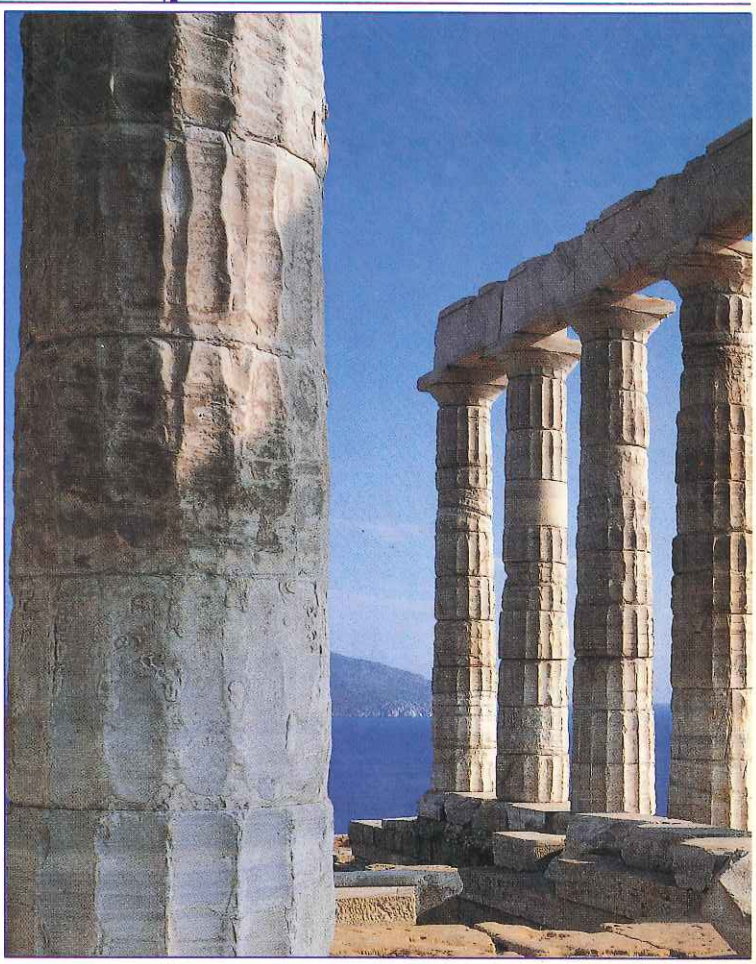
Proceedings
of the Fourth European Logo Conference
University of Athens, Department of Informatics
28-31 August 1993, Athens Greece

Logo-like Learning Environments: Reflection & Prospects



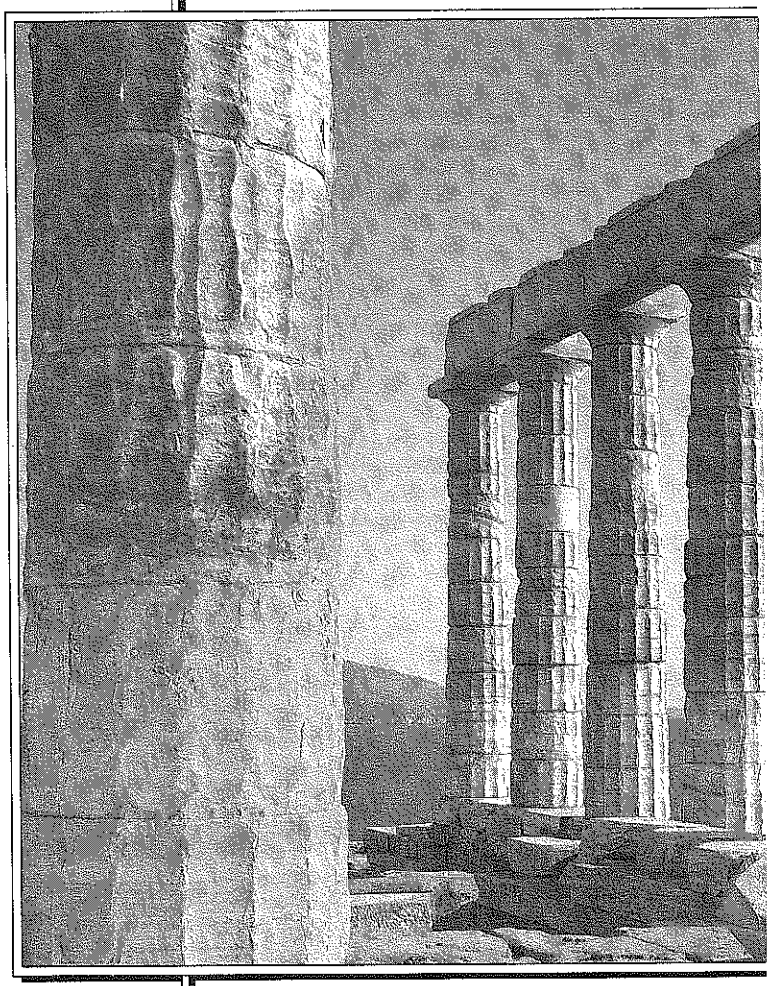
editors:
P. Georgiadis, G. Gyftodimos, Y. Kotsanis, C. Kynigos

DOUKAS SCHOOL



Proceedings
of the Fourth European Logo Conference
University of Athens, Department of Informatics
28-31 August 1993, Athens Greece

Logo-like Learning Environments: Reflection & Prospects



editors:
P. Georgiadis, G. Gyftodimos, Y. Kotsanis, C. Kynigos

DOUKAS SCHOOL

Proceedings
of the Fourth European Logo Conference
University of Athens, Department of Informatics
28-31 August 1993, Athens Greece

Edited by:
P. Georgiadis, G. Gyftodimos, Y. Kotsanis, C. Kynigos

ISBN: 960-7203-01-1

Published & Produced by:
Doukas School S.A.
36-40, Messogion Str.
151 25 Maroussi
Athens, Greece
tel: 6822500, fax: 6818277

Printed by:
Ideographiki Ltd
39-41 Th. Sofouli Str.
154 51 N. Psychiko
Athens, Greece
tel: 6723180, fax: 6475622

ΠΑΙΖΩΝ ΜΗ ΒΙΑ ΔΙΔΑΣΚΕ ΤΟΥΣ ΠΑΙΔΑΣ

ΠΛΑΤΩΝ

International Scientific Committee

J. Arias (Spain), S. Bouillon (Switzerland), E. Calabrese (Italy), S. Close (Ireland), S. Correia (Portugal), D. Dicheva (Bulgaria), M. Doyle (U.K.), J. Enkenberg (Finland), K. Farkas (Hungary), M. Gionannini (Italy), Y. Give'on (Israel), M. Greobelnik (Slovenia), J. L. Gurtner (Switzerland), K. Johnson (U.K.), V. Kaptelinin (Russia), C. Kynigos (Greece), H. Loethe (Germany), C. Montandon (France), E. Neuwirth (Austria), R. Noss (U.K.), C. Parmentier (France), H. Pinxtereen (Netherlands), G. Reuteler (Lichtenstein), G. Schuyten (Belgium), E. Sendova (Bulgaria), J. Sklenar (Malta), M. Valcke (Belgium), J. Weinberger (Czechoslovakia), U. Wittrup (Sweden).

Scientific Organisation
University of Athens, Department of Informatics

The Department of Informatics, University of Athens maintains a keen interest technological applications in Education and holds a permanent seminar in Computers in Education, directed by Prof. G. Philokyprou.

In Cooperation with
the Computer Technology Institute (ITY)
The Greek Computer Society (ΕΠΥ)

Local Scientific Committee

Prof. G. Philokyprou, Prof. D. Maritsas, Prof. M. Hatzopoulos, Prof. S. Vosniadou, Prof. G. Polydoridi, Dr. P. Georgiadis, Dr. V. Makrakis, Dr. M. Grigoriadou, Dr. C. Kynigos, Mr. G. Gyftodimos, Mr. Y. Kotsanis, Ms L. Steriopoulou, Ms. E. Kapnia, Mr. A. Drivas, Mr. C. Kilias, Mr. S. Bakogiannis, Mr. S. Triantafillou.

Scientific Secretary

Dr. C. Kynigos
University of Athens, Department of Informatics
Panepistimioupolis, TYPA Buildings, 157 71, Athens, Greece,
tel: 7291855, fax: 7219561, e-mail: elogo93@grathun1.bitnet

Administrative Organisation

M. Konstantinidou
Triaena Congress
24, Harilaou Trikoupi Str. Athens 106 79 - Greece
tel: 3609511-15, fax: 3607962

Proceedings Organisation

Y. Kotsanis, S. Stefanidis, M. Drimona
Doukas School S.A.
36-40, Messogion Str., Maroussi, 151 25, Athens, Greece
tel: 6822500, fax: 6818277, e-mail: ikotsa@leon.nrcps.ariadne-t.gr

Preface

The Fourth European Logo Conference is taking place at a time when a multitude of conferences on the issue of technology in education reflect an increasing pluralism regarding related technological developments and (perhaps not so often) developments in the educational process. The European Logo community is confronted with perceptions of Logo as something of the past but at the same time with ideas, technological applications and their educational uses very mush based on discourse, argumentation and educational principles generated from within the Logo community and its tight relations to education. This is why we are proposing that within the framework of this conference we discuss the invariant characteristics of Logo-based learning environments and pinpoint what is relevant, within those environments, to the way we use technology in education.

This, of course, is only part of the story. Even though Logo-based educational ideas are relatively mature, they have never ceased to develop, in line with the Logo "philosophy" itself. The days when constructivist learning with computers was almost synonymous to general problem-solving with Logo are rapidly becoming a visionary past. In an attempt to encourage a more focused and situated kind of learning, we are developing both our pedagogy beyond expectations of knowledge transfer from short-term pedagogical "treatments" and our technology beyond the use of a uniquely defined Logo programming environment. What lessons are we learning from our educational research, technological developments and classroom practice? What are the invariants of Logo-like learning environments and the principles for their further development?

The *themes* in which the conference papers have been classified reflect our intent to give an opportunity to participants to discuss amongst them particular interests in depth. We gave the opportunity to all participants to present their work, respected the choice of those authors who specified a theme for their paper, since it reflects the audience they may wish to address and classified the rest in the same spirit. The relatively even distribution of papers amongst themes and the variety of nationalities (we are particularly happy to host participants from 24 countries including the USA., Canada, Brazil, Australia and Japan from outside Europe) will hopefully give rise to discussion on a breadth of issues. The themes were:

a. Learning Process:

collaborative learning in Logo-like environments, understandings based on intuitive knowledge, Logo and metacognitive skills, Logo and modelling, transfer and synthesis between ideas understood in Logo and other environments, children with special needs

b. Domain Specific Learning in Logo-like Environments:

programming, recursion, curriculum content areas (maths, physics, music etc.), control technology, electronic communication, databases

c. The School Interface:

cultural aspects of Logo-like environments in the classroom, teacher intervention in child-centred classrooms, integrating Logo-like environments in curriculum subjects, Logo-based teacher education

d. Logo-like Software Design:

exploratory environments and microworlds, design principles for microworlds and exploratory environments, primitive design

e. Logo-like Programming Environments:

object oriented applications, logic programming, functional programming, parallel processing techniques

The discussion will not be restricted to the use and presentation of Logo -based software, but will invite discussion of exploratory software such as (but not restricted to) Boxer, ISETL, CABRI, etc.

The conference is taking place at a time when the issue of the use of computer technology is receiving special interest from authorities in Greece. In addition to our *plenary* speakers from abroad (Prof. A. diSessa, Prof. C. Hoyles, Prof. E. deCorte and Dr. R. Noss), we have invited a number of Greek researchers (Prof. S. Vosniadou, Prof. D. Maritsas, Prof. G. Polidoridi and Dr. C. Kynigos and Dr. V. Makrakis) who's interests relate to the theme of the conference from a variety of vewpoints and who will outline a picture of the state of affairs in the Greek setting hopefully generating discussion related to what's happening elsewhere.

Three *panel sessions* are also organised in the following themes:

a. Psychological and cultural issues concerning the infusing of Logo-like education in real curricula,

b. Integrating the use of computer technology in Greek compulsory education, and

c. The dialectic between the design of content-specific and general applications.

The relatively large number of plenary addresses and the three panel discussions will no doubt give rise to and reflect the plurality of issues generated from Logo-like education. Hopefully, the conference will be successful enough so that we come out of it with no clear definition - but a lot of new ideas - about what "Logo-like" means...

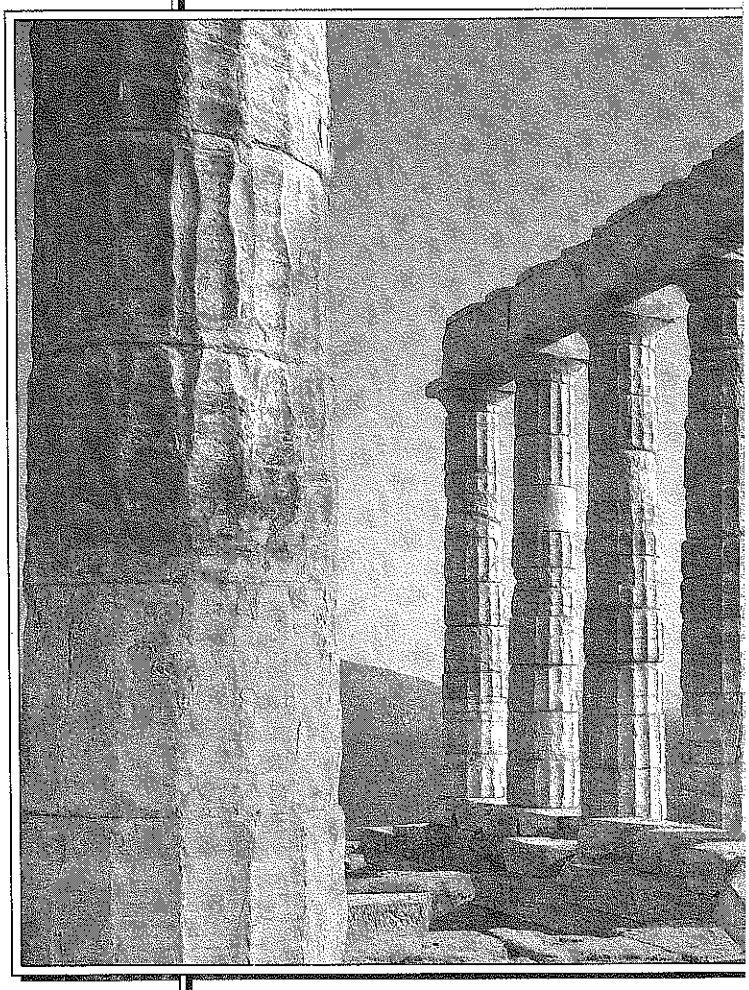
We would particularly like to thank the Director of the Department of Informatics, University of Athens, Prof. G. Philokyprou, for hosting the Conference, our Co-organisers and Sponsors and Yannis Kotsanis from Doukas School for putting endless hours in bringing these proceedings in their present shape and form.

Panayotis Georgiadis, Chronis Kynigos and George Gyftodimos,

on behalf of the Local Scientific Committee

Theme A

Learning Process



A. Learning Process

3 • *Bornholt L., Crawford K., Summers F.*
Learning in LogoWriter Environment: a Hierarchy of Students' Conceptions of Tasks

11 • *Dicheva D., Close S.*
Misconceptions and Mental Models of Recursion

21 • *Doyle M.*
Logo: Windmills, Writing, and Right Triangles

33 • *Elliott A.*
Preschoolers' Logo Play: Characteristics Associated with Logo Competence

41 • *Koukouvinou S., Triantafillou S., Marouka E.*
Logo as an Assessment Tool

47 • *Koutsogiorga I.*
Logo, as a Studying Procedure

53 • *Mladenic D.*
Inducing Knowledge from Examples

63 • *Ripper A.*
Logo Environment in Pre-school: a Mediation Tool

69 • *Suomala J.*
Natural Learning in a Lego-Logo Learning Environment

Learning in a LogoWriter Environment: a Hierarchy of Students' Conceptions of Tasks

Laurel Bornholt, Kathryn Crawford, Fiona Summers

Faculty of Education, University of Sydney,
New South Wales 2006 Australia

Abstract

Primary school students who use LogoWriter were asked to talk aloud during four construction tasks (poem, budget, sociogram, design). Cumulative categories of students' awareness of task behaviour were derived by taking a phenomenographic approach to analysis of the transcripts. A general pattern of transfer of knowledge across content areas was not evident. Students' responses showed individual approaches to each task, rather than consistent patterns due to features of individuals (gender, Verbal IQ, involvment in computing) or types of tasks. Content knowledge and technical knowledge were interwoven in a hierarchy of interpretation of the task, implementation, and reflection on the task.

Keywords: metacognition, task analysis, computer literacy, phenomenography

Laurel Bornholt *is a lecturer in the School of Educational Psychology, Measurement and technology who teaches courses in research methods in education, and adolescent development. Her research is in the area of social cognition and achievement, and social comparisons in academic self-concepts).*

Kathryn Crawford *is a lecturer in the School of Teaching and Curriculum Studies, and is the Director of SMITE (Science and Mathematics Information Technology in Education). Her research is mainly concerned with the process of learning and technology, in mathematics and science education.*

Fiona Summers *was an honours students in the Faculty of Education and is currently teaching primary school students. Her research interests are in teaching and learning, particularly about children's understanding of mathematics.*

Learning in a LogoWriter Environment: a Hierarchy of Students' Conceptions of Tasks

Introduction

In this paper, a phenomonographic approach was taken to understanding learning through 'detailed analysis of learning outcomes derived from tasks set by the researcher' [5, p212] with primary school students who were beginning to use LogoWriter. They were asked to talk aloud as they completed open-ended construction tasks. The focus was on students' awareness of their behaviour, that is, their conceptions of task interpretation and implementation, and their reflections on task behaviour and alternative strategies. The tasks were designed to allow flexible use of content and technical knowledge in the construction of something new.

The research is part of a wider study [7] with teachers and students re-defining school processes in a technologically rich environment. The students habitually used their personal laptop computers, along with pencil-and-paper, for most of their school tasks.
Working with these students was therefore a unique opportunity to derive an analytical framework, which adds considerably to our understanding of student's learning with computers.

Students' technical LogoWriter knowledge and knowledge in the content areas, was made evident as they completed regular school tasks using computers. The students used the technology to construct products in four content areas (language, mathematics, social studies and science). We were interested in the transfer of knowledge [8] when the use of technology was common to all tasks. The question was whether students demonstrate similar categories of awareness of task behaviour across the content areas.

The study explored the interaction between what the students bring to these tasks and what they make of them, thought of in terms of supports and constraints [6]. The task process may be influenced directly by the students' involvement and competence with LogoWriter which the student brings to all four tasks. It is also possible that some students, especially bright students, may prefer, for example, to write their poems with pencil and paper (Crawford, in press). The relations between technical and content knowledge was therefore central to this study.

It was expected that the quality of students' responses would describe an interplay between their understanding of each kind of task, what they bring to the situation, and specific features of each task [1], rather than explanations of task behaviour which relied on an underlying ability as a feature of the student. The transfer across tasks or generalising of knowledge as a 'unitary phenomenon' seemed unlikely in such a complex context [8, p115].

The role of the researcher was initially to elicit the students' self-knowledge, and then to develop an analytical framework to integrate students' expressions of task and technical self-knowledge. While our ideas or implicit theories are acknowledged [4], the categories of student awareness were derived from the transcripts.

Asking the students to talk aloud during the task is taken to be an expression of meta-cognition. Idol, Jones and Mayer (1991) have suggested that meta-cognition has been redefined as "shared behaviour (thinking aloud), and it includes the learners' beliefs, judgements, attitudes, motivation, and self-concept" (p.73). Because language reflects student's construction of meaning and assists in the process of tackling the task [9], it was expected that self-talk would be as important for distinguishing individual differences in the formation of plans as for monitoring their reflections on thoughts and actions.

Method

Design A sample of eight students (age 11 years) completed four tasks (poem, budget, design and sociogram). Students were chosen by Verbal IQ (WISC-R), involvement in computing (interview), and gender. Categories for students' awareness of behaviour were derived from their explanations of how they went about each task, in terms of technical and content knowledge.

		Involvement in computer use	
		High	Moderate
Verbal IQ	High	girl	girl
		boy	boy
	Average	girl	girl
		boy	boy

Table 1. Design for characteristics of students

Procedure Students worked individually on the tasks, and the interviewer prompted the student to talk aloud (eg 'What happened there?' 'Can you say what you're doing now?').
Analysis A phenomenographic analysis was based on the transcriptions of the interviews. The researchers negotiated by consensus on the emerging categories of description of students' awareness of task behaviours, and the classifications of students' responses.

Results and Discussion

Analyses developed for each task were inclusive of the students' technical knowledge of Logo procedures and knowledge of the content area. Although some tasks elicited more information than others, separate structures were redundant. Five characteristics of task behaviour were used (planning, the steps taken, response to mistakes, describing procedures, and explanations) to build the hierarchy from a simple understanding of what the task is, what is required, procedural initiative to start the task, completing the task, and reflecting on it, to alternative task strategies. In the hierarchy of students' awareness of task behaviour in Table 2, each category is inclusive of previous categories.

Category 1	Understands what the task is
Category 2	Can start the task
Category 3	Can decide on how to do the task
Category 4	Can describe the task process
Category 5	Can explain the language, processes and procedures used to do the task
Category 6	Can use technical and content knowledge to facilitate task completion
Category 7	Can think of other ways of doing the task

Table 2. Hierarchy of awareness of task behaviour

The hierarchy described students' awareness of technical and content knowledge, and, given that the tasks were set by the researcher, considerations of the technology seem to have been subordinated to their conceptions of the task (e.g. category 1 was in terms of content area knowledge). Content and technological knowledge were interwoven in a structure of task implementation, refections on the task process or task awareness, and task alternatives. There is a sense in which the content area knowledge predominated even in the categories where students reflect on their actions. Unless students imagined alternative possibilities for the task, they did not talk about or demonstrate alternative technological solutions.

The hierarchy presents qualitative shifts in the focus of student awareness, from task interpretation (Category 1); implementation as starting, doing and describing the task (Categories 2 to 4); to reflection, facilitating, and alternative task strategies (Categories 5 to 7). However, the classification of students' awareness of task behaviour shown in Table 3 demonstrate considerable variation across the four tasks for individual students.

			Construction tasks			
			POEM	BUDGET	ROBOT	SOCIOGRAM
Personal characteristics						
Verbal IQ	Involved	Gender				
High	High	Girl	1	5	7	1/2
High	High	Boy	3	5	1	4
High	Moderate	Girl	4	1	2	3
High	Moderate	Boy	3	5	1	4
Average	High	Girl	6/7	7	7	7
Average	High	Boy	7	7	6	7
Average	Moderate	Girl	2	3	2	7
Average	Moderate	Boy	4	7	7	4

Table 3. Categories of student's awaremess of task behaviour in four content areas

Given the variation in students' responses, the robust structure of the categories is a significant result. The categories of students' responses ranged from 1 to 7 on all tasks, and their responses cannot be explained simply in terms of features of the students (boys and girls, their involvement with computers, and their cognitive ability) or in terms of systematic differences between the four tasks. One might speculate about each student's level of motivation for each type of task, although it was evident that all students could express themselves well in this context.

The inclusive nature of the higher categories is clearly illustrated in the case of Robbie's explanations at Category 7 as he writes poems. He understood the task

> You think of words that rhyme and try to put them in sentences... (page 2 line 25)

completed it, and then described another type of poem

> Well this is an acrostic poem... and you write letters down the side and you write a sentence to each letter...
>
> (page 3 line 15)

He completed this task and explained the processes and procedures he used (page 1 line 11)

> I'd first get into LogoWriter then get a new page, and CONTROL/UP to get up to the top of the page and write the poem

used technical knowledge to facilitate task completion (page 4 line 16)

> Selecting that so that I don't have to write it all the time.
> [How do you do that?]
> Oh, you F1 and then you use the arrow keys, and PAGE UP/PAGE DOWN and say I wanted to write that, and I didn't write it, all I'd have to do is press PASTE

and thought of other ways to do the task (page 2 line 36)

> Can you tell me how you would do it [write a poem]? I'm just getting into LogoWriter [types...] and get into a new file in Language, and you name the page...
> [What might you do ...?]
> You think of words that rhyme and try to put them in sentences....
> [Is there another way to do that?]
> You could think of a sentence and try to put a rhyming word on the end of it.

There was some evidence to suggest that students' use of LOGOWRITER opened up possibilities for them as they worked. For example, Darren in category 5 changed his method of calculating the costs of the party budget, and the way he displayed the information, as he completed the task. He understood the task (page 2 line 2)

> Yes, it's when you write down the things you need and how much it will cost

he completed the task (page 2 line 23)

> How do you know how much you're going to spend?
> Oh, just add it up

```
    3.00
+   7.80
+   5.00
+   3.00
   ------
  $18.80
```

described what he did (page 2 line 12,14)

> Just add it up - and you might need two chips and two bottles... Figure out how many people are going to your party and then times it by two or three

and the language, processes and procedures he used to do the task (page 2 line 25-27)

```
1.00        2.60
* 3         * 3
```

> [What are you doing now?]
> Just writing a sort of table of how you'd work it out.

