo has been the focus of a number of projects. Murnane and Warner (2001) illustrated examples where children with prior experience in LogoWriter but without specific teaching about recursion, could have, but failed to use recursion in programming. In fact, there was almost no trace in their algorithms of them having encountered Logo at all.

The nature of some of the difficulties students have with using recursion in programming were analysed by McDougall in her Doctoral thesis (McDougall 1985). Anne was as able to show that young children can recognise, devise and interpret recursive structures and processes presented independently of the computer, in pictures, stories and everyday situations in their experience. She used techniques based on this finding to facilitate effective use of recursion in programming by children of upper primary school age (McDougall, 1985; McDougall, 1990b). Pamela Gibbons (1995a; 1995b) extended this work in a study of individual differences among adult students learning to program with recursion.

Gibbons (1997, p.7) discusses the value for a researcher of listening to students talking about their learning with Logo, and comments on students' remarkable ability to analyse and articulate their own thinking when they talk about their Logo experiences. John Vincent, working with upper primary school children and MicroWorlds found similar articulateness. He also studied individual differences, but focused on the visually rich aspects of the technology and interaction of these with children's writing. He found strong interactions, with excellent support for writing development in some students previously considered weak in language skills but with preference for visual communication (Vincent, 2001; 2002; Vincent *et al*., 2010).

The weight of international research suggests that programming in Logo, by itself, does not teach Mathematics. Ross and Howe (1981, p.147) found that "the research of the last decade into 'mathematics through programming' has been more encouraging than discouraging, but only mildly so." Students, unless specifically taught about these points, keep Logo and Mathematics entirely separate in their minds, and few teachers seem to work to overcome this, or do so with much success. Even Abelson, Bamberger, Goldstein and Papert (Abelson et al., 1976, p.10) rather sadly remark that "Logo did not succeed in displacing Basic as the almost universal computer language for schools."

Despite encouraging results such as those with robotics, recursion and visualisation, Australian research on wider use of Logo and its subsequent versions has been consistent with the above. Tony Adams wrote "The Logo community has never come to terms with lists, they are just too hard to manipulate. Unlike turtle graphics they do not have a low entry threshold and for most applications a good working knowledge of recursion is required." (Adams, 1985, p.22). Pam Gibbons adds, "Despite what I have seen in the classroom, I know that Logo struggles into adulthood. …[I]t fights the perception that 'real programmers don't program in Logo, that is, it's just for kids; in a discipline where obtuseness and mystique command respect, Logo is its own worst enemy." (Gibbons, 1997, p.17)

This problem has been exacerbated by a turn-around in one of Logo's original, fundamental, principles: keep the language small and have the student develop their own set of useful procedures. This forces them to write most of their own material and, the theory says, thereby understanding it.

Logo has been extended far beyond the limits any of its creators could have imagined: the Computer Science of the 60s gave no inkling of the possibilities the personal computer and object-oriented programming would bring. Logo, in the form of MicroWorlds, is part of a full-blown multi-media/robotics environment and in 2014 is probably the only language that makes Cobol look small, or offers the same invitation to write the same thing in so many different ways. Feurzeig's successors seem to invent a new command every time they have a new idea, even when existing commands would seem to be perfectly suitable to the purpose. For instance, the MicroWorlds Robotics version adds a completely new, quite separate set of commands to talk to the Logo RCX 'brick.' This leaves the existing, and quite adequate, 'Talkto' protocol in the main body of the language and separates Lego Robotics from the use of its rich array of logic. Redundancy in the language is therefore rife, while it is axiomatic in computer language design that there should only be one way to do something. On the other hand, the MicroWorlds Backpack is a brilliant model of an object, though the language itself cannot really be said to be object-oriented

Gibbons is correct: commercial programs are not written in Logo, and it was never intended that they should be. Instead, it was developed as a vehicle for learning to work with and physically model otherwise complex and difficult situations and problems, making their nature visible, documented and testable. Automating the glittering array of operations offered by later versions does not materially add to the educational possibilities, it just changes their nature. It may well prove even more difficult to show the new multi-environments enhance learning and the general curriculum than did the simple original.

7. Conclusion

Sadly, the weight of research concludes that Logo's advocates have not demonstrated the gains that exposure to its ideas are supposed to bring. That said, given the enormous number of contributing factors, research demonstrating that experience with programming carries over into written expression, problem-solving and clear-thinking is inherently extremely difficult. Essentially the teacher of today is in no better position than the pioneers, and is really just dependent *on their own belief in the promise that having students write programs will bring educational and other advantages*, a belief shared by the authors of this paper. The pioneers of educational computing *knew* that programming a computer had educational benefits and set out to prove it, but at the moment, when programming has all but disappeared from the curriculum, it seems that the world was not listening.