He used technical knowledge to facilitate task completion

> What are you doing now? [typing in the command centre]

Just telling the computer to write the answer out

```
2.60 * 3 = [7.8]
```

It was difficult to determine whether students' conceptions of the task limited their technical behaviour, or vice versa. For example, Jodi Category 3 on the Sociogram task understood the task

> Claire's my best friend but she doesn't go to this school... Now what? ...Print it out and draw lines between them... I printed 'Sociogram' on it too. I don't need my computer for that. I'll just have to go and get a pencil.... (page 3 line 51)

completed the task and described it (page 4 line 17)

> Fiona likes Jane, but Jane doesn't like Fiona. I'll put friendly and unfriendly.... and I'll put 'not sure' for Claire to Angela... And we're best friends. There.

and the language, processes and procedures used to complete the task (page 4 line 57)

> Save text as 'Sociogram Text'. No that would be too long to remember... 'Sociotext'.

What is evident, however, is that the order of the categories listed in Table 2 was consistent. For tasks at lower categories, students could explain what the task was satisfactorily but did not go further. For example, Lisa in Category 1 understands what a poem is

> You think about what it's going to be about, and then you brainstorm and jot down your ideas, and then you try and get them to rhyme and just keep going like that (page 1 line 19)

but could not start the task (page 1 line 25,42)

> [Could you write a poem..."?]
> I don't think I could ... I have to think about it
>
> [We'll come back to your poem]
> Oh, No.

In the context of this study, students did not tend to exhibit higher technical skills than their conception of the demands of the task or their expectations of the quality of response required. For example, Steven doing the Budget task in category 2 could understand the task (page 2 line 3)

> I'm just changing the shapes [dot in drawtool]
> [What are the lines for?]
> Columns...
> [Tell me what you're doing her]
> Putting all the headings
> Drinks Food Other

but could not complete the task (page 3 line 24)

> What would you do if I asked how much it would cost you?
> Umm - don't know.

He didn't recognise the mathematical operation required, which he may probably could not program.

At middle categories, students could analyse the task, but were unable to explain the procedures in relation to the task as a whole. For example, Steven in Category 4 for the Sociogram task clearly understood the

task and completed it, but used a somewhat erratic trial-and-error method of programming. His imagination of the task exceeded his technical competence.

> Dad doesn't know Chris .... But how am I going to fit them all in?' (page 5 line 30).

Although, he could separately explain the processes and procedures used, (page 5 line 42)

> Can I use DRAWTOOL? I'll use a dot - it's better , It's like different brushes ...then I've got to write on them

he had a fragmented view of LogoWriter as commands that tell him how to do separate parts. His early attempts were unsuccessful when he used RG to erase a line and erased everything and said (page 6 line 1)

> 'Coz when you clear graphics all the writing goes, and I forgot and I thought it was text

Conclusion

This study revealed an ordered hierarchy of categories which described students' thinking aloud during open ended construction tasks. The order of the hierarchy was consistent, regardless of the task content domain, or specific features of the students (Verbal IQ, gender, or involvement with computers). In the context of the task being set by the researcher, and regardless of the content area, levels of technical awareness were demonstrated through students' task awareness. We might speculate whether this relationship between tasks and technology would hold if the tasks were embedded in play activities, or were instigated by the students.

The hierarchy of categories suggested that students' task awareness (both content and technical knowledge) can be thought of in terms of the students' conceptions of what the task is, how it might be completed and reflected upon, and the generation of alternative task strategies. The study supports the view that students' meta-cognition about their task behaviour in this context cannot be based on a unitary phenomenon of task knowledge. These case studies revealed the complexity of the interweaving of technical and content knowledge during task completion. Furthermore, it is likely that for some students, that during the process of using of the computer with LogoWriter new task strategies evolve.

It is apparent that students' use of computing technology is a fertile research base in which to explore student awareness during tasks. While awareness of task behaviour is inherent in the method of talking aloud for the researcher's benefit and in reflecting on their own behaviour, there is also a sense of them "speaking" to the computer to bring about an action. The screen display also provides an external image for reflection. Further research is needed to determine to what extent the richness of students' task reflections is due to the nature of the technology and due to the nature of the reflective methods used in the study.

Acknowledgements

To students and staff and the school for their co-operation, and to Helga Rowe and Irene Brown of the Australian Council for Educational Research, and to Sam Ball at the University of Sydney, for their support

REFERENCES

[1] Biggs, J B and Collis, K F (1982) *Evaluating the quality of learning, the SOLO taxonomy (Structure of the Observed Learning Outcome)*, Academic Press.

[2] Crawford, K P (in press) From the Outside Looking In, to appear in *Sharing Perspectives*, a published report on the Pupil Autonomous Learning with Microcomputers PALM Project, B. Somekh (Ed.), East Anglia University, UK.

[3] Idol, L, Jones, B F, and Mayer, R E (1991) Classroom Instruction: The Teaching of Thinking, in L Idol and B F Jones (Eds.) *Educational Values and Cognitive Instruction: Implications for Reform*, Erlbaum, New Jersey, pp.65-119.

[4] Lather, P (1992) *Getting Smart*, Harvard University Press.

[5] Marton, F, Houndsell, D, and Noel Entwistle (1984) *The Experience of Learning*, Scottish Academic Press, Edinburgh.

[6] Norman, D (1988) *The Psychology of Everyday Things*, Basic Books, New York.

[7] Rowe, Helga (1992) *Learning with personal computers*, Australian Council for Educational Research. [Pre-publication report, Learning with microcomputers: Issues, Observations and Perspectives, April, 1992].

[8] Salomon, G and Perkins, D N (1989) Rocky Road to Transfer: Rethinking Mechanisms of a Neglected Phenomenon, *Educational Psychologist*, 24, 113-142.

[9] Vygotsky, L S (1978) *Mind in Society*, Cambridge, Harvard University Press.

Misconceptions and Mental Models of Recursion

Darina Dicheva[1], **Seán Close**[2]

[1] Dept. of Computer Science,
University College Dublin,
Dublin 4, Ireland
e-mail: ddicheva@ccvax.ucd.ie

[2] St. Patrick's College,
Dublin 9, Ireland
e-mail: 75901668@dcu.ie

Abstract

This paper presents a study of children's difficulties in learning the concept of recursion in Logo. A number of incorrect mental models of recursion identified in the study are classified and described. Misconceptions underlying the erroneous mental models are presented. Possible reasons for adopting these misconceptions are put forward.

Keywords: mental models, recursion, Logo,

Darina Dicheva *is Associate Professor of Computer Science at University of Sofia, Faculty of Mathematics and Informatics. Her research interests include artificial intelligence and education, especially intelligent tutoring systems and learning environments, and methodological issues of teaching informatics and using computers in teaching school mathematics. Currently, she is a Visiting Fellow at University College Dublin.*

Seán Close *is Lecturer in Mathematical Education at St. Patrick's College, Dublin. He is also Director of the St. Patrick's College Logo Enrichment Project which has been providing extra-curricular problem-solving courses in Logo for high-ability children since 1985.*

Misconceptions and Mental Models of Recursion

1 Introduction

Recursion is a fundamental principle of programming. Using recursion many complex computational problems can be efficiently and elegantly solved. On the other hand, novice programmers experience unusual difficulty with recursion. A number of studies have examined the difficulties and misconceptions of school and college level students in understanding recursion. Kahney [4] identified mental models college students used to represent the behaviour of procedures in SOLO, an AI programming language. The two dominant models were the "Loop" model and the "Copies" model. Kahney found that approximately 50% of novices possessed the "Loop" model of the recursive process while 90% of experts possessed the "Copies" model.

Logo has been widely used to teach recursion to students from 10 years of age upwards to graduate students. Kurland and Pea [5] studied a group of 11 - 12 year-olds' mental models of embedded recursion. The 7 children were asked to interpret brief recursive programs using hand-simulation. The predominant model was a misguided "looping" interpretation of Logo recursion which persisted even in the face of contradiction. Lee and Lehrer [6] found that adult students, with previous experience of programming in Basic, were more likely to develop a "loop" model of recursion in Logo than were those with no prior programming experience.

A number of researchers have attempted to classify the misconceptions underlying inadequate and inappropriate mental models for programming ideas in a number of computer languages, e.g. Du Boulay [2]- Basic, Pascal; Bayman and Mayer [1] - Basic; Pea [7] - Logo; Sleeman, Putnam, Baxter and Kuspa [9] - Pascal; Soloway, Bonar and Ehrlich [10] -Pascal; Lee and Lehrer [6] - Logo; Scherz, Goldberg and Fund [8] - Prolog. Some sources of misconceptions were identified including: misapplication of analogy, overgeneralisation, natural language problems, intentionality, previous experience problems, and scope problems. In general, it appears that novices, in addition to lack of adequate understanding and facility with the ideas and techniques of the computer language, often ascribe to the computer a kind of intelligence which enables it to ignore inconsistencies, infer intentions of the programmer, understand the context of the programme and generally behave as an average person would in responding to a written set of instructions.

The present study is one of two studies in which the authors seek to build on, and extend, the existing knowledge base concerning children's learning of the concept of recursion. Accordingly the goals of the study are:

(1) To analyse children's interpretations of a range of recursive Logo procedures.
(2) To identify and classify the mental models of recursion used by the children.
(3) To identify the misconceptions underlying incorrect mental models and the possible sources of these misconceptions.

2 Method

2.1 Subjects

The subjects of study were 29 highly able and motivated children, aged 10 to 14 years attending extra-curricular Saturday courses on problem-solving with Logo. Each course runs for 10 Saturdays (2.5 hours each Saturday) with one child per computer and one tutor per 12 to 15 children. The children qualified for the courses by achieving a score of 95th percentile or better on the Raven's Progressive Matrices. The children's Logo experience varied from 1 course (25 hours) to 6 courses (150 hours) with an average of about 3 courses (75 hours) per child; 8 of the 29 children were female.

Very little formal teaching takes place during the courses except when a new concept is introduced. For the most part children work away on their own solving problems set by the tutor. A certain amount of discussion takes place among the children about their solution attempts. The children are encouraged to work independently and to call for the tutor's help only when they are completely "stuck". The tutor questions and provides hints rather than telling the children what to do.

2.2 Procedures

The children were divided into 3 groups on the basis of their experience and progress with Logo: a beginner's group (12 children), an intermediate group (14 children), and an advanced group (3 children). Prior to the present course the beginners had no experience of recursion apart from tail recursion. They were able to define procedures with parameters, and understood iteration and procedure interaction. The intermediate and advanced groups had some limited experience of embedded recursion. As the present course focused on recursive graphics and list-processing it provided the opportunity to study the children's developing knowledge of recursion.

A series of 7 worksheets involving tasks calling for the interpretation of short recursive Logo-procedures was administered to the children during the course. Worksheets 1 and 2 were designed to test the children's understanding of prerequisite ideas and processes including:

- flow of control in a procedure
- flow of control in interacting procedures
- STOP and END commands
- iteration
- local variables
- scope of variables
- tail recursion

Worksheets 3, 4, 5, 6 and 7 tested knowledge of embedded recursion in a variety of contexts. Sample tasks from each of the 7 worksheets are given below.

Worksheet 1 (6 tasks; variables)

If you type in `HOUSE 200` will this program draw a house? If not why not?

```
TO HOUSE :S
SQUARE :S
FORWARD :S RIGHT 30
TRIANGLE :S
LEFT 30 BACK :S
END
```

```
TO SQUARE :A
REPEAT 4 [FORWARD :A RIGHT 90]
END
TO TRIANGLE :B
REPEAT 3[FORWARD :B RIGHT 120]
END
```

Worksheet 2 (2 tasks; tail recursion - graphics)

Make a rough drawing of what this command will do:

`PATTERN1 20`

```
TO PATTERN1 :S
IF :S > 300 [STOP]
FORWARD :S
RIGHT 90
PATTERN1 :S + 20
END
```

Worksheet 3 (2 tasks; embedded recursion - graphics)

Look at the procedure `PATTERN2` given and make a prediction of what the command `PATTERN2 200` will do. Explain your suggestion.

```
TO PATTERN2 :S
IF :S < 20 [STOP]
REPEAT 4[FORWARD :S RIGHT 90]
PATTERN2 :S/2
REPEAT 4[FORWARD :S LEFT 90]
END
```

Worksheet 4 (3 tasks; embedded recursion - words & lists)
Look at the procedure MYSTERY given below and make a prediction of what the command MYSTERY 7 will do.

```
TO MYSTERY :N
IF :N = 0 [PRINT "AH! STOP]
MYSTERY :N - 1
PRINT :N
END
```

Worksheet 5 (2 tasks; embedded recursion - words & lists)
Look at the procedure PATTERN3 given below and make a prediction of what the command PATTERN3 "RABBIT will do:

```
TO PATTERN3 :W
IF EMPTY? :W [STOP]
PRINT :W
PATTERN3 BF :W
PRINT :W
END
```

Worksheet 6 (2 tasks; embedded recursion - graphics)
Look at the procedure SHAPE given below and make a prediction of what the command SHAPE 20 will do. Explain your suggestion.

```
TO SHAPE :S
IF :S > 200 [STOP]
FORWARD :S RIGHT 90
SHAPE :S + 20
LEFT 90 BACK :S
END
```

Worksheet 7 (2 tasks); more complex recursion - graphics)
Look at the procedure SHAPE given and make a rough drawing of what the command SHAPE 180 2 will do.

```
TO SHAPE :S :N
IF :N = 0 [FORWARD :S STOP]
SHAPE :S/3 :N-1
LEFT 60
SHAPE :S/3 :N-1
RIGHT 120
SHAPE :S/3 :N-1
LEFT 60
SHAPE :S/3 :N-1
END
```

Where the children's reasoning was not clear from the worksheet, an individual interview was conducted. All the worksheets were completed over the period of the course when they had been introduced to embedded recursion.

3 Results

The interpretations of the recursive procedures of the 29 children were analysed and a number of misconceptions were identified. Possible reasons for the adoption of these misconceptions are put forward. In most cases the reasons given are similar to ones already discussed by others in relation to some common bugs and misconceptions that novice programmers acquire, i.e. previous experience problems [11], natural language problems [2, 5, 11], overgeneralisation [2, 3], misapplication of analogy [2], intentionality [7].

3.1 Misconceptions in interpreting recursive procedures

Note: (i) For the purposes of the present discussion we call the block of commands before the recursive call the head-block and the block of commands following it - the tail-block (one recursive call only); (ii) Misconceptions are labelled MC1, MC2 etc and suggested reasons, R1, R2, etc.

3.1.1 Misconceptions about the STOP command

MC1 STOP stops the main recursive procedure.

R1 Previous experience confusion - Since the children's first experience of the STOP command is in the context of tail-recursion where it appears to halt the main program they assume it does the same thing in an embedded recursive procedure.

R2 Another misconception confusion - Children possessing MC4 consider a recursive procedure to be a single object instead of a series of procedure "copies" and so STOP would stop its execution.

MC2 STOP stops all currently active procedures.

R3 Overgeneralisation - Follows from R1 in MC1. Children overgeneralise the fact that STOP stops the main program in a tail recursive case.

R4 Natural language confusion - Children use the natural language meaning of STOP, i.e. it halts everything.

MC3 The STOP command does not stop the recursive procedure. It is seen as part of an IF statement which acts as a check (filter) on the input values.

R2 Another misconception confusion - Occurs as a consequence of adopting MC5 about the way recursion works.

3.1.2 Misconceptions about the recursive call

MC4 A recursive call is a looping construct similar to Pascal's WHILE loop. The head-block of the procedure is the loop body. The loop is executed until the STOP command in the stopping rule halts it.

R1 Previous experience confusion - This misconception has its origin in children's experimenting with tail recursion which is Logo's analogue for a WHILE loop.

R3 Overgeneralisation - The children overgeneralise the correct model of flow of control of tail recursive procedures.

MC5 The recursive call is a call for executing the head-block. Thus it says that the commands of the head-block should be executed just once more; the recursive call itself wouldn't be executed again.

R3 Overgeneralisation - The head-block is treated as having the status of a procedure. The recursive call is treated as an ordinary procedure call for executing the head-block.

MC6 The recursive call just evaluates its own input and does "what the procedure is expected to do", i.e. it moves the turtle forward in the case of graphics procedures, or prints the input, in the case of procedures manipulating words and lists.

R5 Intentionality - Program seen as intentional 'being' with the ability to look forward or back in the procedure.

3.1.3 Misconceptions about the local variables

MC7 In a Logo program (a set of procedures) there is created only one variable with a given name.

R4 Natural language confusion - This misconception is based on the belief that Logo "understands" the

natural language meaning of the variables' names such as SIDE, ANGLE, LENGTH and so on, thus whenever and wherever one uses them throughout the program Logo would understand the same thing.

MC8 A recursive procedure maintains only one variable with a given name. At each iteration the variable can be assigned a new value specified by the input in the recursive call.

R2 Another misconception confusion - The misconception can follow as a consequence of adopting misconception MC7.

R2 Another misconception confusion - If children accept that every single procedure has its own variables they might be confused in the case of a recursive procedure. This might follow from adopting misconception MC4 or MC5, i.e. seeing a recursive procedure as a single object instead of a series of procedure "copies".

R1 Previous experience confusion - Some children believe that if Logo calls the same procedure this procedure should use the same variables. This has proved to work in their experimenting with tail recursive procedures.

MC9 A procedure call is interpreted as: MAKE commands changing the values of the variables involved, and a command for calling the procedure, for example,

CIRCLES :R/2 is interpreted as MAKE "R :R/2 CIRCLES :R

This misconception about the general procedure calls is also applied to the recursive calls.

R5 Intentionality - As in R5 for MC6.

R1 Previous experience confusion - Both of the above variants have produced the same correct result in tail recursive procedures.

MC10 Logo "remembers" the initial value of a variable, for later use in the commands following the recursive call, i.e. in the tail-block.

R6 Misapplication of analogy - The "box" metaphor for variables may mislead the children - if more than one thing could be put in a box why not store more than one value in a variable.

R3 Overgeneralisation - The children tend to treat the head-block of a recursive procedure as a unit having the status of a procedure. Thus, if the child has a correct concept about local variables she would imagine that the head-block would have its "private" variable whose value is changed at each iteration as specified in the recursive call but leaving the head-block (going to the tail-block) Logo will use the "old" variable, i.e. will use the variable's initial value.

MC11 A variable holds two values - its initial (input) value and a "current" value which is changed as specified in the procedure. Logo can use both of them.

R6 Misapplication of analogy - As per R6 in MC10.
MC12 The input specified in the recursive call is evaluated and used (at the next iteration) but is not "stored" as a new value of the corresponding variable. So, after it is used the evaluated value is not available any more.

R2 Another misconception confusion - Follows from a combination of MC5 with MC4.

3.2 Mental Models of Recursion

In our study we found several common incorrect models of recursion possessed by the children. In this section a brief description of these models is presented as well as the misconceptions on which they are based.

The given examples are based on interpreting the following procedures:

```
TO PATTERN3 :W
  IF EMPTY? :W [STOP]
  PRINT :W
  PATTERN3 BF :W
  PRINT :W
END
```

```
TO PATTERN2 :S
  IF :S < 20 [STOP]
  REPEAT 4 [FORWARD :S RIGHT 90]
  PATTERN2 :S/2
  REPEAT 4 [FORWARD :S LEFT 90]
END
```

3.2.1 Loop Models of Recursion

The Loop model is commonly identified as an incorrect model of recursion possessed by novice programmers [4, 5, 6]. This model is usually described as a view of recursion as looping and never discussed in detail. In fact the so called Loop model could be considered as a family of mental models of recursion. We describe below several loop models of recursion which we observed to be possessed by the children. A general characteristic of all members of this family is that recursion is seen as a looping construct similar to Pascal's WHILE loop. The recursive call modifies the values of the procedure parameters and passes control back to the beginning of the head-block. This continues until the STOP command in the stopping rule causes exit from the loop.

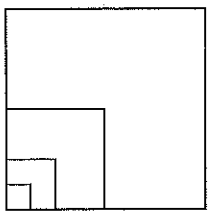
"Procedure Cutting Loop" Model

The head-block is executed as a WHILE loop. The tail-block is not executed at all.

The recursive call passes the control to the beginning of the recursive procedure. The loop is executed until the STOP command in the stopping rule causes a complete halt of the recursive procedure. The commands following the recursive call are never executed.

Interpretations of `PATTERN3 "LEGO` and `PATTERN2 100` based on this model predict the following result:

```
LEGO
EGO
GO
O
```



This model is based on misconceptions **MC1, MC2,** and **MC4**.

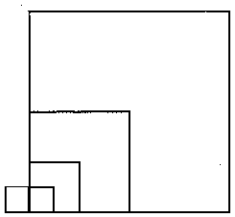
In the next two Loop models the loop is executed until the STOP command in the stopping rule causes exit from it and the control is passed to the command following the recursive call. Then the tail-block is executed and END halts the execution of the recursive procedure.

"Proper Loop" Model

The head-block is executed as a WHILE loop. The tail-block is executed just once. In the tail-block the variables have their last assigned values.

Interpretations of `PATTERN3 "LEGO` and `PATTERN2 100` based on this model predict:

```
LEGO
EGO
GO
O
O
```



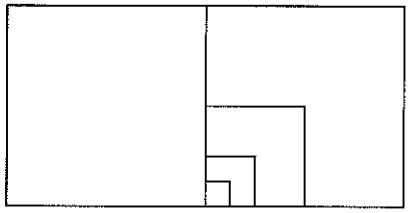
This model is based on misconceptions **MC4, MC7, MC8,** and **MC9**.

"Embedded Loop" Model

The head-block is executed as a WHILE loop. The tail-block is executed just once. In the tail-block the variables have their initial (input) values.

Interpretations of `PATTERN3 "LEGO` and `PATTERN2 100` based on this model predict:

```
LEGO
EGO
GO
O
LEGO
```



This model is based on misconceptions **MC4** and **MC10.**

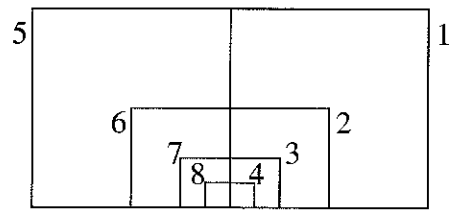
A variation of the "Embedded Loop" model was also observed, incorporating incorrect interpretation of the variables in the head-block - one variable is interpreted as holding two values - its initial (input) value and another value, which is changed by each successive recursive call.

"Two Loops" Model

Both the head-block and the tail-block are executed as WHILE loops. Both loops start with the initial values of the variables.

Interpretations of `PATTERN3 "LEGO` and `PATTERN2 100` based on this model predict:

```
LEGO
EGO
GO
O
LEGO
EGO
GO
O
```



Note: Numbers indicate the order of drawing the squares

This model is based on the misconception MC10.

3.2.2 "Do-it-once-more" Models of Recursion

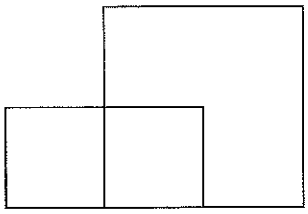
In the models of this family the recursive call says that the commands of the head-block should be executed just once more. So, when it reaches the recursive call, Logo goes back to execute the head-block once again; the recursive call wouldn't be executed again because it was just executed (by running the above commands a second time). Next, the tail-block is executed. END ends procedure execution.

"Proper Do-it-once-more" Model

The head-block is executed twice, and the tail-block - once. When executing the tail-block the variables have the last assigned values.

Interpretations of `PATTERN3 "LEGO` and `PATTERN2 100` based on this model predict:

```
LEGO
EGO
EGO
```



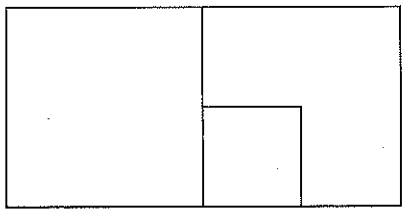
This model is based on misconceptions **MC5, MC7, MC8,** and **MC9.**

"Embedded Do-it-once-more" Model

The head-block is executed twice, and the tail-block once. When executing the tail-block the variables have their initial (input) values.

Interpretations of PATTERN3 `"LEGO` and `PATTERN2 100` based on this model predict:

```
LEGO
EGO
LEGO
```



This model is based on misconceptions **MC3, MC5** and **MC10.**

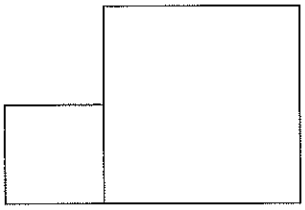
3.2.3 Other Models of Recursion

"Simple" Model

The recursive call is nothing to do with iteration. It just evaluates its own input and possibly either moves the turtle forward (in case of graphics procedures) or prints the input (in case of word and list processing).

Interpretations of `PATTERN3 "LEGO` and `PATTERN2 100` based on this model predict:

```
LEGO
EGO
```



This model is based on misconception **MC6.**

4 Summary

The present study identified a number of models belonging to the "Loop" class of models. It also identified another class of erroneous mental models, the "Do-It-Once-More" models. The children's interpretations suggested that these erroneous models were based on one or more misconceptions concerning the flow of control and the use of local variables in recursive procedures. The origins of these misconceptions appeared to be similar to those identified by previous authors (previous experience confusion, overgeneralisation,

intentionality, natural language confusion, misapplication of analogy). The ways in which the reasons for the misconceptions were linked to the misconceptions themselves were clarified.

The study also confirmed the prevalence of "Loop" models as a class of erroneous mental models used by novice programmers to interpret recursive procedures. Of the 140 interpretations of embedded recursion procedures made by the 29 children approximately 40% were interpreted using the correct model of recursion, about 35% using a "Loop" model, about 10% using a "Do-It-Once-More" model, about 10% using some other model, and about 5% where there was no attempt or the model was indeterminate.

From a learning/teaching point of view the results suggest that many novice programmers get too much experience with tail recursion before going on to embedded recursion. An earlier introduction of embedded recursion could alleviate this problem. Systematic efforts to clarify the role of STOP, the recursive call, and local variables should also help substantially. Following the second part of the study, i.e. the analysis of the children's errors in constructing embedded recursive procedures it is hoped to discuss the implications for teaching strategies in greater detail.

Note

The study reported here is part of the Logo Enrichment Project of St. Patrick's College

REFERENCES

[1] Bayman, P., and Mayer, R.E. (1983) A diagnosis of beginning programmers' misconceptions of Basic programming statements. *Communications of the ACM, 26(9),* 677-679.

[2] Du Boulay, B. (1986) Some difficulties of learning to program. *Journal of Educational Computing Research, 2,* 57-73.

[3] Johnson, L., and Soloway, E. (1985). PROUST: Knowledge-based program understanding. *IEEE Trans. Soft. Eng. 11(3),* 267-275.

[4] Kahney, H. (1984) Modelling novice programmer behaviour. In: Yazdani, M. *New horizons in educational computing.* Ellis Horwood Ltd., 101-118.

[5] Kurland, D.M., and Pea, R.D. (1989) Children's mental models of recursive Logo programs. In: Soloway, E., Spohrer. *Studying the novice programmer,* LEA, 315-323.

[6] Lee, O., and Lehrer, R. (1988) Conjectures concerning the origins of misconceptions in Logo. *Journal of Educational Computing Research, 4(1),* 87-105.

[7] Pea, R.D. (1986) Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research, 2,* 25-36.

[8] Scherz, Z., Goldberg, D., and Fund, Z. (1990) Cognitive implications of learning Prolog — mistakes and misconceptions. *Journal of Educational Computing Research, 6,* 89-110.

[9] Sleeman, D.H., Putnam, R.T., Baxter, J., and Kuspa, L. (1986) Pascal and high school students: A study of errors. *Journal of Educational Computing Research, 2,* 5-23.

[10] Soloway, E., Bonar, J., and Ehrlich (1983) Cognitive strategies and looping constructs: An empirical study, *Communications of the ACM, 26(11),* 854-860.

[11] Spohrer, J.C., and Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM, 29(7),* 624-632.

Logo: Windmills, Writing, and Right Triangles

Mike Doyle

37 Bright Street, BD23 1QQ SKIPTON,
North Yorkshire, UK
Honorary Chair Logos, PO Box 43,
Houghton-on-the-Hill, LE7 9GX, United Kingdom

Abstract

In the United Kingdom, Logo, the high ratio of computers to children notwithstanding, has been subsumed to, concrete, turtle graphics. The messianic aspirations of Mindstorms have been discarded, the "Logo Philosophy" has become a totem. The paper analyses the process of assimilation into the extant culture of Logo. It is suggested that accomodation to the computer, as a medium, and to Logo, as a language of that medium, entails a radical re-evaluation - from the educational viewpoint - of both. A visualisation, the language cone, is used to place the computer within a written language continuum. From this viewpoint, Logo becomes an educational computer language rather than a computer environement: a means of expression intrinsic to the medium instead of one for exploration. The English national curriculum, developed over the past decade, is then analysed for signs that the world of digital language has begun to pervade educational thinking. That this analysis proves negative raises serious issues concerning the introducing computers into school. Specific suggestions are made for the insertion of growth points for a digital curriculum.

Keywords: education, natural language, computer language, Logo, computers, children, curriculum.

***Mike Doyle**, (Mícheál Ó Dúiohil) an Englishman if Irish extraction, is a psychology graduate who has taught children with learning difficulties for the past two decades. His interest in computers and language began in 1982 when microcomputers first reached British schools. He was awarded an MPhil by Manchester University in 1986 for research into overlay keyboards, computer speech, and turtles. In 1987 he was elected Honorary Chair of the British Logo User Group (now Logos). He has been a regular contributor to Logo conferences in Europe and is editor of EUROLOGOS.*

Logo: Windmills, Writing, and RightTriangles

Introduction

A decade ago, Logo left the MIT Logo Lab and came to school. It came on two early microcomputers: the Apple II and the TI 99/A. Two individuals guided these implementations: Hal Abelson and Seymour Papert. Papert was an academic, Piagetian, mathematics educator; Abelson a computer scientist. Papert was a utopian; Abelson an engineer. Both authored books: Seymour Papert, *Mindstorms*: Children, Computers, and Powerful Ideas (1980); Harold Abelson with Andrea diSessa, *Turtle Geometry*: The Computer as a Medium for Exploring Mathematics (1981). The two books could not have been more different. Papert's was messianic, with visions of computers rendering formal thinking more accessible and schools unnecessary. Abelson and diSessa provided a college course in a new relative geometry; a geometry only feasible with a computer. Both adopted a metaphorical cybernetic animal: the Turtle. To Papert the Turtle was a metaphor for control over the computer by children; to Abelson it was the inspiration for a formal geometry. Both had borrowed the "turtle" from William Grey-Walter, for whom it was a "Machina Speculatrix" Thus were set the seeds of confusion.

Part 1: Assimilation

1. Darting about England: a decade of decay

Beginning in 1982, the UK government controlled "Microelectronics Education Programme" introduced computers into the classroom. The majority of the personnel involved in this were mathematics specialists. The development of an indigenous implementation of Logo[1] by the AI Department of Edinburgh University, as a tool of general application to the mathematics classroom notwithstanding; these teachers, researchers, and advisers, focused on but one aspect of Logo: "Giving instructions to The Turtle".

Turtle graphics packages

To serve this limited need, government advisers sponsored a number of low-cost "turtle graphics" packages written in BASIC. The most famous of these, DART inspired by Edinburgh Logo. The DART "language", though borrowing Logo vocabulary, had very weak syntax. It failed to differentiate data and instruction, restricted statements to one per line, and had limited capability in passing parameters. Moreover, the author credited DART as being an implementation of "Logo graphics", rather than an implementation of Abelson's Turtle Procedure Notation (Abelson & diSessa, 1981, Appendices A & B). Later apologists claimed that this syntax was "easier for children than the " and : of other versions of Logo". Thus, the Logo expression:

```
make "forward 50
forward :forward
```

was "simplified" to the following syntax:

```
MAKE FORWARD 90
MAKE RIGHT 50
REPEAT 4
RIGHT FORWARD
FORWARD RIGHT
END
```

Figure 1 Functionally correct DART.

DART, not Logo, was the formative experience for teachers in the UK - the philosophy without the language through which it was expressed. Consequently, the notion of computer language degenerated to "precise sequences of instructions". Logo became little more than "Turtle Talk". Products called "Logo" but with DART's syntax, began to appear in the UK. DART itself is now sold as a product which will "fulfil all

National Curriculum Logo requirements."

Bit switching

The technical control domain followed a similar course. The physicists and engineers involved focused on the bitwise control of input/output ports and the mechanisms connected to them. CONTACT, a dedicated software package, again with the appearance of Edinburgh Logo but the syntax of DART, was produced. This package has reached version 3.5 and is marketed by the National Council for Educational Technology as a "control language" that is "Logo-like".

Logo becomes turtle graphics

At the 1993 BETT Exhibition (the major UK annual computers-in-education exhibition) a software package called *The Crystal Rain Forest* won the BBC Educational Computing gold award for primary school software. This package, voted top by schoolteachers, was described as "an adventure in Logo". In actuality the "Logo" was turtle geometry and the *Crystal Logo* supplied, a simple turtle graphics package implemented in BASIC, *which retained the syntax of BASIC.* (A legal challenge to this under the UK Trades Descriptions Act failed because, "there is a general agreement in the trade that use of the turtle graphics is equivalent to Logo".)

Turtle graphics becomes control

The school attended by my son, a high performing state grammar school with an excellent reputation and a five hundred year history, currently uses "DART to cover elements of control technology"; following examples given in the statutory orders for Technology in the national curriculum.

2. The process of Assimilation

What happened to Logo? In truth, it was never used in schools. Piaget, though he made an uncertain contribution to the theory of teaching, was a superb epistemologist. Logo, as he would have predicted, was assimilated to existing schema within education; schema not in the minds of children but in those of the grown-ups (adults) who controlled the system. The child-centred "discovery" approach of English primary education could assimilate an exciting concrete object like the Turtle; but it could not cope with the need first to learn, then to teach, computer language. Nor, indeed, could it cope with the notion of procedural notation - still absent from the maths curriculum (Mathematics (1991) Orders) - nor yet turtle geometry.

Old bottles

Two extant conceptual frameworks in particular came into play: the classic mathematics of shape and space, and mechanistic notions of technical control.

- The first UK use of the Turtle was to draw the classic figures of plane geometry.
- Technical control was concretely expressed as switching bits on a parallel computer port, embodying the mechanistic concept of a precise sequence of operation.

Neither gave any scope for expressing the syntactical strengths of Logo: its list and word orientation; its facilities for recursion. New wine had been poured into old bottles.

Commanding?

The assimilation of new ideas to older frameworks was, of course, to be expected; after all, the motor car was first perceived as a "horseless carriage". Papert, himself was not immune from this. Look carefully at the examples in *Mindstorms*[2]. He is always telling the Turtle what to do. He orders it about like a conscript on a drill square. Yet, Logo has many primitives[3] to provide users with information. Consider the Turtle. Alone on the screen, it has no memory of where it has been and no sense of direction. To gain both, the turtle needs:

- a map - home at [0 0], and the primitive **pos** to tell the user where it is;
- and a compass - 0 towards the top of the screen and the primitive **heading** to tell the user where it is pointing.

It almost appears that Papert was so concerned to avoid the computer programming the child that his examples of "Turtle Talk" became a monologue on turtle geometry.

The rise of Turtling

The rapid progress of assimilation in the UK is illustrated by first book on Logo commissioned by the MEP. Beverley Anderson's *Learning with LOGO* (1986), was a report on classroom experiences by an experienced primary specialist and computer "non-expert". Her report showed that programmable toys such as Milton Bradley's Big Trak were seen as "Turtles"; turtle graphics programs written in BASIC, such as DART , were not distinguished from Logo; and Logo itself was viewed as difficult, expensive, and (possibly) not necessary for doing Logo: "On the issue of full (sic) LOGO versus DART and ARROW both sides seem to have a point." The British LOGO User Group protested this. However, DART was a freely copyable program and hence infinitely less expensive than an implementation of Logo.

Vanishing geometry

Despite clear evidence that it was turtle geometry that schools were teaching, Anderson (ibid.) did not list *Turtle Geometry* in the references. Why not? There are two possible reasons: firstly, a clear, concise course on turtle geometry contradicted the "mathematical exploration" approach then fashionable in UK primary education; and secondly, the experts were quite convinced that *Turtle Geometry* would frighten English primary school teachers[4]. Later books by mathematicians followed this pattern. *Making Logo Work* by Ainley and Goldstein (editors of MicroMath) and *Children Learning with Logo*, the report of the PLUM project (funded by the Nuffield Foundation and published by the National Council for Educational Technology) both omitted any reference to this basic text. Analysis of the Logo published in these works evidences not a single use of a Logo reporter (operation). They, like Papert, were encouraging children simply to order the computer about using commands.

Logo in aspic

Five years after the start of the Microelectronics Education Programme, the UK government decided to introduce a National Curriculum for schools in England and Wales. The word "LOGO" appeared only in the mathematics curriculum (Doyle, 1990); the Technology (Information Technology) curriculum studiously avoided its use[5]. In both curricula Logo was equated to turtle graphics. In Mathematics the turtle was for teaching aspects of algebra, number, and shape and space. In Technology "controlling and arrowhead on the screen" helped to teach control. Additionally, there was evidence that neither Logo nor turtle geometry were understood by the curriculum constructors. The following excerpt is taken from the Mathematics Orders (1991):

Attainment Target 3: Algebra, Level 4.
Statement of Attainment: *work with coordinates in the first quadrant.*
Example: *create shapes using turtle geometry in an appropriate computer language.*
Thus developed crook-back Logo in the UK.

Right triangle

And nowhere is there any recognition of the Aristotelian impossibilities inherent in using turtle graphics: That a child can cut out, with scissors the semicircle or right triangle that s/he cannot draw using turtle geometry is nowhere considered.

3. Language beyond the Turtle

Elsewhere in the world, in Quebec to be precise, Brian Silverman had developed Logo into a product more suitable for the classroom: LogoWriter. The vocabulary had been both simplified and extended. Out went property lists; in came the vocabulary of word processing. Out went the separate editor and filing commands; in came word-processing facilities throughout. Out went files; in came the page metaphor: a sheet of paper in a scrapbook. Graphics and text could be freely mixed on the same page. The Turtle acquired all the graphics features needed to write a simple drawing program. Teachers who were frightened by disks, filing systems and editors found this new working environment far more comfortable. The metaphors for computer operations were those of the classroom, not the office. The convenience of a word-processing replaced the computerist's text editor.

LEGO/Logo

The concepts within LogoWriter were extended to technical control: LCSI teamed up with LEGO. The

notion of setting bits high or low was replaced by one of "talking to" a socket on the interface box. The language was message, not electronics, oriented: 'talkto 1 on' instead of 'turnon 1'; 'listento 1' followed by a primitive signifying the information required instead of 'sense? (bit)1'.

Overlay input
This message-passing approach has also been extended by adding overlay keyboard primitives. Here a key on touch sensitive matrix keyboard is assigned a message which it passes to LogoWriter when pressed. The meaning of the message for the child is expressed in graphics on a paper overlay. This is a very powerful facility. It is one contribution to educational IT of which the English can be proud.

Microcomputer worlds
The most recent LCSI/LEGO product has taken this process further. "MicroWorlds" has a DPT/graphics working environment. The screen turtle has been joined by other objects: text boxes, buttons, and sliders. The multicoloured turtle can grow and shrink. And all the on-screen elements can be placed and altered using tool palettes and a pointer. The Logo vocabulary has been extended to enable all these new concrete activities to be carried out using Logo as an alternative to point -and-click.

Within this same framework, LEGO Dacta, in conjunction with LCSI, have taken control a step further. The "port" has gone, to be replaced by serial communication with an interface box. The software has been "told" what may be attached to the interface box. Hence, 'talkto 1' has been superseded by 'talkto "motora'. Indeed, with Control Lab this process has gone further. The command 'motora,' (i.e. with a comma suffix) may be used, whilst 'listento' has vanished to be replaced by a reporter derived from the sensor's name at its point of attachment, e.g. 'light4'.

Objects
The above developments owe much to the development of Object-oriented Logo. But they differ significantly in one respect: MicroWorlds and Control Lab have on-screen objects that may be selected and created using a point-and-click graphic interface. This is additional to the creation of objects by command. Thus, they provide a pre-linguistic, concrete level; whereas Object Logo works from the linguistic to the concrete. This is an important difference; it is similar to that between creative and descriptive writing.

Tension
But, there is a tension between these developments and the more "classic" Logo implementations. Though the same language is implemented in both Brian Harvey's Berkeley Logo and in the LCSI/LEGO Dacta Control Lab, this is not immediately apparent. The relationship between the "LCSI" developments and Harvey's computer science oriented implementation, in many ways, mirrors the position in the UK where Logo is assimilated to "Turtling". Harvey's implementation keeps Logo tight within the parameters of pure computer science; and the evolution of the language is restricted to this domain. These tensions have arisen because we have reached the point at which we need to re-examine our pedagogical rationale for using Logo. Education cannot resolve this tension without radically reconceptualising the computer in school.

Part 2: Accommodation

1. A brief history of language
Question, What is a computer?
Answer, It is writing.

Drawing and Writing (En4/1a)[6]
Amongst the paintings of reindeer and bison at Lascaux there appear more formal designs, for example a row of dots and a square. These 17,000 year old non-representational marks signify one option for the development of written language.

The earliest written languages were pictographic; the formalisation of drawings of objects. Examples are, Egyptian hieroglyphs of 4,500 years ago and Chinese logographs from 3,800 years ago. Alphabetic writing developed somewhat later. The Minoan Linear A, at 3,500 years, is the oldest known. With it began the use of arbitrary symbols to express the phonetics instead of the referents of language .

Alphabetic writing took a number of directions. Our "Roman" alphabet originated in Etruscan culture 2,700 years ago. It retains today, the chiselled, monumental, characters of antiquity. The other major modern alphabet, Arabic, like the characters of Chinese, was optimised for handwriting on paper; the form of a character depending upon its position in a word. The Roman (and Cyrillic) alphabet did not undergo this optimisation, retaining individual characters of fixed form, adding only the lower case.

Tallies and numerals

The technology of writing encompasses not only to the words of a language, but also its numeral system. All early systems were derived from a simple tally. The Ionian Greeks did use arbitrary characters: letters of the alphabet. The Romans, however, reverted to mixed letter/tally system. Our modern system of numerals, including zero and the decimal point, originated in Hindu mathematics and reached Europe as late as the 12th century via the Arabic culture of Moorish Spain. The merchants of Pisa were amongst the first Europeans to make use of it.

Alphanumeric logic

The alphanumeric system of the Roman alphabet and Hindu-Arabic numerals proved to be extremely powerful. Released from concepts of counting, number theory on the one hand and practical accountancy on the other developed rapidly. The discrete nature of each letter and numeral facilitated the development of the movable-type printing press; thus facilitating universal literacy and making modern education practicable. But the notation made possible more radical developments than this.

In the mid-nineteenth century language and mathematics met (in Ireland) in George Boole's algebra: the binary truth tables of "and", "or", and "not".

Awaiting technology

Thus were put in place all the necessary concepts for the development of the digital computer. Practical realisation, however, had to await a suitable "writing material". Mechanisms, as Babbage demonstrated, were insufficiently powerful. Theory could, however, progress; for example, the work of Kurt Goedel, on arithmetising language and Turing's work on computability. A war, coupled with the development of radio, led to the first binary computer being built, in Manchester. The full practical realisation of the computer, however, was only achieved when, following Shockley's development of the transistor in 1948, semiconductors replaced the thermionic valve.

Digital writing paper

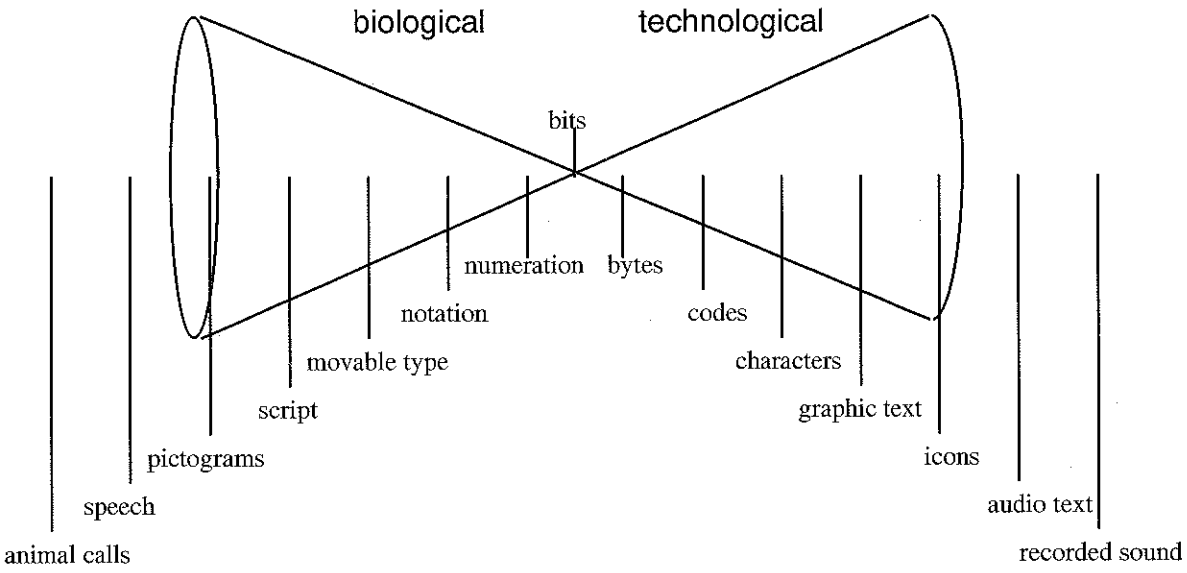
The digital computer, as we now know it, adds, does Boolean algebra and remembers. It is a "something" that reads, writes, and records, according to instructions. What it writes depends upon what it has been told to do with what it has read. This makes the computer a very odd sort of machine. When we realise that "computing" can be done using mechanisms, thermionic electronics, semiconductor electronics, light, and so on, any notion that we remain within the framework of physical mechanics appears tenuous. Computing, like writing, can take place on and within a variety of media.

The computer, product of high technology that it is, is probably best thought of as a development in writing materials. The essential difference between the computer and, for example, a pencil and paper or Etruscan writing tablet, is that the computer is not an artefact in the sense that the writing tablet is. The computer is an expression of language itself. It is concrete language: writing with the fluidity of spoken language

2. Forms of formalisation

Let me introduce a diagram I have found helpful in my thinking:

The language cone



If we accept that this is a useful representation, certain interesting questions arise.

Language and computing
Natural languages have particular strengths and weaknesses. Some achieve the status of lingua franca: Latin for medieval Christendom; French for seventeenth century diplomacy; English, currently, for science and technology.

In the context of computers, English has some interesting characteristics. Firstly, English, like Irish, does not encourage the division of the world into things feminine and things masculine; this usage is reserved for the biological world. This is not surprising, the other two major influences on English, the Romance and Germanic languages, do not always agree on the gender of objects. Secondly, again for etymological reasons, English spelling is not fully phonetic. As a consequence, written English is more independent of spoken English than is the case for most languages. English retains the alphabetic simplicity of Latin. It makes no use of diacritical marks; has no accented characters. (Was it accidental that the typewriter was invented in the USA four years after George Boole's death?)

Character coding
These characteristics of English are helpful when thinking about computers. Letters and words may more easily be thought of as inanimate objects manipulated as binary code. The enumeration of letters is easier for English. The ASCII code needed only to assign two numbers for any given letter: one for upper case, one for lower - seven bits sufficed. Consider the coding system for French: where à, á, and â are also required; to include other European languages, add ã, ä, å, etc.

3. Schooling beyond the binary horizon
Let us return to the "language cone". We see that the highest level of technological development of natural language comes with its encoding as binary arithmetic. This, realised through semiconductors has brought writing "alive". Through a form of recursive process, written instructions can act upon written language, giving it a time dimension. We are now beyond the binary horizon. Clearly we need to explore the potential of this development in technology as a pedagogical tool. But how should we approach language beyond the binary in schools? How shall we best adapt our pedagogy and extend the curriculum to the computational?

Computer language
Our discussion has focused upon language. Let us remain in that domain. At the "native" level computers are given instructions using binary codes. This is not a system found amenable by people.

It was obvious that as natural language may be expressed in binary form, so must it be possible to develop a language which people and computers might share. Linguistic science had already developed the necessary concepts. All that was necessary were decisions on appropriate lexicon and syntax. Many languages were developed: FORTRAN (formula-translation) for mathematical work; COBOL for business; ALGOL for science; LISP for natural language research; BASIC was developed from FORTRAN as a general introductory language; ALGOL gave rise to Pascal; C developed as an applications programmers language; Prolog returned to computing's roots in logic; object-oriented programming techniques appeared. One language, Logo, was specifically developed for use by children.

Computer language and children
As the UK experience shows, there has been considerable ambivalence towards the use of computer language by children. Early experience with BASIC[7] proved most unfortunate. This was because, as a language designed to be an easy introduction to computer language for college students, it presupposes a range of well developed literacy and numeracy skills; skills that young children do not have.

Logo's aetiology is very different. It emerged from work in "Artificial Intelligence". The agenda for AI, we should recall, was set by Alan Turing when he asserted that we must deem the computer intelligent if, where a person was having a conversation with a computer, it was not possible to determine which was the computer. At the heart of AI is natural language processing[8]. LISP, the progenitor of Logo is an AI language.