References

Abelson, H., Bamberger, J., Goldstein, I., Papert, S. (1976) Logo progress report 1973–1975. Cambridge, Mass., Massachusetts Institute of Technology. AI Memo 356, 22 pp. ERIC ED 128 181

Adams, P. (1985) Various Computers in the Kindergarten. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 81 – 87.

Adams, T. (1985) Towards a Theory of Microworlds. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 13 – 24.

Best, N. (1997) MicroWorlds in a Year Seven Class. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 70 - 77.

Betts, J. (1997) Teaching Programming Without Teaching Programming. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria,

Brown, J.S., Rubinstein, R. (1974) Recursive functional programming for the student in the humanities and social sciences. Irvine, California, University of California. 53 pp. UCI–ICS–TR–27a. ERIC ED 108 664

Carter, P. (1987) *Thinking Logo: An Introduction to (the Universe through) Programming*. Loxleys, South Australia.

Carter, P. (1990) Step by Step: Logo Legged Locomotion. In McDougall, A. & Dowling, C. (eds.) *Computers in Education* Amsterdam, North-Holland, 743 – 746.

Chapman, T. (1997) Logo and MicroWorlds at Westall Primary School: A 4 Year Tale. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 101 - 108.

Clarke, V. (1985) Logo Tool Kits. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 38 – 45.

Dowling, C. (1985) Logo and Language Development. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 31 – 37.

Duncan, C. (1997) Microworlds: Making the Connections between the Abstract Concepts in Elementary Science. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 109 - 121.

Evans, P. (1992) *What is Logo*? Deakin University Press, Geelong, Australia.

Feurzeig, W., Kukas, G., Faflick, P., Grant, R., Lukas, J.D., Morgan, C.R., Weiner, W.B., Wexelblat, P.M. (1971) An Introductory LOGO Teaching Sequence: LOGO Teaching Sequence on Logic. Cambridge, Mass., Bolt, Beranek and Newman. 329 pp. ERIC ED 057 579

Feurzeig, W., Papert, S., Bollm, M., Grant, R., Solomon, C. (1969) Programming-Languages as a Conceptual Framework for Teaching Mathematics. Final report of the first fifteen months of the Logo project. Washington, D.C, Bolt, Beranek and Newman. R–1889. 329 pp. ERIC ED 007 932

Gibbons, P. (1995a) A Cognitive Processing Account of Individual Differences in Novice Logo Programmers' Conceptualisation and Use of Recursion**.** *Journal of Educational Computing Research*, Vol. 13, No. 3, 211 – 226.

Gibbons, P. (1995b) Recursion: An Analysis of Individual Differences in Logo Programming. Ph.D. thesis, University of Sydney.

Gibbons, P. (1997) Logo Learning: What the Learners Say. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Melbourne, Computing in Education Group of Victoria 7 - 19.

Hopkins, Josie (1997) Stars and Sprites. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 133 - 141.

Hopkins, J. & McDougall, A. (2003) Constructionist Learning and Teaching in a Computer Clubhouse Environment. In McDougall, A., Murnane, J., Stacey, C. & Dowling, C. (eds.) *ICT and the Teacher of the Future*. Sydney, Australian Computer Society, 65-66.

Johnson, D.C. (2000) Algorithmics and programming in the school mathematics curriculum: support is waning—is there still a case to be made? *Education and Information Technologies* Vol. 5, 201–214**.**

Jones, A. (1985) Logo in Preservice Teacher Training: An Australian Experiment. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 54 – 61.

Kemeny, J., Kurtz, T. (1967) *The Dartmouth Time-Sharing System.* Washington, D.C., National Science Foundation

Lipson, D. E. (2008) Gearing Up for Robotics: An Investigation into the acquisition of the concepts associated with gears by teachers in a constructionist robotics environment. In *Learning to Live in the Knowledge Society*. Proceedings of the International Federation for Information Processing TC3 Conference, Milan, September 2008.

McDougall, A. (1985) Teaching and Learning about Recursion. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 46 – 53.

McDougall, A. (1990a) Student Difficulties in Programming with Recursion in Logo. In McDougall, A. (ed.) *Back to the Future, Forward to the Past*, Melbourne, Computer Education Group of Victoria, 108 – 115.

McDougall, A. (1990b) Children, Recursion and Logo Programming: An Investigation of Papert's Conjecture about the Variability of Piagetian Stages in Computer-Rich Cultures. In McDougall, A. & Dowling, C. (eds.) *Computers in Education* Amsterdam, North-Holland, 415 – 418.

McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria.

McDougall, A., Adams, T. & Adams, P. (1982) *Learning Logo on the Apple II*. Melbourne: Prentice-Hall.

McDougall, A. & Betts, J. (1997) Logo Supporting the P – 12 Curriculum in a Technology Immersion School. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 92 - 100.

McDougall, A. & C. Dowling (eds.) (1997) *Learning in Logo Microworlds*, Computing in Education Group of Victoria.