Writing as technology
A long and tortuous path led to the development of the binary digital computer. That path was one of development in the technology of written language.

Language, even if we accept only the weak cultural hypothesis, influences thought. Computer language carries within it those aspects of our culture that facilitated its development. Logo, developed in a language conscious research setting, best encapsulates the essential aspects of that culture and counterpoises it with natural language.

The native mode of the computer is intrinsically, therefore, the written form of language. Computer programs are not, as Edinburgh Logo would assert, function machines built using code. They are literature written in computer language.

If children are to use the computer at this, intrinsic, level, then they can only do so through written language. That the computer can only do what can be described in writing needs be understood almost subconsciously. That children need to work at this level should - but in UK education is not - be accepted as axiomatic.

Part 3: Application

Structures, mechanisms and messages.
Children learn about the built environment by playing with bricks and sticks. They learn the fundamentals of machines playing with wheels and axles. What do they need to learn the digital fundamentals? They need words and syntax; but technological words and technological syntax, not natural language. The relationship between computer language and natural language may be likened to that between a branch and a beam. The branch is constrained by the limits of biological processes; the beam is a branch transformed by forces beyond the biological. Written natural language is constrained by the physiology of speech; computer language can extend beyond this. Take a simple example: compare a computer graphic with those in the Lascaux caves. The computer graphic can always be reduced to a linguistic description without degrading it; those of Lascaux cannot.

1. The storm abates

Before turning to the school curriculum itself, we need to rebuild the devastation caused by Papert's Mindstorms. The word "storm" in English comes from the same root as the German "Sturm". The French have another word, "orage" (Italian "ura'gano" - English "hurricane"). In colloquial English both words may be used together: "a raging storm", but orage is very close to two other English words, "rage" and "outrage". *Mindstorms* was an outrageous book. In it Papert gratuitously attacked schools and made outlandish claims for computer language. (The word "mindstorm" also calls forth associations with the mental seizures of epilepsy.)

Mathland was an outrageous notion if only because the linguistic level at which children must interact with the computer interposes itself between the pupil and the binary just as much as natural language does. Computer language does no more than open up a quicker route to the mathematical for those who have the capability to, and interest in, following that road. To presuppose that a child can "discover" several millennia of subtle intellectual development by typing on a keyboard is nonsense. A child, to develop, needs a competent language user as a teacher. The school, for all its faults, is the most efficient mechanism for teaching yet devised. A good teacher, like a good mother, will aid development by pacing learning to the pupil. But, for teaching to take place, the outcome, or at least the objective, should be known. The problem with Logo has been that neither teacher, pupil, nor language developer, had any idea what the outcome might be. Let me illustrate the problem by reference to the Turtle.

The Turtle as drosophila

Recall the cultural environment within which Logo developed: a high powered University research lab. How, typically, do researchers focus their work? They simplify. They search for a framework within which to isolate the critical features of their object of study. In the early days of genetic research the fruit fly, drosophila, was a favourite research tool. Drosophila breed rapidly and mutate easily[9].

Consider the status of the Turtle. Papert refers to it as a "computer controlled cybernetic animal". In *Mindstorms* he asserts, "its principle role here is to serve as a model for other objects, yet to be invented." (p11). The Turtle was then, and is now, the Logo researcher's drosophila. It has been used as such by cognitive and educational researchers world-wide. That the Turtle has proved such a powerful research tool does not imply that it may easily, or sensibly, be introduced into the classroom.

Calm words

If we accept the hypothesis that the computer is language, then the most obvious way to introduce Logo is through language. Papert, in *Mindstorms*, correctly identified the word-processor as the most important use of the computer as a tool. Brian Silverman, a Quebecois more aware of the tensions of language than most of us, extended Logo's vocabulary to this domain. In doing so, he gave us the short Logo sentence which epitomises the essence of the computational: ***run selected***.

"Run selected" encapsulates the difference between language in a computer and on paper. The highlighting of a word on paper merely draws attention to that word. Highlighting a word on a computer screen implies that the computer can know the sequence of characters that has been highlighted. If that sequence of letters, that word, corresponds to an instruction the computer has been give, it can carry it out. Put another way, if the word highlighted on the screen has meaning for it, it can act on it.

Hypertext

This property of language on the computer makes possible hypertext systems. In the classroom it opens up the possibility of turning plain, sequential stories into branching "adventures", where decisions are tied to active "keywords" in the text. It enables one to play with and explore the elements which go up to make text: consonants and vowels; words and spaces; numerals and operators; punctuation; and so forth.

2. Machines and Messages

In our discussion so far we have focused solely on language. Let us now widen out a little. Consider our machines. By replacing manual energy with that derived from fossil fuels both the power and precision of

machines has been vastly increased. Control of these machines has, however, depended on specially designed feedback systems: the steam engine governor or domestic thermostat, for example. Computer language has given us a more general way of expressing the system of control. Machines are now fitted with sensors that transmit digital messages to computers which then process them and, in turn, send digital messages to an actuator of some sort. Because this makes possible the economic use of otherwise uneconomic resources, it has significant consequences for our industrial and ecological systems.

Farming wind

Take a specific example: wind power. Windmills are machines of great antiquity, they used the same technology as sailing boats. Modern wind turbines, on the other hand, owe more to flight. We see the start of the necessary developments in Leonardo da Vinci's "helicopter", which could not work in practice because: a) people cannot deliver sufficient power, and b) rotating aerofoils need to change pitch for takeoff, forward flight, and landing. The internal combustion engine (or powered turbine) solved the first problem; the skill of the pilot the second. To use the same technology to generate power from the wind economically, we must do away with the pilot.

Modern wind turbines can economically be used to generate electricity because digital technology has enabled us to replace the pilot. Sensors connected to a wind vane provide the information on wind direction; an anemometer provides information on wind speed. This information, in digital form, not only provides the feedback to keep the turbine facing into the wind and the blades set to the correct pitch for its speed, but can also be translated into a form easily understood by a supervising engineer at the end of a telephone line many miles away.

3. Fair winds, fair words

As the Mindstorms abate, we may reflect on the linguistic history of the computer and perceive the post binary language of the computer as one to share with our technology.

A glimpse of the Logo of tomorrow may be seen in LEGO Dacta's "Control Lab". Here, the vocabulary has been extended to include computational "objects not yet invented": a text box; graph; picture; monitor. Dacta Logo now knows about temperature, both Fahrenheit and Celsius!

Modelling our wind turbine may be done at many levels. With a single sensor, wind speed alone may be measured. The power output of the turbine relative to wind speed could then be modelled simply by inserting the formula: $P = .5rAV^3$ (in Logo form) into a monitor box. An ambitious project might model all the electrical, mechanical, and communication features of a small wind farm.

But Control Lab is more than just the Logo language. The software provides all the conventional tools used for running an experiment and writing a report. Where LogoWriter was a word-processor you could communicate with, Control Lab is a desk-top publishing. But it is an active DTP package: graphs record as well as represent, monitors monitor.

4. The digital curriculum

Does the current curriculum contain the necessary elements for growth into the digital age? We ask not the question "Is Logo in the curriculum?" but the precursor question, "Is the curriculum suited to an age where people and machines can share a language?" The English National Curriculum is a very recent phenomenon. Is it possible to see how well it has incorporated the growth points of the digital age?

There are a number of ways in which this might be done. We shall use two.

Any researcher, journalist, academic, or pupil, will search for information using keywords. Six keywords from this paper are: language; writing; codes; computer language; digital; binary; logic. Our second approach will be to examine the programmes of study for the core subjects: English, Mathematics, Science, and Technology (IT) for key concepts.

Keywords

We find the occasional occurrence of our keywords. As we might expect "language" occurs quite frequently,

but only in English in such contexts as "the appropriate use of language". In Science we find: digital (watch); (electronic systems, both analogue and) digital; logic (gate). In Technology we find other words: programmable; precise sequence of instructions. The words "computer language" appear the same number of times as "glasnost": once, in Ma3/4c.[10] Which leads us neatly to our second search.

Concepts

For the English National Curriculum to have the growing points for the digital age, certain new areas should be present, at least in embryo. Here is my list for the core.

- English. We would expect to find some reference to linguistics, perhaps psycho-linguistics. To formal grammars; concepts of parsing; the technology of writing. We do not. What references there are have a literary focus: grammar and spelling are highlighted in a language use, not a language study context. (I wonder how the notion of "dyslexia" would fare were competence with written language to be equated with competence with other technologies, mechanisms. say?)
- Mathematics. There would be a separate attainment target for computational mathematics. This would include: binary numbering; systems of coding (e.g. ASCII); procedural notation; turtle graphics and turtle geometry; notation systems; Boolean algebra; discrete mathematics; logic.
- Science. Linguistics; speech production; synthetic speech; cybernetics; aspects of psychology including notions of intelligence and learning.
- Technology (IT). Obviously, children should experience and develop skills in the uses of the current, or recently obsolete, technology. The requirements here are no different than those for any other artefact. But there needs to be another strand. Culture, it is said, is embedded in language. Thought is either influenced by language or formed by it, depending upon your viewpoint. Computers are language. Hence, a major component of IT must be computer language; for the language of the computer is the language of the digital age.

None of these elements is present. Particularly disappointing is the lack of recognition for English's contribution to the digital. The language of "Shakespeare, Milton and the (King James) bible" takes on a new light when viewed as the mother of the computer.

To Logo, or not to Logo

We now need to take one critical step: to accept that, in a modern education system, children need a language they can share with technology. Having taken this step, the remainder follows. No computer language other than Logo is suitable. If we opt to use Logo, then we need to learn it. To learn it implies that we teach it.

Mícheál Ó Dúiohil, Skipton April 1993
Honorary Chair Logos

Notes

[1] Edinburgh Logo differed from both MIT and LCSI Logo in that it incorporated the concept of "building a function" as opposed to "telling the computer how to perform a procedure". This difference in underlying metaphor has profoundly influenced the understanding of Logo in the UK.

[2] Written using Abelson's Turtle Procedure Notation, not Logo.

[3] Reporters, as LCSI now calls them; operators in other and earlier implementations.

[4] Until the 1970's there as no requirement for primary school teachers to have even "O" level mathematics.

[5] The original proposals did use "LOGO" in the same context as the Mathematics curriculum; however, when it was discovered that Logotron had a monopoly on Logo fro the BBC Microcomputer all occurences of the word were replaced by "turtle graphics", or a similar phrase.

[6] English in the National Curriculum statement of attainment "begin to show an understanding of the

differences between writing and drawing ... "

[7] There were objections to the form of the assignment statement, which contradicts mathematical usage of the equallity sign, and its tollerance of missing spaces between words (cf. early Etruscan).

[8] But, we must be careful of our understanding of language in the computer's world. We always mean written language: those black marks on white paper which inspired G Patrick Meredith's *Instruments of Communication.*

[9] Richard Dawkins "biomprphs" let us simulate this on a computer.

[10] The original correct, if loosely phrased, sentence was provided by the writer, it was edited, without consultation, by an apparatchik into a fascinatingly false statement.

BIBLIOGRAPHY

Papert S (1980), "Mindstorms", Brighton: Harvester Press
Abelson H & diSessa A (1981), "Turtle Geometry", Cambridge, Mass: MIT Press
Harvey B (1985), "Computer Science Logo Style", Cambridge, Mass: MIT Press
Grey-Walter W (1964) "The Living Brain", Harmondsworth: Penguin
Anderson B (1986), "Learning with LOGO", Loughborough: Techmedia
Ainley J & Goldstein R (1988), "Making Logo Work", London: Basil Blackwell
Blythe K (1990), "Children Learning with Logo", London: NCET
NCC (1988), "Mathematics in the National Curriculum", London: HMSO
NCC (1989), "Technology in the National Curriculum", London: HMSO
NCC (1991), "Mathematics in the National Curriculum (1991)", London: HMSO
NCC (1988), "English in the National Curriculum", London: HMSO
NCC (1991), "Science in the National Curriculum (1991)", London: HMSO
NCC (1993), Proposals for English in the National Curriculum, London: HMSO
Schuyten G & Valke M (1990), Teaching & Learning in Logo-based environments", Amsterdam: IOS
EUROLOGOS, Vol 1, 1992

Preschoolers' Logo Play: Characteristics Associated with Logo Competence

Alison Elliott

Faculty of Education
University of Western Sydney Nepean
PO Box 10, Kingswood, 2747
Australia
e-mail: a.elliott@uws.ed.au

Abstract

The purpose of this paper is to describe social and cognitive factors inherent in the Logo play of four year old children who scored highly on measures of Logo competence. Particular focus here is on metacognitive aspects of the play transactions and ways in which they framed children's active organisations and negotiations of their play. Play characteristics are discussed in the light of Vygotskian notions on the role of cognitive scaffolding in facilitating problem-solving.

Keywords: Logo, social play, interactivity, metacognition.

Alison Elliott *is Associate Professor of Early Childhood Education at the University of Western Sydney Nepean. A major emphasis in her research is on cognitive and affective dimensions of young children's learning in technologically rich classroom contexts.*

Preschoolers' Logo Play: Characteristics Associated with Logo Competence

Introduction

There is continuing evidence to suggest that Logo environments have the potential to engage young children in interactive experiences believed to facilitate the growth of cognitive, and particularly mathematical competence. However, there is little detailed information on the nature of this interactivity or ways in which interactivity helps shape mathematical understandings. This paper draws on findings from an investigation of 60 preschoolers' Logo play to highlight characteristics of interactive play transactions that seem central to the development of Logo competence. Analyses of Logo play transactions indicated that children engaging in high levels of social play scored significantly better on measures of Logo competence than did children who played independently (Elliott, 1992). Particular focus here is on metacognitive aspects of these play transactions and ways in which they framed children's active organisations and negotiations of their play and problem-solving. Play characteristics are discussed in the light of Vygotskian notions on the role of cognitive and metacognitive scaffolding in facilitating problem-solving.

Play, learning and Logo

Play transactions are considered central to young children's construction of meaning, thought and action. Hence their consideration in any investigation of preschoolers' development of mathematical awareness. What ever their play, the interactivity generated by children's involvement with playmates and adults and their use of "tools" and objects as they explore their environment, afford the mediational means by which cognitive development and functioning are supported.

For the preschool child play affords opportunities for interactions through which events are interpreted and given meaning. If, as Vygotsky (1976) proposed, play enables a "child's greatest achievements", then play experiences in new and powerful computer-based contexts may prove especially valuable. Given the strong social and cognitive scaffolds inherent in many computer-supported settings interpersonal communication about objects in the external world and the subsequent transition from external to internal planes of operation should be facilitated (Luria, 1973; Vygotsky, 1962; 1978).

With the recent addition of computers to preschool classrooms, many children now "play" in computer-generated contexts, just as they "play" in more traditional environments (Blemings, 1988; Elliott, 1984; Klinzing, 1985). Yet, despite their growing use in early childhood settings there are few studies providing coherent pictures of children's play or problem-solving behaviour in computer-supported contexts, let alone in Logo environments. Indeed, there appears to be very little Logo activity occuring in preschool settings, let alone a strong emphasis on Logo research.

Interactive play contexts

Recognition of the value of an interactive milieu for cognitive development derives largely from the sociocultural theory of Vygotsky and subsequent followers such as Rogoff (1990) and Wertsch (1991). For Vygotsky (1966, 1978) the interpersonal nature of cognitive activity is central to explanations of intellectual development and provides a framework for the analysis of connections between social interactions and cognitions. He contended that children must be active participants in social interchanges for communicative skills to develop and that such skills are central to achieving consensus about multiple realities. It's through their activity, both their real-world experiences and play experiences, that children are afforded opportunities for this development. The problem-solving nature of the activity, whether it is making a cake (a real world activity) or playing house (a pretend play activity) becomes a point of convergence and synthesis for these multiple realities and results in the formation of changed perspectives for participants. Communication with others as part of the problem-solving process contributes to understanding and the development of

knowledge. As Vygotsky (1978) and Luria (1982) emphasised, it is the cognitive scaffolding afforded by this communication that activates thought processes and gives meaning to actions.

While children's play is usually mediated by peers who share some of the cognitive workload through provision of cognitive and metacognitive guidance within the activity (Day, French and Hall, 1985) computer environments may also provide such scaffolds, for example through memory cues, multiple representations of information and so on (Pea, 1987; Salomon, Perkins & Globerson, 1991). Such supports, especially when complementing those afforded by peers and adults, may foster new opportunities for children to engage in high order operations, such as testing novel and complex hypotheses, discovering relations among concepts, or engaging in construction of new knowledge (Papert, 1990; Salomon, 1990; Salomon, Globerson and Guterman, 1989).

Situating Logo in a preschool play context, means that children usually set problems and discover solutions on their own. In making a design or directing floor turtle from A to B they must define a problem goal and draw on existing knowledge of shapes, direction, angles, lines in space and so on to selectively establish a goodness-of-fit between existing knowledge and the requirements of the new problem (Elliott & Hall, 1985). As in other play experiences, children bring their own knowledge and experiences to use as a basis for their exploratory Logo activity (Hoyles & Noss, 1987). From a Vygotskian perspective, it is the "intellectual partnerships" established between children and mediational supports in the environment that enables the coordination and synthesis of actions and thoughts that lead to a progressive restructuring of cognition (Forman & Cazden, 1985; Palinscar, Brown & Martin 1987).

For preschoolers, Logo provides a context in which they can play the mathematician through the construction and solving of problems related to space, direction, shape and number in a constructionist context (Papert, 1980; 1990). Logo provides a setting for playing with variability, patterns, naming, associations, properties and communication in their play; children have to make sense of an idea, and represent and test hypotheses about the rules governing problem domains (Noss, 1985). Such rich settings, as Brown (1984) and de Corte (1990) contend, are necessary if children's discovery processes are to have important cognitive outcomes. As children direct a turtle up a road, left at an intersection, and through a tunnel to reach its house they discover that some solutions work and some fail. Importantly, they also discover "why some solutions fail" (Hofman, 1986, p. 99) which may lead to piecing together a new solution from the lessons of the most recent failure and fragments of earlier models (Try, 1989).

Critical to young children's play is their language. According to Luria (1973) and Vygotsky (1978) language both reflects children's construction of meaning and assists in this process. First, "speech accompanies the child's actions and reflects the vicissitudes of problem-solving... At a latter stage...(it) functions as an aid to a plan that has been conceived but not yet realised in action" (Vygotsky, 1978, p. 27). In this sense speech assumes a metacognitive function in problem-solving. As young children talk through a problem in order to solve it, they reflect on their thoughts as well as their actions as they orchestrate their play. In Logo environments children's talk is likely to reflect their predicting, anticipating, comparing, explaining and collaborating behaviours (Anselmo & Zinck, 1987; Clements, 1985, 1986, 1987; Clements & Nastasi, 1988; Elliott & Hall, 1985; Genishi, McCollum & Strand, 1985; Hughes & Macleod, 1986; Newton & Henderson, 1987; Papert, 1980; Papert, 1990; Try, 1989). As such language reflects the self-regulatory or metacognitive dimension of thinking and learning inherent in the overlapping experiences that involve monitoring problem-solving. Language indicative of metacognitive processes includes verbs like think, mean, decide, understand, remember, forget, know, and guess (Brown, 1987; Day, French and Hall, 1985). The resolution of cognitive conflict, specifically the need to make sense of, or give meaning to the actions of others, to restructure cognitive performances, to coordinate with others, and to continually modify personal goals is central to this processing (Bearison, 1982; Perret-Clermont, 1980; Clements, 1990).

For Vygotsky (1967) a major importance of play is its role in facilitating transitions to abstract thought. Fully developed thought is abstract rather than anchored in the physical givens of the environment; it does not depend upon concrete objects and actions in the perceptual field such as "props". In contrast preschoolers' play *does* depend on props. "Props", particularly objects, serve as the pivot for meaning. An object can stand for something that it is not, such as a stick for a horse; at the same time it provides support for the child's embryonic ability to imagine. In a Logo context, the situational features of the microworld, or its "props"

seem to straddle the boundary between abstract and concrete. With this support it may be that Logo play can act as an important transitional medium between the situationally determined thought of infants and the more liberated, flexible thought of the older child.

That children's social interactions within play settings are largely responsible for engendering a continuing organisation and reorganisation of information and experience finds robust support in the play literature (Donaldson, 1978; Pellegrini, 1986; Quilitch & Risley, 1973; Schwartzman, 1978; Shure, 1963; Sutton-Smith, 1979; Tyler, 1975), the wider psychological arena (Rogoff, 1990). For young children, increasing facility with language allows external processes to be displayed, shared and practised in their play. Language that is both explicit and elaborated is vital to planning, communicating and solving mathematical problems in many traditional preschool play activities such the block, house and environmental play areas (Pellegrini, 1986; Quilitch & Rislev, 1973; Shure, 1963; Vandenberg, 1981). Evidence from several studies of young children's activity in Logo contexts suggests that this joint coordination of social and cognitive activity is central to successful problem-solving (Clements & Gullo, 1984; Elliott & Hall, 1985; Miller & Emihovich, 1986; Newton & Henderson, 1987; Noss, 1983, 1987; Try, 1989). In contrast, some preschool activities such as painting and drawing, completing puzzles and reading have a greater focus on individual accomplishment of tasks.

The Logo setting

Children self-selected to play in the Logo environment during "free play" sessions over a four to five week period. Free play sessions generally lasted for one and a half hours. Prior to the introduction of Logo activities teachers involved children in a series of action songs, floor and board games, puzzles and directed art and craft activities, designed to familiarise them with Logo actions and language. Children had also participated in computer-based play activities involving early mathematical ideas such as one-to-one correspondence, counting and matching to develop familiarity.

The Logo play centre required children to manoeuvre a "Tasman Turtle" within a setting enriched by addition of a number of concrete props (blocks, construction materials and toys) so that it looked much like more traditional play areas. The turtle was commanded using single keystroke commands in the immediate mode. Combinations of a set of commands F (Forward), B (Backward), R (Right turn) and L (Left turn) enabled the screen or floor turtle to be moved a prespecified distance and to turn through a set angle. Operative keys were highlighted on the keyboard. The directive "TOOT #" enabled the turtle to "talk".

Children were introduced to the Logo play area in much the same way as to other new play activities. That is, while a teacher showed the first two or three children basic commands for directing the turtle, subsequent participants in the play area were usually absorbed into existing groups where they were supported by peers. As they became involved in the activity the original players tended to move to other play areas.

Detailed insights into children's play were provided through observations of play and recordings of self-talk and conversations with peers.

Characteristics of Logo play

As mentioned earlier the playstyle most associated with subsequent Logo competence was social in nature and very clearly embedded in a mathematical context. Children frequently referred to "playing with the computer" and "playing with the robot" and clearly equated the computer corner with other play areas. In fact much of the robot play was very similar to that observed in construction play areas.

As children first came to terms with the capacities of the turtle their manoeuvrings were exploratory and very simple; for example, moving the turtle forward and back- analogous perhaps to placing blocks end-to-end or one on top of the other. With more experience, play became more complex and sometimes embedded in pretense. Increasingly, the presence of props such as blocks and cardboard boxes were taken into account

as children played. As they became more ambitious in the their play and the turtle was guided through longer and more complex journeys. Increasingly, successful use of counting (turtle steps), direction, position and estimation of length and distance became important. For example, children had to move the turtle left and right, round the box, under the bridge and estimate the number of turtle steps required to cover a large distance. Consistent with findings from other studies (Papert, 1980; Adams, 1985; Elliott & Hall, 1985; Try, 1989) "body geometry" became a pronounced and important aid in problem-solving as play became more complex. There were also occasions, as noted in other studies (Elliott & Hall, 1985; Wright & Samaras, 1986), in which play became embedded in a fantasy context. Such situations usually entailed giving the turtle "human" or "animal" qualities and involving it in day to day activities such as "shopping", "going home", "going to bed" and "eating dinner". According to Vygotsky (1978) these imaginative play transactions scaffold articulation between the anchored thought of young children and the more flexible and abstract thought of the older child. Most often, though, play was constructive in nature and involved getting from point A to B to C.

Interactivity and language

Language was central to most social play transactions and was, in most cases, explicit, elaborated and evaluative in nature. According to Vygotsky (1978) language associated with clarification and communication of problems and monitoring and evaluation of problem solutions is characteristic of young children's need to articulate their problem-solving activity especially as they encounter situations which involve a degree of uncertainty or conflict.

Clearly apparent in most Logo social play was children's use of language to guide and monitor their activity and seek structure and guidance from those around them. This is not to suggest, however, that all social play involved lots of talk. As suggested by Rogoff (1990) there were many occasions in which non verbal communication, such as gestures, seemed equally important in orchestrating play activity. Indeed, while verbal dialogue was strong in most social play groups, some competent Logo users, made very little contribution to the group discourse.

An important characteristic of language associated with subsequent Logo competence was talk reflective of children's efforts to actively monitor and regulate their play. For example, when Skye told Jane "don't go round, left, left" she was explicitly directing her activity. But her support became more metacognitive in nature when she said, *"shall we try that. Again, again, let's see if it'll go more. What did it do then? Are you going around?"* Here she was asking Jane to reflect on her thoughts and actions and how they would influence the outcome of their joint endeavour. Frequently, social play transactions afforded both metacognitive cues as well explicit guidance and direction.

The importance of negotiation in problem-solving was apparent from analyses of competent children's play transactions. As reported in previous Logo studies with young children negotiation and conflict seemed integral to problem-solving (Clements & Gullo, 1984; Clements & Nastasi, 1985, 1988; Try, 1989). Typically, this was characteristic of the planning and trialling of command sequences, identification and correction of errors, and the monitoring and reflection of the problem-solution, as illustrated in below.

Gemma:	Oh, press space. Watch out, it will knock down the whole cubby. TOOT, press space and a number and R. Don't you teach anything? (He meant "learn") I want that number...Now....Good, good.
Andrew:	Now go T...OOT.
Gemma:	There are too many spaces. Look, too many.... Space, space, space (pointing to each one). Let's fix it.
Andrew:	This many (counts to 5). Is that right?
Gemma:	No, it's four, four. Come on four!
Alice:	It's a funny noise. It's really long.
Edwina:	Don't you think that its going to crash.
Alice:	Move that block.... It's going all around the room.
Edwina:	Did you press it twice?

Alice:	No I didn't..... Here that, hear all the noise.
Edwina:	You did it twice.
Alice:	No I didn't.
Edwina:	What happened?.... (Edwina takes control of the keyboard.) I've just started. Haven't I?....(Silence while E commands the robot to move forward). No it's going to crash...Look, we're doing it, aren't we? It's going up to my chair. You made it go up to my chair, didn't you?
Alice:	Watch out....Let's go round the leg.
Edwina:	No, back.... and more.....

Expert guidance

Situations in which a more "expert" player seemed to initiate and monitor ideas and actions facilitative of a successful task completion were similar to those described in more traditional learning activities (Cooper, 1980; Day, French & Hall, 1985) and in Logo environments with older children (Clements, 1990; Clements & Gullo, 1984; Clements & Nastasi, 1988; de Corte, 1990; Try, 1989).

A finding not previously highlighted in other Logo studies was that opportunities for children to assume an "expert" role varied from situation to situation depending on peer groupings and complexity of the tasks. Interestingly, most Logo competent children seemed very flexible about the roles they assumed in play and most were observed in "novice" as well as "expert" roles depending on the play grouping and play context. Equally, competent children seemed well able to afford complementary and similar levels of support to each other. The value of these alternating roles seems to lie the opportunities they afford to experience problem-solving from different perspectives. Not unexpectedly, there was little evidence of less competent children's flexibility in role taking. Generally, it seemed, children scoring poorly on the Logo competence tests were the "followers" in most play settings.

Concluding comments

The social world of the young child is recognised as critical to cognitive development and learning in the preschool years. In the study from which the information presented here is drawn, Logo competence was clearly associated with social play experiences rather than independent play.

Fine grained examinations of play associated with Logo competence support the importance of an interactive milieu for developing Logo skills. As contended by Vygotsky and subsequent followers, the cognitive scaffolding afforded within social play contexts seemed to activate thought processes and give meaning to events and actions. Central to this interactivity was the use of language as children orchestrated thinking and action to define and solve a range of problems related to guiding the turtle. Speech, both private and shared thus assumed an important metacognitive function in Logo play.

In summary then, it seems that peer mediated play provided ongoing stimulation and motivation for learning, as well as support of a more metacognitive, and particularly, self-regulatory nature. The greater Logo competency displayed by children who engaged in social play suggests that these mechanisms, or aspects of them, may have become incorporated as part of learners' independent repertoire of competencies and appropriately activated in the Logo testing situation.

Identifying and highlighting these strategies serves to focus on their potential importance in mathematical contexts. In an early childhood classroom , early development of these strategies might serve to pave the way for an easier passage to mathematics competence in the later years.

REFERENCES

Anselmo, S. & Zinck, R.A. (1987). Computers for young children? Perhaps. *Young Children*, 42 (3), 22-27.

Bearison, D.J. (1982). New directions in the study of social interaction and cognitive growth. In F.C. Serefica (Ed.), *Social-cognitive development in context* (pp. 199-221). New York: Guilford Press.

Blemings, S. (1988). *Microcomputers in Queensland preschools: A study*. Brisbane: Queensland Department of Education.

Brown, A.L. (1987). Metacognition, executive control, self-regulation and other even more mysterious mechanisms. In F.E. Weinhert & R.W. Kluwe (Eds.), *Metacognition, motivation and learning.* (pp. 65-116). Hillsdale, NJ: Erlbaum.

Clements, D.H. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology,* 76, 1051-1058.

Clements, D.H. (1985). Research on Logo and social development. Logo Exchange, 5(3), 22-24.

Clements, D.H. (1986). Effects of Logo and CAI environments on cognition and creativity. Journal of Educational Psychology, 78, 309-318.

Clements, D. H. (1987). Computers and young children: A review of research, Young Children, 43(1), 34-44.

Clements, D.H. (1990). Metacomponential development in a Logo programming environment. Journal of Educational Psychology, 82, 141-149.

Clements, D.H. & Gullo, D.F. (1984). Effects of computer programming on young children's cognition. Journal of Educational Psychology, 76, 1051-1058.

Clements, D.H. & Nastasi, B.K. (1985). Effects of computer environments on social emotional development: Logo and computer assisted instruction. Computers in the Schools, 2(2-3), 11-31.

Clements, D.H. & Nastasi, B.K. (1988). Social and cognitive interactions in educational computer environments. American Educational Research Journal, 25, 87-106

Cooper, C.R. (1980). Development of collaborative problem solving among preschool children. Developmental Psychology, 16, 433-440.

Day, J. D., French, L. A., & Hall, L. K. (1985). Social influences on cognitive development. In D.L. Forrest-Pressley, G.E. MacKinnon & T.G. Waller, (Eds.), Metacognition, cognition, and human performance, Volume 1, (pp. 33-56). Orlando: Academic Press.

de Corte, E. (1990). Learning with new information technologies in schools: perspectives from the psychology of learning and instruction. Journal of Computer Assisted Learning, 6, 69-87.

Donaldson, M. (1978). Children's minds. London: Fontana.

Elliott, A. (1984). Computers in the preschool: Thinking it through. In J.Hughes (Ed.).Computers and education: Dreams and realities (pp. 43-46.). Sydney: CEGNSW.

Elliott, A. & Hall, N. (1985). First encounters with computers: A case study. Report to the Apple Education Foundation. Sydney.

Elliott, A. (1992). Effects of Logo play on cognitive competence. Paper presented at the International Congress of Psychology, Brussels.

Forman, E.A. & Cazden, C.B. (1985). Exploring Vygostskian perspectives in education: The cognitive value of peer interaction. In J.V. Wertsch (Ed.), Culture, communication and cognition: Vygotskian perspectives (pp. 323-347). Cambridge, England: Cambridge University Press.

Genishi, C., McCollum, P. & Strand, E. (1985). Research currents: The interactional richness of children's computer use. Language Arts, 62, 526-532.

Hofman, R. (1986). Microcomputers, productive thinking and children. In P.F. Campbell & G.G. Fein (Eds.), Young children and microcomputers (pp. 37-44). Englewood Cliffs, NJ: Prentice-Hall.

Hoyles, C. & Noss, R. (1987). Synthesising mathematical conceptions and their familiarisation through the construction of a Logo-based school mathematics curriculum. International Journal of Mathematics Education in Science and Technology, 18(4), 581-595.

Hughes M. & Macleod, H. (1986). Using Logo with very young children. In R.W. Lawler, B. du Boulay, M. Hughes & H. Macleod (Eds.), Cognition and computers: Studies in learning (pp.179-219). New York: Wiley.

Klinzing, D.G. (1985, April). A study of the behavior of children in a preschool equipped with a computer. Paper presented at the Annual Meeting of the American Educational Research Association, Chicago, IL. (ERIC Document Reproduction Service, No.ED. 255320).

Luria, A.R. (1973). The working brain: An introduction to neuropsychology. London: Penguin.

Luria, A.R. (1982). Language and cognition. J.V. Wertsch (Ed.) Washington.D.C.: Winston.

Miller, G.E. & Emihovich, C. (1986). The effects of mediated programming instruction on preschool children's self-monitoring. *Journal of Educational Computing Research*, 2, 283-297.

Newton, J. & Henderson, J. (1987). The computer in the preschool. In E. Shaw (Ed.), *Putting Kids First.* Proceedings of the ACT Computers in Education Group Annual Conference, (pp. 35-37), Canberra.

Noss, R. (1983). *Interim report of the Children's Logo Project.* Hatfield, England: Advisory Unit for Computer-based Education.

Noss, R. (1987). Children's learning of geometric shapes through Logo. *Research in Mathematics Education,* 18, 343-362.

Palinscar, A.S., Brown, A.L. & Martin, S.M. (1987). Peer interaction in reading comprehension instruction. *Educational Psychologist,* 22, 231-254.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York: Basic Books.

Papert, S. (1986). Are computers bad for children: A public or private affair? In P. Campbell & G. Fein (Eds.), *Young children and microcomputers.* (pp. 171-189). Englewood Cliffs, NJ: Prentice-Hall.

Papert, S. (1990). Introduction. In I. Harel (Ed.), *Constructionist learning*, (pp. 1-8) Cambridge, Mass: The Media Laboratory, MIT.

Pea, R. D. (1987). Integrating human and computer intelligence. In R.D. Pea & K. Sheingold (Eds.), *Mirrors of the mind: Patterns of experience in educational computing* (pp. 128-146). Norwood, NJ: Ablex.

Pelligrini, A. (1986). Communicating in and about play. In Fein, G. & Rivkin, M. (Eds.), *The young child at play.* (pp. 78-91). Washington, DC: National Association for the Education of Young Children.

Perret-Clermont, A.N. (1980). *Social interaction and cognitive development in children.* New York: Academic Press.

Quilitch, H.R. & Risley, T.R. (1973). The effects of play materials on social play. *Journal of Applied Behaviour Analysis*, 6, 573-578.

Rogoff, B. (1991). *Apprenticeships in thinking.* NY: Oxford University Press.

Salomon, G. (1990). Cognitive effects with and of computer technology. *Communication Research*, 17(1), 26-44.

Salomon, G., Globerson, T. & Guterman, E. (1989). The computer as a zone of proximal development: Internalizing reading related metacognitions from a reading partner. *Journal of Educational Psychology*, 89, 620-627.

Salomon, G., Perkins, D. & Globerson, T. (1991). Partners in cognition: Extending human intelligence with intelligent technologies. *Educational Researcher,* 20(2), 2-9.

Schwartzman, H.B. (1978). *Transformations: The anthropology of children's play.* New York: Plenum.

Shure, M.B. (1963). Psychological ecology of a nursery school. *Child Development,* 34, 979-992.

Sutton-Smith, B. (1979). (Ed.), *Play and learning.* New York: Gardner Press.

Try, K.M. (1989). Cognitive and social change in young children during Logo activities: A study of individual differences. Unpublished doctoral dissertation. University of New England, Armidale, NSW.

Vandenberg, B. (1981). Environmental and cognitive factors in social play. *Journal of Experimental Child Psychology*, 31, 169-175.

Vygotsky, L. (1962). *Thought and language.* Cambridge, MA: MIT Press. (Original work published 1934).

Vygotsky, L. (1966). Development of the higher mental functions. In A. Leontiev, A. Luria & A. Smirnov (Eds.), *Psychological Research in the USSR,* Vol.I, (pp. 11-46), Moscow: Progress Publishing.

Vygotsky, L.S. (1976). Play and its role in the mental development of the child. In Bruner, J.S., Jolly, A. & Sylva, K. (Eds.), *Play: Its role in development and evolution,* (pp. 537-534). New York: Basic Books. (Original work published 1933).

Vygotsky, L.S. (1978) *Mind in society : The development of higher psychological processes.* M. Cole, V. John-Steiner, S. Scribner & E. Souberman, (Eds.). Cambridge, MA: Harvard University Press.

Wertsch, J.V. (1991). *Voices of the mind.* Cambridge, Mass: Harvard University Press.

Wright, J.L. & Samaras, A.S. (1986). Play worlds and microworlds. In P. Campbell & G. Fein (Eds.), *Young children and microcomputers* (pp. 73-86). Englewood Cliff, NJ: Prentice-Hall.

Logo as an Assessment Tool

Sophia Koukouvinou[1], Stathis Triantafillou[2], Eleni Marouka[3]

[1] Pammakaristos Foundation
1 Marathonos Str.
190 05 Nea Makri
tel: +30 294 92306

[2] Theotokos Foundation
Nea Liosia Attiki
131 22 Greece
tel: 30 1 2311070, fax: +30 1 2314888

[3] Microelectronics Educational
5 Loudia Str.
115 23 Athens, Greece
tel: +30 1 6918286, fax: +30 1 6493461

Abstract

The assessment of a child's essential skills in a number of basic cognitive abilities such as memory, visual discrimination, categorisation, number concept, is a prerequisite for the development of a suitable program to meet his/her educational needs. With this project an attempt is made to investigate the use of Logo environment in one more area of the educational process, namely the assessment of children's cognitive skills, proving thus how effectively can be used by specialists working in the field of education.

Keywords: Logo, cognitive development, assessment, microworlds

Sophia Koukouvinou *has a bachelor and masters degree in Psychology from Surrey University and has been specialised in learning disabilities. She has been working with mentally retarded children as well as with children with special learning disabilities at Pammakaristos' special schools, where she is also concerned with the application of information technology in the field of special education.*

Stathis Triantafillou *is a BA Psychology graduate of a Canadian University. He has been working in the field of Rehabilitative Technology for the past 10 years. His work specifically concerns software development and special peripherals for normal pre-schoolers & mentally and physically handicapped children. Work places: Theotokos Foundation, Social Work Foundation, Play & Learn Computers, Sakellariou-Kavaratzi School and Institute of Informatics Crete.*

Eleni Marouka *did her educational studies and received her first course on the application of computers to education at Patras University. She joined Microelectronics Educational in 1991 and has been in charge of the teacher training at the Maraslios Academy and of the "Adventures of Leto" development project. She is interested in continuing her research in computer educational environments.*

Logo as an Assessment Tool

Introduction

Assessment can be seen as an attempt to observe and collect information systematically in a short period. The 'act ' of assessment really focuses our attention as we look at a number of attributes that contribute to a successful learning experience for a given child. It is a vital prerequisite to teaching as it allows the educator to arrive at a datum point from which to start. Furthermore, remediation can only follow from having first identified a particular learning difficulty.

The cognitive school of thought favours psychometric testing which are skill specific tasks looking at areas such as memory (both visual and auditory), visuospatial ability, verbal conceptualisation, arithmetic etc. From the data obtained it is possible to analyse the child's problems in terms of the questions "What strengths / weaknesses has a particular child?".

These measurable strengths and weaknesses are then used to form a profile and can be matched against the population norm, in order that suggestions about particular teaching strategies and methods can be made. (Thomson, 1988).

Computer based assessment

Having considered the necessity for assessment of a child's basic cognitive skills, the methodology that best arrives at an accurate evaluation of those skills, must be examined.

As the competence versus performance issue has always been a problem of the procedure traditionally adopted by psychologists, computer based assessment gave a solution to several of the difficulties arising in testing situations.

Motivation can be enhanced by colourful graphics, active participation is ensured by the interaction with the computer, self-confidence and absence of anxiety settle as the conditions resemble a play rather than a testing situation and, finally, rapport with the examiner is easily established.

Using the assessor

The development of an assessing tool within the LogoWriter environment called *"Assessor"* made possible the use of Logo in that particular aspect of the educational process. Using the Assessor a number of items can be presented on the screen for a given amount of time, the correct one chosen and the answers recorded. The graphics capabilities of Logo that make it suitable for development of attractive software for children, as well as the ability of the language to adapt to different level users, determined the implementation of this particular environment in the evaluation process.

The programme was designed to examine three broad areas of cognitive development, namely:

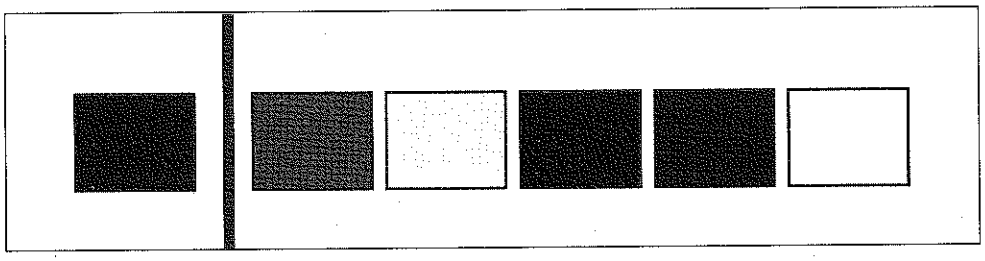
- visual discrimination,
- visual and auditory memory,
- categorisation and
- the number concept,

for children in the pre-operational stage of development or the iconic representation stage as described by

Bruner (1966). The tasks were created in such a way so that children would not be discouraged by the increasing demands of the task at hand.

Description of the tasks

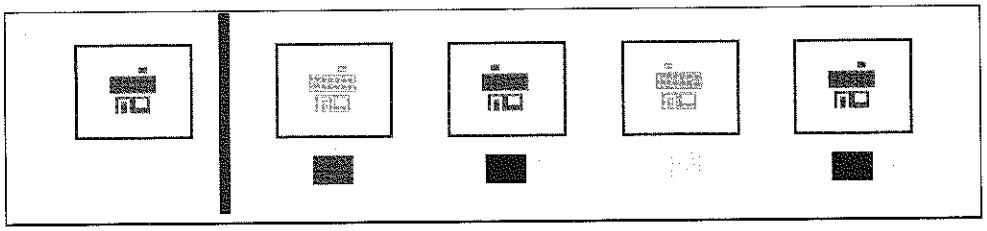
A. The first group of tasks refer to visual discrimination and they are divided into five levels of difficulty, beginning with discrimination of colour. The colour to be identified is presented on the left of the screen and is separated from a series of five colours, in which it is included, on the right.



Interaction with the computer is simplified by the use of an auxiliary keypad containing 5 keys of basic colours: red, yellow, blue, green and white. The keypad also has a slot where a template can be slipped by the instructor whenever required by the task.

In this particular task the child has to chose from the series of colours the one presented on the left, by just pressing the key with the corresponding colour.

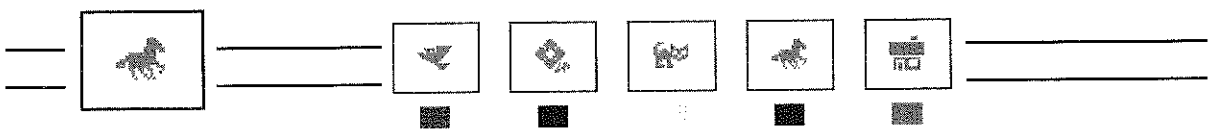
The next two tasks require the child to identify an object with relation to its position in space (up/down, left/right). S/he is then asked to identify an object among five that differ in a more detailed aspect, for instance the position of the chimney on the roof of a house. S/he is finally asked to identify an object among five that differ in two parameters, colour and detail.



In every case the choice is made by selecting the key with the corresponding colour.

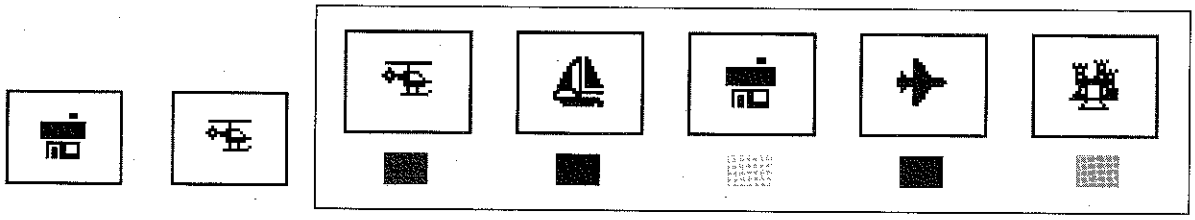
B. In the second category of tasks, the special feature of the assessor is put into full use, as the stimulus to be identified is presented on the screen for a limited amount of time.

More specifically, three kinds of stimuli (objects, shapes and numbers) are presented to the child for a given amount of time. When the stimulus to be remembered disappears, a series of two, three, four or five similar objects, shapes or numbers appear on the screen. The child has to recall which one was presented to him/her before and to respond by pressing the appropriate coloured key.



The objects as well as the shapes were formed using the "shapes" facility of LogoWriter and were chosen to be as much clear and familiar as possible.

The most difficult task in this category refers to memory for sequences: A series of two to four objects appear on the screen for a few seconds and once they disappear, the child has to press the keys in the right order.



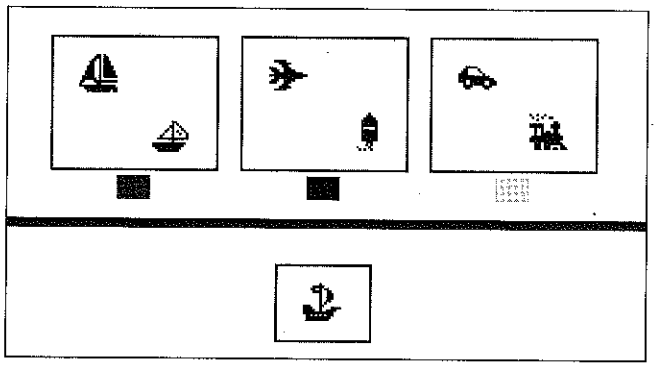
One has to note at this point that templates (cards with the printout of the presented objects) are placed on top of the keys, so that the child has only to retrieve from memory the sequence of objects. In every case, the instructions are given by the educator orally, in order that the screen is free from any interfering stimuli.

Lastly, auditory memory is tested using another of assessor' s features called "sound", with which one can produce melodies using the computer internal speaker.

A familiar sound is presented to the child, namely the siren of an ambulance, the sound of a train or ship and a short section of the National Anthem. S/he is then asked to respond by pressing a key, when s/he recognises the sound previously presented.

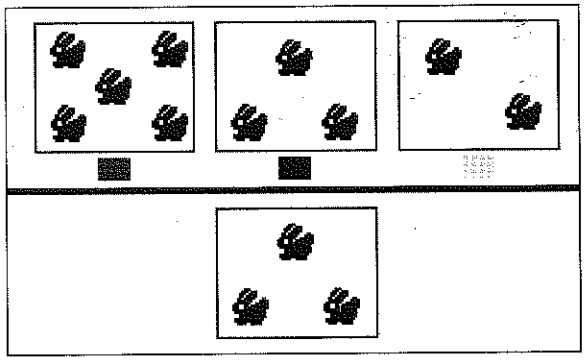
During the presentation of all sounds, the screen remains blank, so as to avoid interference effects.
A record of the correct responses is automatically made by the assessor.

C. The third group of tasks, examines the ability of the child to assign an object to a category, according to common characteristics. It is divided into four levels of difficulty, ranging from assigning an object to a group of identical ones, to assigning an object to a subgroup of the same category. For example, a boat has to be assigned to the group of means of transportation which are used at sea rather than on the ground or in the air.



In all cases, the choice is made between two or three groups and the object to be assigned remains on the screen for the whole duration of the task. The response is always made via the coloured keys which correspond to squares of colour under each group of objects.

D. Finally, the concept of number can be examined by matching either the quantity or the number to the appropriate quantity. A choice between three quantities must be made, using the keys.



Conclusions

It must have become obvious by now, that one can benefit from the Logo environment in a number of different ways, not only by using it as an "educational" tool per se but also as a tool to compliment the various activities related to learning.

Thus, having evaluated a child's essential skills and weaknesses, one can then design microworlds to fit the individual needs. There the child can use the turtle and its travels to discover different shapes, to recall objects and sounds, to categorise concepts and understand numbers. It could, that is, be used as a guide for developing microworlds suitable for very young children or children with special needs.

Our intentions are, that having modified the aforementioned application, one could then move onto designing such a microworld based on the conclusions drawn by the assessment.

Aknowledgements

We would like to thank Y. Kotsanis for his contribution in the development of *Assessor* environment.

BIBLIOGRAPHY

Bruner J.S., Olver R.R., and Greenfield P.M., *Studies in Cognitive Growth,* New York, Wiley (1966).
Goldenberg P., *Special Technology for Special Children,* Baltimore: University Park Press (1979).
Lawler R., *Computer Experience and Cognitive Development,* Wiley, New York (1985).
Papert S., *Mindstorms: Children, Computers and Powerful Ideas,* New York (1980).
Thomson, *Applied Cognitive Psychology 2, 19-33* (1988)

Logo, as a Studying Procedure

Ioanna Koutsogiorga

1 Kanari Street
Voula, Athens 166 73
Greece

Abstract

Logo, which has been created to make children learn and comprehend in a natural way, can also be a model for those teachers who would like to practise new "avenues" in teaching. Logo can prove in its own way, how collaboration between children can be effected. Logo improves children's observation, enables them to practise their working methods and broadens their creativity as they find pleasure in the results of their work. Logo with its methods and rules, can be a helpful teaching model for the teachers who doubt the results of more "open" educational system.