Murnane, J.S. (1991) Models of recursion. *Computers & Education* Vol. 16, 197–201.

Murnane, J.S. (1992) To iterate or to recurse? *Computers & Education* Vol. 19, 387–394.

Murnane, J.S. (2010) *Programming Languages for Beginners.* Saarbrucken, Germany: Lambert. ISBN: 978–383836987–7.

Murnane J. S. and McDougall A. (2006). Bad Computer Science in Beginners Programming Courses: "Considered Harmful"? A case study of the Tufts graphical programming language. In Watson, D. & D. Benzie, D. (eds), *Imagining the future for ICT and Education*. *IFIP WG 3.1, 3.3, & 3.5 Joint Conference*, 26th-30th June, Ålesund, Norway: 329-340. ISBN: 1082-92186-33-6 FISBN: 13978-82-92186-33-6. Published online by Springer and in CD-ROM format.

Murnane, J.S. & Warner, J.W. (2001)An empirical study of secondary students' expression of algorithms in natural language. In: McDougall, A., Murnane, J.S., Chambers, D. (eds.): 7th IFIP World Conference on Computers in Education, Vol. 8. *Computers in Education 2001: Australian Topics*, 81–86. Bedford Park, South Australia: Australian Computer Society.

Nevile, L. (1985) Some Comments on Logo after Ten Years. In McDougall, A. (ed.) (1996*) Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 101 – 108.

Nevile, L. & Dowling, C. (1983) *Let's Talk Apple Turtle*. Sydney: Prentice-Hall.

Newell, B. (1988a). *Turtles Speak Mathematics*. Canberra, Australia: Curriculum Development Centre.

Newell, B. (1988b). *Turtle Confusion*. Canberra, Australia: Curriculum Development Centre.

Oakley, J. (ed.) (1996) *Logo in Australia: Selected Readings*, Computing in Education Group of Victoria.

Oakley, J. & McDougall, A. (1997) Young Children as Programmers: Fantasy or Flight? In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 32 – 51.

Papert, S. (1971a) Teaching Children Thinking. Cambridge, Massachusetts, Massachusetts Institute of Technology. 241 pp. ERIC ED 077 241.

Papert, S. (1971b) Teaching children to be mathematicians vs teaching about mathematics. Cambridge, Massachusetts, Massachusetts Institute of Technology Artificial Intelligence Laboratory. 26 pp. ERIC ED 077 243.

Papert, S. & Solomon, C. (1971) Twenty things to do with a computer. Cambridge, Massachusetts, Massachusetts Institute of Technology. 240 pp. ERIC ED 077 240

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.

Richardson, J. (1997) Logo in Australia: 21 Years On. In McDougall, A. & C. Dowling (eds.) *Learning in Logo Microworlds*, Computing in Education Group of Victoria, 3 – 6.

Ross, P., Howe, J. (1981) Teaching mathematics through programming: ten years on. In: Lewis, R., Tagg, D. (eds.*) Computers in Education Vol. 1*, North-Holland, 143 – 148.

Squires, D. & Sellman, R. (1985) Designing Computer Based Microworlds. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 25 – 30.

Vincent, J. (2001) The role of visually rich technology in facilitating children's writing. *Journal of Computer Assisted Learning* Vol. 17 No. 3, 242 – 250.

Vincent, J. (2002) MicroWorlds and the Integrated Brain. In McDougall, A., Murnane, J. & Chambers, D. (eds.) *Computers in Education 2001: Australian Topics*. Sydney, Australian Computer Society, 131 – 137.

Vincent, J., McDougall, A. & Azinian, H. (2010) Visualisation, multimodality and learning with information technolgy. In McDougall, A., Murnane, J., Jones, A. & Reynolds, N. (eds.) *Researching IT in Education*. Oxford, Routledge, 192 – 199.

Weyer, S.A., Cannara, A.B. (1975) Children learning computer programming: experiences with languages, curricula and programming devices. Stanford, Calf., Stanford University. 228 pp. Technical Report No. 250. ERIC ED 111 347

Wills, S. (1980) Computer Education in Tasmanian Schools: ACS Lecture of the Year. *Australian Computer Bulletin*, Vol. 4, No. 7, 22 – 28.

Wills, S. (1981) Computer Education in Australian Schools. *Australian Computer Bulleti**n***, Vol. 5, No. 4, 4 – 5.

Wills, Sandra (1985) Doodle Design Debug: Process vs Content Issues in Classroom Computing. In McDougall, A. (ed.) (1996) *Logo in Australia: Ten Years On*, Computing in Education Group of Victoria, 1 - 12.

Yelland, Nicola & Jennifer Masters (1997) 21 Years of Logo: Logo for the 21st Century. In McDougall, A. & C. Dowling (eds.) Learning in Logo Microworlds, Computing in Education Group of Victoria, 122 - 132.