Keywords: Logo, children, teaching and learning environment, open education

Ioanna Koutsogiorga *is Co-founder and one of the major partners of the private school "Ergastiri" in Athens (1975 -1985). She has formed theatre workshops for children, organized groups of extra-curriculum activities for departments of Adult Education. Since 1986 she has been working on computing in elementary education. In 1988 she participated in a conference about Computers in Education, at Athens College, with a lecture on "Computers at Elementary School: Emphasis on the means or on the way?" In 1987 she presented Logo in children's magazine, through the comic series of "Telemahus' Routes". Articles of hers have been published in several magazines. In April 1993 she published her book with the title "About Childrens' Acquaintance with Computer and Logo".*

Logo, as a Studying Procedure

After so many years of application, Logo today should have been a modern tool in an expanded education. Since things do not always progress with the same rythm and education changes very slowly, Logo apart from being a dynamic tool could be yet something more: it could very simply prove, through its own practice how some paths in education can be opened.

I had the pleasure, of trying and, I dare say, with success, different forms of an open education, in an experimental school, based on the multi-aspect research, the arts in every form, the dialogue and constant communication among all members of the school community was being alternated in class as well as in all school areas. So for me this has always been a continuous challenge; that is the teaching that can lead successfully to other levels, to different areas of knowledge. Knowledge cannot be enclosed in chapters and the tutor should not become a reproductive machine of these elements of knowledge.

As it is already known, the use of Logo, takes an open educational system for granted. It was constructed by its creators under this base of creativity.

This was the essential element that drove me to get involved with Logo.

But how can all these elements be applied in a conventional elementary school, a school with a formal way of teaching? Because we have to see the reality in most schools.

In an education based on reproducing knowledge, how much and in what way can the power of Logo work as a tool of creativity?

Taking for granted that the desire for an open education exists, I set the question: Could the involvement in the Logo be an applied proof for an open form of teaching, for those teachers who are doubtful of the result?

There are some questions on the essence of education and its methodology, that those of us who work in this area should not forget. Each one's answers to the questions that follow show one's attitude towards education and one's contribution to its improvement.

Do children learn better in collaboration?
How is their observation excercised?
How oftern does the tutor assess himself in relation to his class?
In what degree does children's method improve?
How many chances do they have in their everyday practice, to become more creative?
How much do they enjoy learning?
How much do they practise in overcoming mediocrity?
How open are children to the world and the beauty?

Of course, I will not give my own answers to the above questions. Instead, I will try to answer them in a short way drawing from my experience in working with Logo in Greek elementary schools.

Team work

When children work in groups of 2 to 3 on a computer they have to communicate with each other in order to achieve their common target.

Thus children are given a splendid opportunity to try themselves in team-work. I say try, because nowadays, in the era of individuality, team work is something that needs practice to succeed and though it is most

neccessary in life, school usually give little opportunity to students to develop collaboration.

Those of you with school-class experience know better that collaboration is one of the most difficult problems we face. Replacing a member of a working team, when the first friction or tension takes place, is for the teacher, an easy solution out. But this will not help any of the members of the team or the individual student. It will only bring temporary peace.

Here the role of the tutor is crucial since he must insist on a definite solution to the problem. A solution that will be accepted unanimously aiming at the weakest members of the team coming out stronger, the strongest ones managing to control themselves and the less bold, learning to say their opinion and standing up for their rights.

This can be achieved through contrinuous effort. The tutor is tested in action whether he can keep the balance being himself in collaboration with the children.

Logo gives everyone the opportunity to make an effort. It is understood that continuous effort is well demanded from the part of the teacher because friction is frequent. But the balance can be achieved and the results are excellent after a while, especially so when a team with difficulties in collaboration manages to terminate a project.

Being in a working-team, means giving and taking. In the Logo area, as you know, there are no irreversible mistakes, but continuous alternative solutions. So there is plenty of space for further improvement in the relationship between the members of the team.

Observation - Interest - Exertion beyond the mediocrity.

It is well known that the only way to motivate children to learn, is to give them incentives that attract their interst. This is why in class, I try to give every exercise a special element which will attract the student's attention, make them more observative and guide them one step forward. We can exercise children observation by working on a procedure with Logo.

For example I suggested to the students of the fifth grade that they draw the facade of their school. This was done to make them apply the repetitive procedures, on new procedures, practising on such new elements as doors, windows, balconies etc. These took the place of sub-procedures. Many of the children asked me if they could go out to see the facade of the school!

Can you imagine? They were asked to describe something that they had been looking at for five years every day and yet they need to look at it!

Children, must be familiar with day to day reality, in order to be able to get beyond it and it is up to us, tutors, to give them continuous practice on the task. Children form mental pictures on different subjects influenced most of the time, by what they have seen on T.V. rather than the reality. Let's get them to observe their surroundings more, in order to get them to know the world better.

Taking into consideration the level of the children, I work on the principle of demanding something "extra" for the accomplishment of a task, something that the children themselves might have thought of on their own. This aims at training them to become more demanding from their work.

For instance when dealing with a common subject such as a house, I discuss with every group what more they could do to differentiate their work from the work of other children or I suggest what they could add to their drawings.

We must increase our demands little by little, asking more and more from the children in order to avoid getting from them what most of them are willing to give i.e. the least.

Imagination - Creativity

Nowadays, children don't have the time to create their own mental pictures or to imagine things. Therefore to get them to exercise their imagination is a problem for the tutor because as children are continuously brain-washed by ready made pictures, they take them for granted, instead of making an effort to create something new. This is something that has to be learned. Logo, could contribute to it with its sense of space.

The one-dimension image in a certain space is ruled out as there can be various different positions of this image on a certain surface. Children become very fascinated by turn arounds, up-side-downs and symmetries when they have the opportunity to find them out and use them on the PC screen.
You must have seen their reactions when they manage to print their little "robot" up-side-down on the screen.

Logo gives children the joy of discovering that they are able to create more sophisticated pictures by simple means.

We must never forget that for children few things are taken for granted. So they become surprised seeing the flower they have drawn, printed in various positions or in a multiprint. This is something that has seldom crossed their minds before and as a result, they become over excited. The repetition that gives different results can only be discovered through various and different applications.

The existing educational system, with the overloaded syllabus, does not permit children most of the time, to see knowledge as something flexible that can be seen from different angles and has many different applications.

For example, when children have practiced on a certain type of mathematical problem, they believe they can solve a problem of the same type. But how often can they transfer successfully this classroom activity in daily life practice? Therefore it is imperative that children realise that classroom practice is not done to fill up pages or to cover up a subject.

In the same sense, how often during the lesson, do students try the use of the same word in different meanings? The different application of one and the same thing in Logo, does not fail to astonish children every time which shows how very different the thing seems to them, when seen from an other angle.

Logo, helps us to understand better how children really learn. For instance, we observe their repeating the same thing over and over again. We then realise that if let to choose on their own the subject they want to draw, most of them will choose something they are familiar with.

Let us not forget that children do want to succeed and they do enjoy their success. So let's give them the chance to do it. Repetition is necessary as long as they need it to conquer something.

How ready are we, teachers, to understand it, since we are always running out of time?

How long can we stand by before we decide how and when to proceed further?

Allowing time for repetition seems something quite simple; however the syllabus reality is so pressing that it makes us disregard it.

Many times, I rushed to label something as very simple and therefore something easy and self-understood which could be used further. In this way I left little room for repetition. I was anxious to proceed to the next step. I was wrong. That was easy and self-understood only by me.

Once I was very astonished, with something the children considered a great discovery something that for adults would seem at least naive. After several lessons and many efforts, a group made the discovery of

using the drawing of a leaf -that they had already saved like a procedure- as an eye on a face. They were very happy and proud. Now, I rejoyce together with them, since I can see everything as something new.

The above example in Logo could be applied to other lessons in class.

The pupil learns to write for example the word "newsweek" correctly. He realises with surprise that this word consists of two words he already knew how to write.

The surprise that children feel can be turned into knowledge, if given the chance to understand, through examples of their own, that multi application is something quite common. There is a relation between things, knowledge is interrelated and nothing stands on its own.

To refer back to Logo, it is a fact that children are able to accept an unconventional element and fit it to their design, which can be further exploited so as to train them into being open to different sources of knowledge. Therefore it would enable them to suffle unsimilar elements in such a way so as to create something new and original.

Method of work

Can the way children work be improved and how?

Logo provides an interesting and direct method. There are certain stages to be followed in order for pupils to create a successful design. They will start with the clear definition of their final aim which will lead them to a good working plan. They will proceed with organizing the different elements into subprocedures and finally the procedure will be finished and presented in class. So they come to the final printed design. And it is true that the same method is required of a composition or a project on an other subject such as history.

It is understood therefore that the same methodology can be used in reverse: instead of beginning by giving the theory of composition writing, the teacher can get to it, through the tasted practice of the Logo. The model of the Logo method, can be thus transferred into the organisation in language or mathematics.

Game - Joy

In order for the children to become acquainted with the movements of the turtle it is necessary to make it alive in class by acting. The children get up and move about recreating the turtle movements. There is therefore a kind of role-playing in class. Children certainly enjoy such things.

I wonder how oftern teachers find the right stimuli that bring role-playing every day classroom activities in order to allow their class to be carried away or even to releave tension.

I have had experience in theatrical workshops that have been fascinating. Acting, I found out, gives great pleasure to children while at the same time it sets them free. Schools should offer such opportunities as much as possible. I believe that Logo can serve as yet one more spring-board. Some teachers could be inspired by a small play on the Logo to continue with another play in class on another subject.

I consider Logo as being a creative game and I would not agree with the use of a book on it in class that would make it look like a lesson. This is the principle governing my book. I wanted to present my opinion on how Logo should be taught in elementary schools. At the same time, I wanted my book to be a suggestion on how I would like school to be: a book that would give pleasure to children going through it, where s/he could find the instructions she is looking for, but which would also offer an other dimension to life. I believe that games, knowledge, excursions, information should always challenge children to get out of isolation.

I would never wish for a child to be sitting infront of a screen for hours on end. On the contrary a child who knows how to work with a computer, knows how to play intelligent games but he most importantly knows when to turn it off and run to play ball, talk or go for a walk with friends. This is what I honestly wish for every child.

To sum up, I have insisted on an open educational system taking it as the final aim eventhough for some of you this is a prerequisite. My experience of the Greek Elementary school tells me that there is still broad scope for new openings. Thus I have presented a pedagogic view reversing the terms, touching upon pedagogic practical problems through the use of a PC and Logo. We should therefore investigate into how we can transfer the openmindness of Logo, to the educational system instead of broading an open educational system to include computers. This is so because Logo provides a dynamic means that is beyond the narrow limitations of a PC.

It would be at least naive to regard Logo as the solution to methodology. But certainly regard it as an interesting challenge for the teacher which offers the game of knowledge more and varied paths.

Bibliography

[1] Abelson, and diSessa. Turtle Geometry. Campridge, Mass.: MIT Press 1981

[2] Papert . Mindstorms: Children, Computers and Powerful Ideas. New York: Basic Books, 1980.

[3] Piaget, J. Secience of Education and the Psychology of the Child. New York: Viking Press, 1970.

[4] Watt, Learning with Logo. New York: McGraw Hill, 1983

[5] Watt, Learning with Apple Logo. New York: McGraw Hill, 1983

Inducing Knowledge from Examples

Dunja Mladenic

J. Stefan Institute
Computer Science Department
Artificial Intelligence Laboratory
Jamova 39, 61000 Ljubljana, Slovenia
tel: (+38)(61) 159 199, fax: (+38)(61) 161 029
e-mail: dunja.mladenic@ijs.si

Abstract

The paper presents way of explaining knowledge induction process to help students to understand it, and experiment by means of computer program. The knowledge is induced from examples of some situation with a sort of generalization that use entropy measure to guide the generalization process. Set of examples can be changed to enable analysis of influence of new examples to the induced knowledge. Simple addition of a new example that describe the same situation, possibly leads to modification of the induced knowledge. The process is illustrated with the simple algorithm implemented in LogoS. Furthermore, the algorithm is explained in details to enable simple modification and use with some other input examples. The induction algorithm implemented in LogoS is one of the well known machine learning algorithms. It has been developed in a few ways to some poverful algorithms that have been successfully applied on a lot of real-world problems, to induce knowledge about some real situation. One of widely known real-world application is in medical diagnosis where machine learning system is used to predict a patient diagnosis from the symptoms.

Keywords: teaching your computer, knowledge induction, machine learning

Dunja Mladenic *was born on April 14, 1967 in Pula, Croatia. In 1985 he started studying computer science at the University of Ljubljana, Faculty for Electrical Eng. and Computer Science, Slovenia. He has made his B.Sc. thesis, the "Machine learning system Magnus-Assistant", in the Artificial Intelligence Laboratory, Computer Sc. Department, J. Stefan Institute and graduated on June 29, 1990. In September 1992 he started the postgraduate study of Computer Science at the University of Ljubljanan. Since 1992 he is involved in development of croatian version of LogoS. His main research areas are machine learning and logic programming.*

Inducing Knowledge from Examples

1. Introduction

Knowledge induction is the ability that we oftenly need in our everyday life. For example, observing the current weather situation, we can try to predict the weather for the next day. With days, or better with years of observation our predictions are becoming more and more correct. This process can be viewed as induction of knowledge contained in the weather examples. Each example consists of some values describing the weather situation at the appropriate day and is connected to prediction of the next-day weather. Such induction process can be used as the basis for programming the computer to induce knowledge. The idea of knowledge induction from examples is already known in the *machine learning* , area of the computer science field called *artificial intelligence* .

Section 2 briefly describes the idea of learning from examples. Section 3 introduces the machine learning algorithm used in LogoS program for knowledge induction. The LogoS program is illustrated and discussed in Section 4. Section 5 gives a brief discussion of the presented work. The part of LogoS program that is not in Section 4 is given in Appendix.

2. Problem description

When *learning from examples* set of examples containing knowledge about the target situation is used as program input. Each example can be viewed as a n-tuple of *attributes* where each attribute describes one property of the concrete situation (e.g. weather situation at the appropriate day). To each example, a value called *class* is assigned (e.g. prediction of the next-day weather). The problem is to induce knowledge describing value of the class by means of the attribute values.

To illustrate the way of knowledge induction with computer, one of the basic machine learning algorithms ID3 [3] has been implemented in LogoS. The algorithm belongs to the family of machine learning algorithms called Top Down Induction of Decision Trees. The main characteristic of these algorithms is that knowledge induction is performed using top-down approach and is represented by means of decision tree. *Decision tree* consists of nodes containing tests of attribute values and leaves containing the predicted class values. Once generated, the tree can be used to classify new (unseen) examples, i.e. to decided which class value a new example has. This is done by proceeding the new example from the root of the tree to the appropriate leaf. Each step consists of testing the current node attribute value example has, and proceeding to the subtree according to the test result. The class value assigned to the final node-leaf is the class predicted to the proceeded example.

2.1 An example problem domain

To describe learning from examples as performed by ID3 algorithm a simple 'Saturday morning' [3] domain is used. In this machine learning domain, 14 training examples are given (see Table 1). Each example is labelled with its class: suitable (no, yes), saying if a morning is suitable for some unspecified activity, and 4 attributes: outlook (sunny, overcast, rain), temperature (hot, mild, cool), humidity (high, normal) and windy (no, yes), describing a Saturday morning.

example number	class suitable	attributes outlook	temperature	humidity	windy
1	no	sunny	hot	high	no
2	no	sunny	hot	high	yes
3	yes	overcast	hot	high	no
4	yes	rain	mild	high	no
5	yes	rain	cool	normal	no
6	no	rain	cool	normal	yes
7	yes	overcast	cool	normal	yes
8	no	sunny	mild	high	no
9	yes	sunny	cool	normal	no
10	yes	rain	mild	normal	no
11	yes	sunny	mild	normal	yes
12	yes	overcast	mild	high	yes
13	yes	overcast	hot	normal	no
14	no	rain	mild	high	yes

Table 1: Examples for the 'Saturday morning' domain.

The set of examples given in Table 1 contains the information about the 'Saturday morning'. Some examples describe mornings that are suitable for some unspecified activity (class = `yes`) and the other describe unsuitable mornings (class = `no`). Looking at the first example in Table 1. (`no, sunny, hot, high, no`), one can see that morning is not suitable if `outlook` is `sunny` , `temperature` is `hot`, `humidity` is `high` and `windy` is `no` .

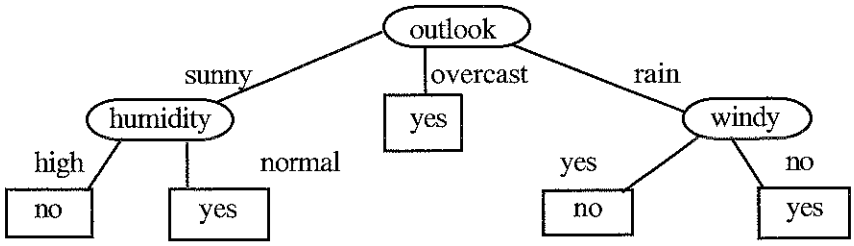


Figure 1. Desicion tree for 'Saturday morning' domain.

To induce some knowledge about the suitability (class) from the given set of examples, one will search some regularities (correlations) between the class and the attribute values. Focusing the attribute `outlook` in Table 1, it is easy to see that always when `outlook` is `overcast` the class is `yes` (notice that the middle leaf of the tree in Figure 1 is saying exactly the same as we have discovered just looking at the table). Described machine learning algorithm is performing its induction in the similar way. Decision tree induced by the ID3 algorithm from the whole set of examples is given in Figure 1 where ovals labelled with the attribute name denote nodes that test attribute value, and rectangles labelled with the class denote leaves predicting the class.

The tree can be interpreted (starting with the most left leaf) in the following way:

if outlook = sunny and humidity = high **then** suitable = no
if outlook = sunny and humidity = normal **then** suitable = yes
if outlook = overcast **then** suitable = yes
if outlook = rain and windy = yes **then** suitable = no
if outlook = rainy and windy = no **then** suitable = yes

A new example about the same situation (same class and attributes) can be classified using the above tree. Suppose that today's weather can be described by means of attributes as `outlook` is `rain`, `temperature` is `hot` , `humidity` is `normal` and `windy` is `no` . We want to know if the day is suitable (if class is `yes`). The class is predicted using the above tree so that example is proceeded from the root of the tree to appropriate leaf. `Outlook` is `rain` and `windy` is `no` leads to the most right leaf labelled with class `yes` . In this situation it is obvious that `temperature` and `humidity` are not important.

3. Algorithm used for knowledge induction

Machine learning algorithm ID3 [3] induces knowledge from examples and presents it in the form of decision tree. Decision tree consists of nodes, each containing test of an attribute value, and leaves labelled with the class value. Illustrated on the tree from Figure 1, the root of the tree tests value of the `outlook` attribute and has three subtrees, one for each value of the attribute. To each node/leaf a set of learning examples is assigned, so that all examples satisfy all the tests (of attribute values) that appear on the path from the root to the current node. Illustrated on the same tree, to the root the whole learning example set is assigned, since there is no test before the root. To the left branch (node with attribute `humidity`) only examples with the `outlook = sunny` are assigned, to the most left leaf (label with class `no`) only examples with the `outlook = sunny` and `humidity = high` are assigned. The union of example sets assigned to all leaves should be the whole learning example set.

The main ID3 algorithm is as follows:

ExSet := set of examples;
if all examples in *ExSet* have the same class *C* **then**
    Make a leaf labelled with the class *C*
else
    Find the most informative attribute to be the root of the current decision tree;
    According to the value of the chosen attribute split *ExSet* into subsets;
    Repeat the whole process for each subset; { generate subtrees }
endif
output decision tree;

The main ID3 algorithm constructs decision tree from examples starting with the whole set of examples (root), splitting it into subsets according to the value of chosen attribute and preforming the same procedure to each subset. When some subset consists only of examples of the same class, the process is stopped by construction of the leaf. The most informative attribute is find according to the informativity calculated for each attribute using the following formula:

$$Informativity(\ Attribute\ ) = Entropy(\ ExSet\ ) - Entropy(\ Attribute)$$

$$Entropy(\ ExSet\ ) = \sum_i p_i \log(\ p_i\ );\quad Entropy(\ Attribute\ ) = \sum_v p_v * Entropy(\ ExSet_v)$$

where p_i is the probability of i-th class in the set *ExSet*, p_v is the probability of the v-th value of attribute and $ExSet_v$ is the example set assigned to v-th branch.

Illustrated on the examples give in Table 1, the entropy of the whole set is calculated from the set of 14 examples, 5 with class `no` and 9 with class `yes`. $Entropy(\ ExSet\ ) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$ bits. Calculating the informativity for the `outlook` attribute that has three values (`sunny`, `overcast`, `rain`) 5 of the 14 examples have the first value, 2 of them from class `yes` and 3 from class `no`. The probability for the first branch is $p_1 = (5/14)$ and its entropy is $Entropy(\ ExSet_1) = -(2/5)\log_2(2/$

5) - (3/5) $\log_2$ (3/5) = 0.971 bits. Entropy of the rest two branches is calculated in the same way. The entropy and informativity of attribute `outlook` are then calculated:

$$Entropy(\ outlook\ ) = (5/14) * 0.971 + (4/14) * 0 + (5/14) * 0.971 = 0.694$$

$$Informativity(\ Attribute\ ) = 0.940 - Entropy(\ outlook\ ) = 0.246$$

4. Description of the Logo implementation

Program is written in LogoS [1, 2] but can be easily adapted for any other LogoS interpreter. It implements described knowledge induction algorithm and consist of three main parts; the first reading domain (description of attributes and classes, and examples), the second generating decision tree and the third writing decision tree to screen. The main procedure `To ID3` illustrates knowledge induction process starting with tree generation from a single example and adding examples one by one. The last generated tree (from the 14 examples) is the same as given in Figure 1.

```
CmLn "ID3 End
To ID3
  TxScr
  Local "TrainExL
  Make "TrainExL []
  RdDomain ;read domain into Class, AttrL and ExL
  Repeat Length :ExL {
    Make "TrainExL AddLast ( Item GetIter :ExL ) :TrainExL
    Make "Tree ( GenTree :Class :AttrL :TrainExL ) ;generate decision tree
    Clr
    print "' ' type GetIter
    type "'. example added :' print ( Item GetIter :ExL ) print "' '
    WrTree :Tree 1 ;write decision tree
    Wait
  }
End
```

4.1 Read of domain description and examples

The `RdDomain` procedure sets values of three variables. `Class` is list of class values (`[ no yes ]`) and must contain all possible values of the class. `AttrL` is list of attributes, each attribute is represented with list of two elements (`[ outlook [ sunny overcast rain ]`) the first is attribute name and the second is list of attribute values. `ExL` is list of learning examples where each example is represented with list of values (`[ no sunny hot high no ]`), where the first (class) value is followed by the attribute values given in the same order as the attributes in the domain description (`AttrL`). Example set thus looks very similar to the examples given in Table 1. Experiment with some other domain is simply performed by changing the contents of these three variables in procedures `GetClass` , `GetAttrL` and `GetExL` .

```
To RdDomain
  Make "Class GetClass
  Make "AttrL GetAttrL
  Make "ExL GetExL
End
```

```
To GetExL
  Output [
      [ false sunny hot high false ]
      [ false sunny hot high true ]
      [ true overcast hot high false ]
      [ true rain mild high false ]
```

```
To GetClass
  Output [ false true ]
End

To GetAttrL
  Output [
    [ outlook [ sunny overcast rain ]]
    [ temperature [ hot mild cool ]]
    [ humidity [ high normal ]]
    [ windy [ false true ]]
  ]
End
```

```
    [ true rain cool normal false ]
    [ false rain cool normal true ]
    [ true overcast cool normal true ]
    [ false sunny mild high false ]
    [ true sunny cool normal false ]
    [ true rain mild normal false ]
    [ true sunny mild normal true ]
    [ true overcast mild high true ]
    [ false rain mild high true ]
    [ true overcast hot normal false ]
  ]
End
```

4.2 Write decision tree

Decision tree is written simply to text screen by the procedure `WrTree` that has two parameters, `Tree` and `Level` . The first is generated decision tree and the second is current level in the tree. `Level` enables indentation, by means of typing ' ', so that all nodes from the same level of the tree (same path length from the tree root) are left justified. For example, the tree shown in Figure 1 is written as follows:

```
attr = outlook
 (sunny)
   attr = humidity
   (high)
    class = no
   (normal)
    class = yes
 (overcast)
   class = yes
 (rain)
   attr = windy
   (yes)
    class = no
   (no)
    class = yes
```

What follows is the procedure that writes decision tree in the described way.

```
To WrTree :Tree :Level
 ( Local "Attr "AVal "SubTrees )
 If (( Item 1 :Tree ) = "Null ) [
   Repeat :Level[ type "' ' ]
   print "Null
 ][
 If (( Item 1 :Tree ) = "Leaf ) [
   Repeat :Level[ type "' ' ]
   type "'class ' type "'= ' print ( Item 2 :Tree )
 ][ ; ( Item 1 :Tree ) = "Branch
   Repeat :Level[ type "' ' ]
   type "'attr ' type "'= ' Make "Attr ( Item  2 :Tree )
   print ( Item 1 :Attr ) ;attribute name
   Make "AVal ( Item 2 :Attr ) ;attribute values
```

```
    Make "SubTrees ( Item 3 :Tree ) ;subtrees
    Repeat Length :SubTrees {
      Repeat :Level { type "| }
      type "'( ' type ( Item GetIter :AVal ) print "' )'
      WrTree ( Item GetIter :SubTrees ) ( :Level + 1 )
    }
  }}
End
```

4.3 Decision tree generation

Decision tree is generated with the procedure `GenTree` that implements the main ID3 algorithm described in Section 3. It has class, attributes and examples as parameters. The first line in the below code is needed because the procedures `GenTree` and `GenSubTrees` are calling each other.

```
 ForwTo GenTree, 3 End
To GenSubTrees :Class :AttrL :SplitedExL
  Local "TreeL
  Make "TreeL [ ]
  Repeat Length :SplitedExL {
    Make "Tree GenTree :Class :AttrL ( Item GetIter :SplitedExL )
    Make "TreeL AddLast :Tree :TreeL
  }
  Output :TreeL
End


To GenTree :Class :AttrL :ExL
  ( Local "Best "BestAttr "BestSplitedExL )
  If EmptyP :ExL {
    Output List "Null
  }{
  If SameClass :ExL {
    Output ( List "Leaf ( Item 1 ( Item 1 :ExL )))
  }{
    Make "Best FindBestAttr :Class :AttrL :ExL
    Make "BestAttr Item 1 :Best
    Make "BestSplitedExL Item 2 :Best
    Output ( List "Branch :BestAttr
     ( GenSubTrees :Class :AttrL :BestSplitedExL ))
  }}
End

To SameClass :ExL; check if all examples are of the same class
  Repeat Length :ExL{
    If ( Item 1 ( Item GetIter :ExL )) <> Item 1 ( Item 1 :ExL ) {
      Output "false
    }{}
  }
  Output "true
End
```

Procedure `GenTree` consists of three parts, two simple stopping criterion and the third containing recursive call. In the third part the most informative attribute is found and example set is splited according to it values (`FindBestAttr`). Procedure `GenSubTrees` is called with the domain information (class, attributes) and the list of splited example set. It just calls `GenTree` again for each element of the list (each example set) and generates list of subtrees `TreeL` .

`FindBestAttr` is procedure that implements informativity/entropy formula described in section 3 and needs procedures given in Appendix.

5. Discussion

Simple knowledge induction algorithm described in the paper, can be used as illustration of the way of learning from examples. It is also implemented in LogoS and can be typed in computer to enable performing experiments. For the purpose of experiments and analyses, input data can be modified to describe some new situation. The idea is also for the students to get some opinions about way computer can perform learning, to try to be a teacher to computer.

Appendix

```
To SplitExL :Attr :AttrL :ExL; split set of ex. according to given attr.
  ( Local "SplitedExL  "Ex )
  ( Local "AValExL "AVal "AValL "AttrNum )
  Make "AttrNum NMember :Attr :AttrL
  Make "AValL Item 2 :Attr
  Make "SplitedExL []
  Repeat Length :AValL {
    Make "SplitedExL AddLast [] :SplitedExL
  }
  Repeat Length :ExL {
    Make "Ex ( Item GetIter :ExL )
    Make "AValNum NMember ( Item ( :AttrNum + 1 ) :Ex ) :AValL
    Make "AValExL AddLast :Ex ( Item :AValNum :SplitedExL )
    Make "SplitedExL Replace :AValNum :AValExL :SplitedExL
  }
  Output :SplitedExL
End

To CalcClassProbL :Class :ExL :Exs; calculate probability for each class
  ( Local "ProbL "Ex "ClassNum )
  Make "ProbL []
  Repeat Length :Class {
    Make "ProbL AddLast 0 :ProbL
  }
  Repeat Length :ExL {
    Make "Ex Item GetIter :ExL
    Make "ClassNum NMember ( Item 1 :Ex ) :Class
    Make "ProbL Replace :ClassNum ( Sum ( Item :ClassNum :ProbL ) 1 ) :ProbL
  }
```

```
  Repeat Length :ProbL {
    Make "ProbL Replace GetIter ( Div ( Item GetIter :ProbL ) :Exs ) :ProbL
  }
  Output :ProbL
End

To CalcEntropy :Class :ExL :Exs; calculate entropy for given ex. set
  ( Local "ClassProbL "ClassProb "Log2Prob "Entropy )
  Make "Entropy 0
  Make "ClassProbL CalcClassProbL :Class :ExL :Exs
  Repeat Length :Class {
    Make "ClassProb ( Item GetIter :ClassProbL )
    If :ClassProb > 0 {
      Make "Log2Prob ( Div Log :ClassProb Log 2 )
    }{
      Make "Log2Prob 0
    }
    Make "Entropy ( Diff :Entropy ( Prod :ClassProb :Log2Prob ))
  }
  Output( :Entropy )
End

To AttrEntropy :Class :AttrL :SplitedExL :Exs; calculate attr. entropy
  ( Local "ExL "AEntropy )
  Make "AEntropy 0
  Repeat Length :SplitedExL {
    Make "ExL ( Item GetIter :SplitedExL )
    Make "AEntropy ( Sum :AEntropy ( Prod ( Div Length :ExL :Exs )
     CalcEntropy :Class :ExL :Exs ))
  }
  Output :AEntropy
End

To FindBestAttr :Class :AttrL :ExL; find best attr. and split ex. set
  ( Local "BestAttr "BestSplitedExL "BestAttrInf )
  ( Local "Attr "SplitedExL "AttrInf )
  ( Local "Exs "Entropy )
  Make "Exs Length :ExL
  Make "Entropy CalcEntropy :Class :ExL :Exs
  Make "BestAttr 1
  Make "BestAttrInf 0
  Repeat Length :AttrL {
    Make "Attr Item GetIter :AttrL
    Make "SplitedExL SplitExL :Attr :AttrL :ExL
    Make "AttrInf ( Diff :Entropy ( AttrEntropy :Class :AttrL :SplitedExL :Exs))
    If :AttrInf > :BestAttrInf {
      Make "BestAttr :Attr
      Make "BestSplitedExL :SplitedExL
      Make "BestAttrInf :AttrInf
    }{}
```

```
  }
  Output ( List :BestAttr :BestSplitedExL )
End
```

REFERENCES

[1] Batagelj, V., Grobelnik, M. and Zupanic, D. (1991) The design of LogoS, *Third European Logo Conference Proceedings* , pp. 287-296, Parma, Italy.

[2] Grobelnik, M. and Zupanic, D. (1991) Inside LogoS implementation, *Third European Logo Conference Proceedings* , pp. 297-311, Parma, Italy.

[3] Quinlan, J.R. (1986) Induction of decision trees. *Machine Learning 1* , pp. 81-106. Kluwer Academic Publishers.

Logo Environment in Pre-School: a Mediation Tool

Afira V. Ripper

Laboratory of Education and Applied Informatics-LEIA
and
Interdisciplinary Center for Informatics and Education - NIED
State University of Campinas, Brazil
C.P 6120, Unicamp,
13085 Campinas, SP, Brazil
fax:55-192-540578

Abstract

The goal of this research is to investigate forms of relationship as generators of crescent levels of reflexive action and the ways in which a Logo-based Environment affects the development of functions in the spheres of written language and conceptualization of number. The theoretical framework is the socio-historical approach to questions of development and learning which can provide illumination to the phenomena. In discussing the relationships between development and learning/teaching, Vygotsky distinguishes two levels of development, the actual developmental level and the level of potential development, being the distance between them the space of shared capabilities, referred as the zone of proximal development (ZPD). The research approach is of qualitative nature, based on participant observation and considering as unity of analisis the triad child-mediator-computer. It is expected to clarify the relationships raised in the objectives above. The documentation is made through video tapes and field notes. The research context is a pre-school classroom with 17 children (10 boys and 7 girls). The mean age was four years at the beginning of 1991. In this class the activities are organized around "interest corners" such as "research", reading, visual arts, games and two computers with the Logo language. The data to be presented comprehend two years of observation.

Keywords: Logo environment, pre-school, ZPD, mediation tool

Afira Vianna Ripper *PHD, in Educational Psychology, Catholic University of São Paulo, São Paulo, Brazil. Professor of Educatioanal Psychology at the School of Education of the State University of Campinas, Campinas, SP since 1975. Presently heads the laboratory of Education and Applied Informatics of the School of Education and coordinates the Eureka Project to introduce informatics in the school system of the city of Campinas. Research on Logo since 1974, following a visit to Seymour Papert*

Logo Environment in Pre-School: a Mediation Tool

Introduction

The study of the teaching-learning process in a socio-cultural perspective has pointed out the relevance of cultural means of mediation for a qualitative shift in the functioning of psychological functions, which gives origin to the so called superior behaviour. Vygotsky (1978) characterizes the usage of sign and tool as mediated activity which will orient human behaviour in the internalization of higher mental functions. But they have a distint role, while sign constitutes an internal oriented activity aimed at controllig oneself (self regulation), the function of tool is external, aimed at controlling nature.

Valsiner (1991) proposes a theoretical-conceptual expansion of Vygotsky's ZPD to provide an understanding about manners of point of conctat between individual and social functions. Valsiner proposes, based on Piaget and Lewin, two other zones: the zone of freedom of movement (ZFM) and the zone of promoted actions (ZPA) characterized as the cognitive structure constructed and the ensemble of activities which guide the action of the subject. How these zones interplay in the ZPD and its consequences for the instructional situation give us a base to inquire in the pedagogical context, about the role of the Logo-based Environment while instrumental mediation of actions which can elevate the functionning for the subjet.

Our objective is, based on the discussion of mediational function and of the relationships betwen sign and tool, to study the influence of a Logo-based Environment on the learning process of pre-school children, in particular the process of written language acquisition. In the Logo Environment when the child is "teaching the turtle to make a drawing" she is dealing with the written language and mathematics, at the same time, with a kind of writing and syntax.

Sign Mediation

The written language has an important mediational role in the development of higher mental functions as observed by Luria in his experiments with literate and illiterate peasants. Based on the psychological studies of Luria on the development of written language, which led to the conclusion that young children are capable of discovering the symbolic function of writing, Vygotsky suggests the teaching of written language in the pre-school. Montessori, among others educators, showed this to be possible.

Vygotsky also pointed out that writing must be meaningful for children, an intrinsic need should be aroused in them: the written language and Mathematics must be incorporated in tasks seen both as necessary and relevant. He also stated that it is necessary to teach the written language in a natural manner, that is, the same way the child learns to speak. Our question is: in which ways a Logo-based learning environment can contribute to the learning of written language?

Instrument Mediation

The "Logo Environment" based on both the computatioanl language Logo and in the "Logo philosophy" is a mediational tool for the process of cognitive development. The metaphor of the "turtle", a cybernetic animal living on the computer screen that can be taught through comands expressed in a peculiar language, the "turtle talk", constitutes an element of mediation between the concrete and the abstract, a quasi-concrete objet because its manipulation on the screen is not physical but symbolic, through the written language.

The child "playing Turtle" uses his own experience of displacements in space which are similar to those of the screen turtle. There are a transfer of action meaning from the child-turtle to the screen-turtle. This

transfer is mediated through language: to teach the "turtle" implies making a description to the "other", taking in account the corporal scheme of this "other". The need for an adequate description in terms ofcomputational language offers an opportunity to the learner to confront his ideas with the description in "turtle talk" and its syntax.

Luria (1988) observed, in his syllogism experiment that this form of reasoning is cultural dependent, and the mediational role of written language in its development. The "turtle talk" is embedded in formal logic, offering the possibility of organising the formal thinking in a process similar to that of language acquisition. Language converted to internal speech, organizes the child´s thought, as Vygotsky´s (1978) pointed out when he used language as paradigm of the relationship between learning and development.

In the "Logo Environment" the child relates to the written language, not only through writing instructions to the "turtle", but also as he writes stories. Actually the LOGO activity goes further; to teach the "turtle" requires to name the drawing or the procedure so it can "learn". Naming provides a compromise between flexibility and rigidity: while it allows any name to be chosen, once chosen it acquires the same character of a primitive command; it can only be recalled if written exactly in the same way. In this context, the requirement of a standard, is shown as a necessary condition so "the turtle can learn and understand"; this can then easily be transferred to the orthographic standard.

The fact that the comunication with the computer is made through writing, leads the child to the communicative function of written language, in a way similar to how he learned to speak: in the beginning he writes to obtain an immediate result, he uses the commands that move the "turtle" to obtain a result in the screen. But, the activity of procedure writing changes the psychological functions.

Methodology

The introduction of the Logo language to pre-school children (4 to 6 year old) in a classroom situation has been usually done through the so called "Easy Logo"; this is based on the implicit presumption that it is necessary for the child to dominate the written language first and the notion of number to be able to work with the computer and the Logo language. Another implicit assumption in the use of "Easy Logo", since it makes drawing easy, is an emphasis in the product, a result recognizable by adults, instead of emphasizing the process of contruction of meaning, which is usually slow. The question is whether the inversion of this priority, working with the computer as a mediating tool of the construction of the written language and of the notion of number, may lead earlier to the emergence of these knowledge.

Selecting as unity of analysis the triad child, mediator and computer, we intend to verify how the Logo Environment alter manners of learning the written language and the concept of number.

The research approach is qualitative, based on participant observation., searching to clarify the relationships pointed in the objetives above, through micro-analysis of classroom activities. The documentation is made by video-tape and field notes.

The research is been done in a pre-school class of the public school system of the city of Campinas, where our university is located. The subjects remain in school all day because their mothers work full time. In general they are from low income families with no professional qualifications.

The daily classroom routine includes outdoor playing, two periods of indoor activities, three meals and afternoon nap. The indoor activities are organised around "corners of activities" such as research, reading, arts; two computers with the Logo language is one of these corners. The children choose the corner they want to go at beginning of each period in a groupal activity with the teacher called "roda" (circle).

Through a three year study, begun in 1991, it is expected to observe the effect of the mediation of the Logo enviroment in the development of the higher mental functions, the appropriation of the written language and

the notion of numbers.

Discussion

In the beginning of the school year of 1991, the subjects showed contextualized reading hability (such as Coca-Cola label), but did not recognize even their names in writing. Cards with the students names in capital letters were made, and they were read in a global way. Working with the Logo commands, the teacher recalled the initial if a child or a known word - P of papai (father), F of Fernando. The numerical parameter was presented as how far to move the "turtle"

A preliminary analysis of the collected material shows:

a) An initial phase pressing the keyboard, without differenciation among keys (letters and graphic signs) and often without paying attention to the screen; soon they established a cause/effect relationship when key is pressed/something appears on screen. The second phase was a more systematic exploration trying to give meaning to screen output. This led to several discussions among the children about what was printed on screen. In the third phase, which occurred after one month, the majority of children mastered the cause/effect relationship between the pressed key and what is printed in the screen, inclusive the two representations of letters: upper case in the key and lower case in the screen. As the child identifies the letters as marks of speech, his exploration of the keyboard reveals the significance he is consctruting about writting: he basically abandons the non alphanumeric keys. Thus, even though in a superficial observation it might appear that he is keying erratically, imitating the adult behaviour, the keying reveals the sense he makes of the keys. After this initial exploration, the keyboard function, as a communication media to the "turtle", imposes itself and the interaction child-computers occurs around Logo: the child writes commands in order to make a drawing, attributes meaning to error messages and learns to handle floppy disks (saves and loads files).

b) An extension of the previous phase is attributing the function of "screen cleaners" to keys BS and ENTER. The Logo syntax command-space-paramenter is dominated in the first weeks.

c) The metaphor of "Turtle understands only written orders" (Logo commands) was appropriated by all. "Order the turtle to read" says a child to another pointing to the ENTER key. Two functions are therefore attributed o this key, orders the "turtle to read" and can clean the screen text if hitted several times.

d) The children acquired the notion that in order to make the turtle move it is needed to use numerals. They use digits in groups (ex. FD 789) because the displacement caused by single digits is considered too little. But each digit conserves its individual value: seven, eight and nine and not seven hundred-eighty-nine. The use of Logo commands with numerical parameters leads to construct the notion that using several digits makes the "turtle" walk a lot and with few digits the result is very little.

e) The mediation of the teacher occurs in the zone of proximity development, leading the subject to construct the significance of the Turtle language related to his corporal scheme. The teacher ask what the child wants to do and tells how to teach the "turtle" to do it.

f) By the end of 1991, the children mastered tasks as turning on-off the computer, the use of floppy disks for loading files. The projects made were drawings with very irregulars formats, what reveals the difficulty in using the turning commands.

As they begin to draw with the turtle, they discover the syntactic rules of Logo. The reading of the error messages in the begining is contxtual linked to the action that the child was trying; she points to the sentence saying its content without identifying the words. The majority of the children showed a low resistance to syntactic mistakes: confronted with an error message, the first reaction was to clean the screen without trying to identify the error. Thus, we see that the global significance of the error is understood, but the disconfort prevents a more constructive action. The perception of the error as a mistake in communicating to the turtle

shows up after a few sessions often with the mediation of the teacher of another child: confronted with a message "I have not learned pf50" another child might say: -"It does not understand without a spacing; it needs a spacing!".

BIBLIOGRAPHY

CLEMENTS,D.H., GULLO,D.F.(1984). "Effects of Computer Programming on Young Children's Cognition". J. of Educational Psychology,76, 1051-8.

LAWLER, R.W. (1985). Computer Experience and Cognitive Development. Chichester: Ellis Horwood.

LURIA, A. R.(1986) Pensamento e Linguagem. P.A.: Artes Médicas.

NEWMAN, D., GRIFFIN, P., COLE, M. (1989). The Construction Zone: Working for Cognitive Change in School. Cambridge Univ. Press.

PAPERT, S.(1981). Mindstorms. Brighton: Harvester Press.

PAPERT, S..(1987). "A Critique of Technocentrism in Thinking about the School of the Future". Sofia, Bulgária, May 1987.

RIPPER, A.V. (1985). "O Computador Chega à Escola. Para Que?" in Revista Tecnologia Educacional: ano XII nº 52, maio/junho 83, (pg. 40-43)

RIPPER, A.V.,(1990). "A Logo-Based learning Environment for the Study of Written Language Acquisition". "East/West Invitational Seminar on New Technologies in Education. Leningrado.

SOLOMON, Cynthia. (1987) Computers Environments for Children: A reflection on Theories of Learning and Education. MIT Press.

TURKLE, S. (1984). The Second Self - Computers and the Human Spirit. London: Granada.

UNDERWOOD, J. D.M. and UNDERWOOD, G. (1990). Computers and Learning. Oxford and Cambridge, MA. Basil Blackwell.

VALSINER, J. and van der VEER, R. (1991) "The Encoding Distance: "The Concept of the Zone of Proximal Development" and its Interpretations" in Development and Meaning of Psychological Distance. Hillsdale, NJ: Erlbaum

VYGOTSKY, L.S. (1978). Mind in Society. Cambridge, MA. Harvard University Press.

VYGOTSKY, L.S.(1986) Thought and Language. Cambrigdge, MA. MIT Press.

VYGOTSKY, L.S. LURIA, A.R. and LEONTIEV, A.N. (1988) Linguagem, Desenvolvimento e Aprendizagem. SP: Ícone/EDUSP.

Natural Learning in a Lego-Logo Learning Environment

Jyrki Suomala

Deparment of Teacher Education, University of Jyväskylä
Seminaarinkatu 15 40351 Jyväskylä, Finland
e-mail: suomala@juy.fi

Abstract

The concept of natural learning is a central concept in the conception of learning of Seymour Papert, who developed the Logo programming language. In this paper, Papert´s view of learning is compared with L. S. Vygotsky´s description of the development of the child´s modes of thinking. On the basis of this comparison, an attempt is made to show that natural learning is an important intermediary stage in the process whose final goal is the genuine understanding of scientific concepts.

In the paper, the concept of natural learning is elucidated by means of describing situations in which primary school pupils act in a Lego-Logo learning environment according to the idea of natural learning. The description is based on a teaching experiment with 104 eight-year-old pupils in the municipality of Laukaa, Central Finland, during Spring 1992. The material for the paper was gathered by means of the method of participatory observation during the lessons that were part of the teaching experiment.

Keywords: Lego-Logo, learning environment, learning, thought

Jyrki Suomala *was born in 1959. He is Lecturer of Pedagogics at the Department of Teacher Education, University of Jyväskylä. He received Master of Education (Primary School Teacher, University of Oulu 1983) and Licentiate of Education (University of Jyväskylä) degrees. His research interest is Logo learning environment and its influence on pupils' thinking skills. Doctoral thesis on the subjects is underway at the University of Jyväskylä. He has worked as class teacher in 1983-1988, but for the past five years he has been working at the University of Jyväskylä as Lecturer and Researcher.*

Natural Learning in a Lego-Logo Learning Environment

Seymour Papert´s view on learning

Logo is a programming language developed by Seymour Papert expressly for children [3]. The central concept in Papert´s views on learning is that of **natural learning**. The concept of natural learning refers to a learning process in which an individual acquires knowledge of phenomena in his surroundings without conscious effort and without conscious teaching. Papert aims at showing that people learn an enormous amount of things according to the principle of natural learning [3]. It is typical of such learning that it is intertwined with emotional experiences and is of very personal nature. Although the effects of natural learning are very profound, it is hard to observe them e.g. by means of psychological tests.
Papert elucidates the significance of culture in learning by means of the concept of "**Mathland**" [3,4,5]. In the Mathland, mathematics is learned as easily as the French language is learned in France [3,5]. The Logo environment is one example of a Mathland in which children learn the models of systematic thinking in a natural way .

Papert´s view of learning converges at several points with other approaches that emphasize the significance of culture in the learning process. For instance Cole and Cole take it that the child´s development has as its prerequisite the acquisition of models that are in force within a culture [1]. According to Wartofsky, an individual constructs his own cognitive models of cultural artifacts [10]. Those artifacts include linguistic models, objects, tools, rules of behaviour, and models of thinking. The Logo environment developed by Papert is an instance of an environment in which pupils can learn thinking skills according to the principle of natural learning. In a LEGO/Logo environment the object of programming can be any device built by a child with LEGO components [6,7,8]. In such a situation, Logo serves as a new cultural artifact produced by information society - an artifact by means of which children can learn cognitive models according to the principle of natural learning. Perhaps the most fruitful theory of human development to have dealt with natural learning is that of L. S. Vygotsky.

In the following, an attempt will be made to specify the concept of natural learning by means of Vygotsky´s theory of human development. In his theory, Vygotsky describes the evolution of forms of thinking through the understanding of everyday concepts to the understanding of genuine and scientific concepts. According to Vygotsky´s theory, concepts are cultural entities which the child gradually adopts during his/her development.

The Development of the Basic Forms of Thinking

A word, according to Vygotsky, has always two basic functions: 1. **referential function** and 2. **meaning function** [9]. In Vygotsky´s theory, this is not just a linguistic distinction; instead, the question is of the functional relationship between thought and word, which relationship changes all the time as the child grows up. The development of thinking means the evolution of the meaning function from the understanding of an everyday concept to the use and understanding of the concept as a part of a scientific conceptual system.

When the child first uses words, his/her use of them is closely connected to the interactive situations in which the words serve as tools for mutual understanding. They have a very pragmatic function, but at the same time a lot of rational experimentation and play is involved in their use. The word "mother", for instance, refers to the child´s own mother, some of whose concrete properties the child associates with the word. At this stage the word´s referential function and meaning function coincide.

Vygotsky [8] uses the term **everyday concepts** of words that are adopted in such concrete situations. An everyday concept refers to some concrete object. The child adopts this kind of concepts, whose meaning function and referential function coincide, as separate entities. They are given a diffuse, subjective meaning. Meaning production, thus, is based on some directly observable property of the object [9]. Vygotsky´s central insight was that such a concept is not static, but that its meaning changes as the child´s thinking

evolves; meaning function becomes separated from the original referential function.

By **genuine and scientific concepts** Vygotsky refers to words that have a generalized and hierarchical meaning in the child´s mind [9]. Everyday concepts evolve towards genuine concepts so that their functional relationship with reality changes. The sign stays the same, but the meaning associated with it changes. Meaning is no longer connected to immediate perception, and the child becomes able to incorporate the word into a conceptual system, thus making it a genuine tool [9]. The child understands that the meaning of the word "mother" is not in his/her subjective use of it, but that the word has a wider conceptual meaning.

Vygotsky showed that the child does not learn the word´s genuine-conceptual meaning until about at the age of twelve [9]. This stage corresponds to Piaget´s period of formal thinking [2].

Everyday concepts, thus, develop in natural interactional situations from concrete reference towards genuine conceptual meaning, "bottom up", while scientific concepts develop "top down". The conscious control of phenomena that is characteristic of genuine and scientific concepts presupposes that the phenomena are related to each other as parts of a general conceptual system [9].

Thinking evolves, thus, in two dimensions. On the one hand, it goes from the unconscious to the conscious, on the other hand its sphere extends from the concrete to the conceptual. The thinking process never emerges as separate of the child´s prior experiences [9].

Natural learning, as described by Papert, resembles the acquisition of everyday concepts as described by Vygotsky. It is only gradually that the concept that emerged as a product of natural learning in a Logo environment can evolve into a concept that the child can use consciously as part of a conceptual system and as a tool for planning. In the case of Logo, that conceptual system consists of the hierarchical programme structure with its subprogrammes. In view of learning, it is necessary that the child can play with concepts and use them in their own ways.

The central problem with school teaching has traditionally been to link scientific concepts to pupils´ prior experiences. At school, it is only seldom that pupils can experiment and play with scientific concepts. The following chapter focuses on situations in a LEGO/Logo learning environment in which the principle of natural learning is at work.

Instances of the occurrence of natural learning in a LEGO/Logo learning environment

General

The following examples were gathered in connection of a teaching experiment carried out in the municipality of Laukaa, Central Finland, during Spring 1992. A total of 104 fifth-grade pupils from two primary schools participated in the experiment. The pupils were divided into two groups, each of which was given 20 hours of teaching[1]. The pupils worked on LEGO/Logo in pairs.

During the experiment, the pupils carried out both the merry-go-round project and the robot project. In both projects, the pupils were taught the Logo commands controlling the engines, the lights, and music. In addition, the "repeat" command and the writing of the program were introduced. The following five examples are taken from the field diary in which the descriptions of the instances of natural learning were recorded immediately after the lessons.

Adding lights

The merry-go-round project was designed so as to include the possibility of adding new lights to the device. When one light had been installed, one of the pupils came to ask if more lights could be added. Permission

granted, all the pupils started adding lights of different colours. They found it very important to have a lot of lights of different colours in their merry-go-rounds.

Installing extra lights seemed to have a positive effect in that it made the programmes written by the pupils more complicated. The pupils were faced with the demanding task of making the lights work at the right moment. And when pupils built a fine light system, they naturally also wanted to test if it worked. All this sets their thinking skills on trial.

The tournament

The pupils had just studied LEGO/Logo for 12 hours. They had completed the LEGO robot and connected it to the computer. They were now supposed to use their skills learned during the merry-go-round project for moving the robot. This time the task was more difficult, however, because the robot had two engines instead of one.

Two of the pairs of pupils started to test whose robot would be stronger. The robots were placed so that they faced each other, and the engines were started. Interested spectators gathered around the table. Finally one of the robots won. After this, the rest of the pupils wanted to try the same. They started building their robots for that purpose.

The pupils started discussing eagerly how the robot should be tuned to be make it stronger. Some of the students made the bumper longer, some set long sticks in front of the robot. To increase friction, one of the students decided to put more weight on the robot.

After taking up a new idea, it was tested in a tournament. In case of defeat, the pupils who lost thought of what was better in the other robot, and tried to build one´s their own robot accordingly. The idea spread at once through the group, and everybody wanted to test it.

The tournament was such an exciting and inspiring experience for the pupils that they wanted to take it up during the next lesson as well. The other groups heard about the tournament, and it was then arranged in their lessons, too. In the tournaments, pupils obtained experience of such basic concepts of physics as friction, power, and velocity. The concepts were given a meaning through practical operations, not as part of a system of physical concepts. Such an experience is a good starting point of later studies in physics.

Jerky movement with the command REPEAT

The pupils were taught, as a new topic, the "REPEAT" command. Now they were to write a programme applying the new command. Two boys wrote the following programme (Fig.1):

COMMANDS	**EXPLANATIONS**
TO PETER	Name of the programme
SETPOWER 7	Full speed
ONFOR 100	Rotation for 10 sec.
RD	Change of direction
ONFOR 110	Rotation for 11 sec.
SETPOWER 5	Decrease speed
REPEAT 55[ONFOR 1 RD]	Rotation for 0,1 sec and change direction 55 times
SETPOWER 7	Full speed
ONFOR 107	Rotation for 10,7 sec.
END	End of programme

Figure 1. Programme "Peter" written by two boys

When the boys ran the programme, it aroused amazement and joy in the group. The "REPEAT" command

made the merry-go-round rotate in a jerky fashion. When seeing this, the other pupils asked how it was done, and the boys told them how they had done it. The others soon started to apply the command with the intention to bring about the jerky movement.

This incident shows the kind of activities that occur in a LEGO/Logo learning environment. The main principle of the action there is not to achieve a specific goal but to encourage pupils to exploit their own ideas and experiences. In view of programming, the pupil´s goal may be quite peripheric. However, such action gives him/her new experiences and is significant for later learning. At first the commands are used as everyday concepts, but later they turn into genuine and scientific concepts.

Connotations of the programme name

When carrying out the merry-go-round project, the pupils to went on practice linking subprogrammes into the entire programme structure. For an eleven-year-old pupil, this process is still difficult to grasp. Although the pupils learned the principle of the subprogramme, they did not use subprogrammes very much on their own initiative.

The following example describes how two pupils killed two birds with one stone. The boys´ intention was to test the "REPEAT" command and use it to bring about a jerky movement. However, they also wanted to use a sentence of their own invention to control the merry-go-round. Since more than one words can not appear in the programme name, the boys went to the command center and wrote the words LITTLE BOY WENT on the same command line. They wrote three subprogrammes and gave them the names "BOY", "WENT", and "LITTLE", so that the programme could be launched from the command center with the sentence "LITTLE BOY WENT".

The programme worked, and the boys managed to carry out their own idea in the LEGO/Logo learning environment. Naming a programme seems to be a very personal process. The pupils give their programmes their own names or those of their friends or idols. The name does not need to have anything to do with the contents of the programme. However, by giving the programmes the names they want, the pupils retain a personal touch with the programming tasks.

Systematic experimentation

Logo programming gives opportunities to many kinds of activities. A popular way to write a programme seems to be to test a specific command systematically in a certain order. The following example illustrates this tendency in more detail.

Two girls were testing the "TONE" command by making the following programme (Fig.2):

COMMANDS	EXPLANATIONS
TO MUSIC	Name of the programme
TONE 262 1	Note C, duration 0,1 sec.
TONE 294 2	Note D, duration 0,2 sec.
TONE 330 3	Note E, duration 0,3 sec.
TONE 349 4	Note F, duration 0,4 sec.
TONE 392 5	Note G, duration 0,5 sec.
TONE 440 6	Note A, duration 0,6 sec.
TONE 494 7	Note H, duration 0,7 sec.
TONE 523 8	Note C, duration 0,8 sec.
TONE 494 7	Note H, duration 0,7 sec.
TONE 440 6	Note A, duration 0,6 sec.
TONE 392 5	Note G, duration 0,5 sec.
TONE 349 4	Note F, duration 0,4 sec.
TONE 330 3	Note E, duration 0,3 sec.
TONE 294 2	Note D, duration 0,2 sec.

Figure 2. Programme "Music" written by two girls

Increasing the duration of the notes is a typical instance of systematic experimentation that follows the principle of "add one". The same kind of systematic experimentation occurred in connection with the commands "SETPOWER" and "ONFOR" as well.

Conclusions

This paper has focused on natural learning. Papert has designed the Logo programming language so as to enable pupils to learn systematic thinking naturally by means of it. Vygotsky´s description of the development of everyday concepts towards genuine and scientific concepts seems to support Papert´s idea concerning natural learning. Teaching situations should, however, be arranged so that pupils could get acquainted with new concepts by giving them personal meanings. It is only after this stage that they can be expected to use concepts consciously as tools for planning and as part of a conceptual framework. In a LEGO/Logo learning environment, natural learning means that pupils may, in projects, use Logo commands to control objects they themselves find significant.

In the situations described in this paper, the pupils built the LEGO device according to their own purposes. These purposes were very personal: winning a tournament, or building light systems, or increasing the rotation speed of the merry-go-round. It was especially important for the pupils to make the merry-go-round to jerk. They also found it important to give the programme an amusing name or to try different values with the commands.

These are instances of natural learning. In this kind of learning, a Logo command is learned as an everyday concept in which referential function and meaning function coincide. This means that the meaning of a Logo command is in the action that is produced in using its concept. It is only later that Logo concepts acquire, in pupils´ minds, a genuine or scientific meaning as part of the hierarchic system of all Logo commands.

Note

[1]The development of genuine and scientific concepts of the pupil´s participating in the teaching experiment has also been studied after the teaching period, in an open problem solving situation. The pupils´ problem solving situation was video taped. This material, however, is not dealt with in this paper.

REFERENCES

[1] Cole, M. & Cole, S.R. 1989. The Development of Children. New York: Scientific American Books.

[2] Inhelder, B. & Piaget, J. 1958. The Growth of Logical Thinking from Childhood to Adolence. New York: Basic Books.

[3] Papert, S. 1980. Mind-Storms. New York: Basic Books.

[4] Papert, S. 1986. Are Computers Bad for Children? In Campbell & Fein (eds.) Young Children and Microcomputers. New Jersey: A Reston Book, 172 - 180.

[5] Papert, S. 1987. Computer Criticism vs. Technocentric Thinking. Educational Researcher 16 (1), 22-30.

[6] Resnick, M. & Ocko, S. 1991. LEGO/Logo: Learning Through and About Design. In I. Harel & S. Papert (eds.) Constructionism. New Jersey: Ablex Publishing Corporation Norwood, 141 - 150.

[7] Resnick, M., Ocko, S. & Papert, S. 1988. Lego, Logo, and Design. Children´s Environments Quartely, 5 (4), 14-18.

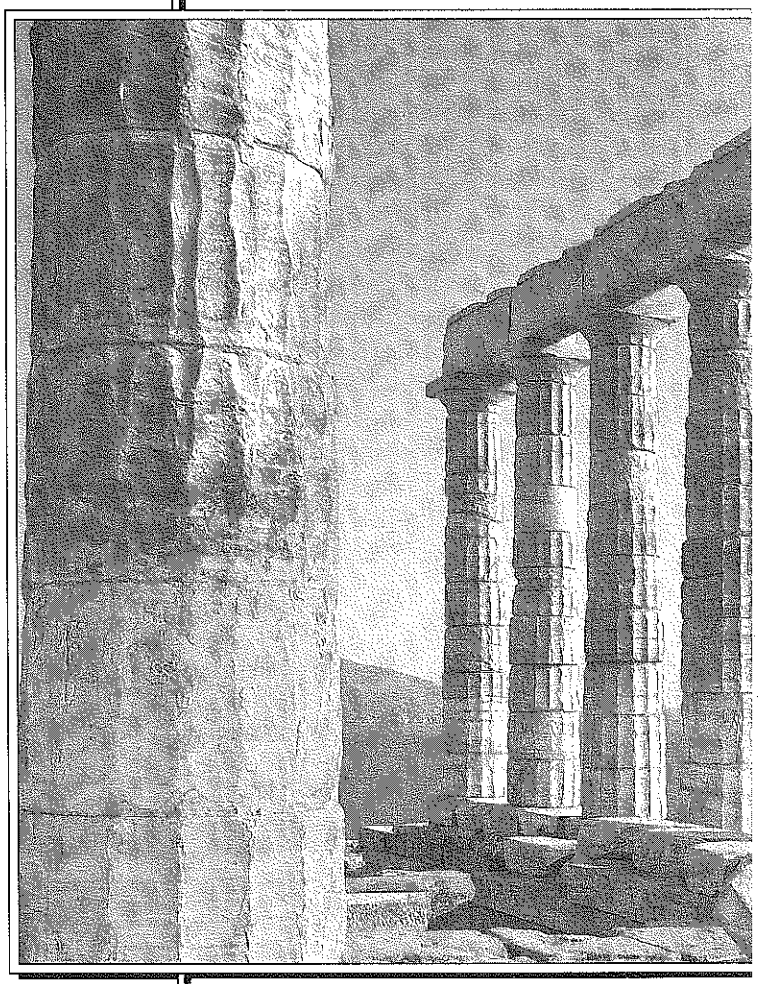
[8] Rosen, M. 1988. Lego meets Logo. Classroom Computer Learning, April, 50 - 58.

[9] Vygotsky, L.I. 1979. Thought and Language. Cambridge: The M.I.T. Press.

[10] Wartofsky, M.W. 1979. Models; Representation and the Scientific Understanding. Dordrecht: D.Reidel Publishing Company.

Theme B

Domain Specific Learning



B. Domain Specific Learning

77 • *Beguin C., Vitale B., De Marcellus O., Tryphon A.*
Experimentation et Programmation pour la Construction de l' Espace Sonore

85 • *Calabrese E., Chiavarini G., Rossi A.*
An Introduction to Database with Logo

97 • *Dapontes N.*
La Tortue, le Formalisme Mathematique, et les Connaissances Intuitives ...

105 • *Futschek G., Schauer H.*
Exploring Recursion: from Random Walks to the Design of Mazes

113 • *Lapidot T., Rachman R.*
MicroBridge: a Combinatorical Microworld

119 • *Martins M. C.*
Analysis of the Musical Domain in the Logo Environment

127 • *Minopoulos C., Michaelides T., Patrineli A.*
Discovering Angles with Logo

137 • *Shafee Give'on Y.*
Catch Infinity by its Tail: Infinite Sequences in Logo, Revisited

145 • *Turcsanyi-Szabo M.*
From Square to a Thousand Cranes

153 • *Vieira da Rocha H.*
The Use of Computational Representations in the Teaching and Learning ...

Experimentation et Programmation en Logowriter pour la Construction de l' Espace Sonore

Claude Beguin[1], Bruno Vitale[2], Olivier De Marcellus[2], Anastasia Tryphon[3]

[1] College du Cycle d'Orientation de la Gradelle de Geneve et DGCO

[2] Centre de Recherches Prycho-Pedagogiques du Cycle d'Orientation de Genève

[3] Fonds National Suisse pour la Recherche Scientifique (F.N.R.S.)

Abstract

The experiment presented here deals with 12 to 13-year-old children's production of sounds using vibrating strings. This experiment was extended by the pupils' programming in LOGOwriter on a computer. Programming allowed them to generalise easily the first partial models they had established in terms of simple ratios (octave and dominant relations) between the length of two strings.

Keywords: Logo, programming, simulation, sounds, vibrating strings

Claude Béguin *est biologiste. Il a fait une thèse en biologie moléculaire à l'université de Genéve. Après des recherches en bactériologie et physiologie cellulaire, il est entré comme enseignant au cycle d'orientation (CO) du département de l'instruction publique du canton de Genève. Il y anime présentement l'enseignement de l'informatique et de la didactique des sciences expérimentales.*

Bruno Vitale *est physicien théoricien, en congé sabbatique de l'université de Naples. Il travaille au laboratoire de didactique et d'epistémologie des sciences (LDES) de l'université de Genéve, et au centre de recherche psychopédagogiques (CRPP) du CO.*

Olivier de Marcellus *est psychologue au CRPP.*

Anastasia Tryphon*, psychologue, est conservatrice des archives Jean Piaget, à Genéve.*

Experimentation et Programmation en Logowriter pour la Construction de l' Espace Sonore

1. L'initiation à l'informatique au 7ème degré par la programmation en LOGOwriter

Une initiation à l'informatique par la programmation en LOGOwriter est donnée dans tous les collèges du cycle d'orientation du canton de Genève, à tous les élèves du 7ème degré (12-13 ans).

L'objectif de cet enseignement est de familiariser les élèves avec des systèmes formels utilisés ultérieurement dans d'autres disciplines, en même temps que sont exercées des commandes et des fonctions communes à tous les logiciels que l'élève sera ultérieurement appelé à utiliser. Il a aussi des objectifs trans-disciplinaires, et devrait donc aussi permettre à l'élève de pouvoir s'aider, grâce aux éléments acquis de la programmation en LOGOwriter, à structurer sa pensée dans divers domaines relevant de la logique formelle, et à aborder certains problèmes de nature algorithmique, numérique, géométrique, musicale ou linguistique.

Cette initiation est conduite grâce à un ensemble d'activités du type 'résolution de problèmes', regroupées sous le titre de séquences didactiques en LOGOwriter (1).

2. LOGOwriter: une application programmable par les élèves

Plusieurs des activités proposées pour l'initiation (1) sont déjà orientées vers l'interdisciplinarité à travers des problèmes relevant de divers champs d'application.

Des tentatives sont par ailleurs en cours pour permettre à l'élève d'intégrer ultérieurement les compétences qu'il a acquises en LOGOwriter pour s'aider dans différents apprentissages scolaires, notamment les mathématiques et la géométrie (2).

Le travail présenté ici relève d'une tentative d'intégration de l'informatique aux pratiques pédagogiques à la fois en musique et en sciences expérimentales.

3. L'initiation à la pratique expérimentale au 7ème degré

Le Cycle d'Orientation de Genève dispense, aux mêmes élèves du 7ème degré, une pratique pédagogique originale sous la forme d'un cours dit d'observation scientifique.

Il s'agit de leçons de stratégies et techniques de base en sciences expérimentales (telles que prise de mesures, organisation d'une expérience, évaluation des erreurs, interprétation et communication des résultats).

L'exercice des conduites liées à la mesure, en tant qu'estimation numérique de l'effet produit par la variation d'un paramètre, y est prioritaire, pour placer l'expérimentation dans un cadre interprétatif.

Au cours de l'année scolaire 1992-93, le premier auteur de cette communication a enseigné dans la même classe les cours d'initiation à l'informatique et d'observation scientifique. Ce contexte a grandement facilité l'expérimentation relatée ci-dessous.

4. Les cordes vibrantes et la construction de gammes

La longueur d'onde du son fondamental produit par une corde vibrante est reliée à différents facteurs; elle est en particulier proportionnelle à la longueur de la corde, et inversement proportionnelle à la racine carrée de la tension.

On définit comme 'gamme des physiciens' celle qui correspond aux critères essentiellement perceptifs suivants:

- la pertinence du rapport d'octave (universellement défini par un rapport des longueurs de 1 : 2 de la corde);

- la consonnance de sons produits par des cordes dont les longueurs sont dans des rapports définis et simples entre elles.

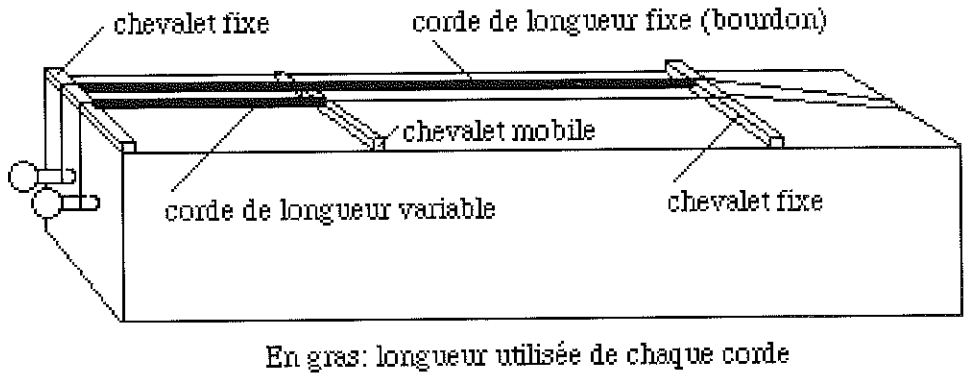


figure 1: le bicorde utilisé pour produire des sons consonnants

Un bicorde (figure 1) pour étudier la production de son par la vibration de cordes pincées a été construit par le premier auteur en collaboration avec André.M. Huwyler, luthier à Genève.

Il est constitué de 2 cordes de violoncelle tendues sur une table de résonnance à l'aide d'une cheville et d'un cordier. Une longueur de 75 cm est délimitée sur les 2 cordes par 2 chevalets fixes. Un 3ème chevalet, mobile, en contact avec une seule des 2 cordes, permet de varier aisément sa longueur en laissant inchangée celle de l'autre corde. Les 2 cordes sont pincées à l'aide d'un plectre.

En utilisant ce dispositif, nous avons expérimenté avec 10 élèves d'une classe d'observation scientifique, par groupe de 2, dans un local séparé, avec 2 maîtres observateurs. Parmi les consonnances reconnues, les élèves sont parvenus à modéliser en termes de rapports simples:

- les premiers rapports d'octave correspondant aux longueurs L, L/2 et L/4;

- les premiers rapports de quinte correspondant aux longueurs L, 2 * L/3 et L/3.

Aucun élève n'a défini spontanément des relations de consonnance correspondant à d'autres rapports de longueur des cordes. Nous avons alors envisagé que la programmation par l'élève en LOGOwriter d'un simulateur dans lequel il pourrait introduire ses modèles partiel, lui permettrait peut-être de les généraliser.

5. Du dispositif expérimental à l'ordinateur

Nous avons alors conçu avec les élèves, en LOGOwriter, le programme d'un 'synthétiseur' de son qui simule le bicorde utilisé au laboratoire.

Le programme-type qu'ils ont réalisé est constitué des procédures suivantes:

```
pour CORDE1
lc fpos [-150 5] bc
fpos [150 5]
fin

pour CORDE2
lc fpos [-150 -5] bc
fpos [150 -5]
fin

pour CHEVALET
lc fpos [-150 0]
fcap 90
fin

pour AVANCE_LE_CHEVALET :l
fpos [-150 0] av :l * 4
fin


pour ESSAI :l
AVANCE_LE_CHEVALET :l
NOTE 75
NOTE :l
fin

pour DEPART
vg vt vc CORDE1 CORDE2
fforme 77
CHEVALET
donneoutils "BOUTILS
fin
```

La procédure `NOTE :longueur` a été préfabriquée à partir de la primitive `SON :fréquence :durée` dans une boîte à outils, `BOUTILS`, qui se réfère directement à une notion construite au cours de l'expérimentation, et permet d'éviter d'aborder, avec ces jeunes élèves, la notion de fréquence. `NOTE 75`, par ailleurs, est accordée à la note produite par la corde de longueur fixe (75 cm) du bicorde.

```
pour NOTE :l
son 19400 / :l 20
fin
```

6. La transposition d'une situation expérimentale dans un programme en LOGOwriter et la généralisation de modèles partiels

Chaque paire d'élèves expérimentateurs utilise le 'synthétiseur' pour prolonger l'expérimentation commencée avec le bicorde. Les maîtres observateurs leur rappellent les expériences déjà faites et les modèles énoncés. Les élèves disposent à la fois du bicorde et de l'ordinateur, installés côte-à-côte. Ils peuvent ainsi recréer les conditions de leur expérience préalable et les transposer dans la procédure ESSAI :longueur de leur programme en LOGOwriter.

L'intérêt de la transposition de l'expérience en LOGOwriter est double D'une part, la consonnance avec la note de base des membres, après le 3ème d'une famille de notes, ne peut être reconnue, à cause de leur timbre 'métallique', sur le bicorde; ceux qui sont émis par le générateur de sons de l'ordinateur sont parfaitement audibles jusqu'au 5ème, voir 6ème membre. L'intérêt fondamental, cependant, réside dans la possibilité de donner l'ordre de déplacement du chevalet et d'émission du son, en attribuant à la variable :longueur de ESSAI l'un des rapports identifiés de consonnance. En d'autres termes, l'élève peut introduire dans le simulateur, non pas seulement une valeur (longueur de la corde), mais une opération modélisant une consonnance. Cette situation représente un riche potentiel de développement; en modifiant les valeurs numériques du modèle partiel initial, l'élève va créer des situations sonores nouvelles et être incité à le généraliser en un modèle plus global. Nous avons appelé ce processus l'induction, ou entraînement algorithmique, à la généralisation d'un modèle.

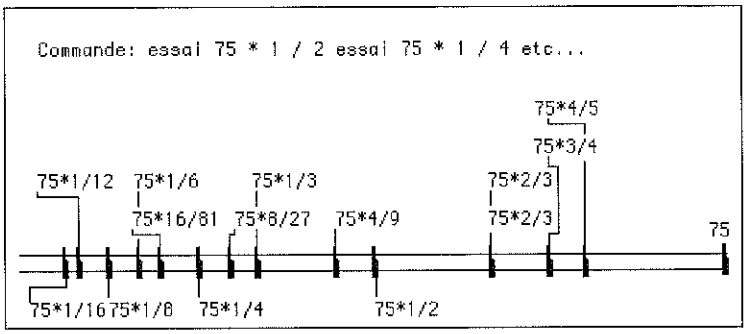


Figure 2: image d'écran du simulateur en LOGOwriter du bicorde avec les essais successifs reconstitués (de bas en haut) d'un groupe d'élèves.

La figure 2 résume les rapports essayés avec le simulateur, et reconnus consonnants, par l'un des groupes d'élèves expérimentateurs, à partir des rapports d'octave (L, L/2 et L/4) et de quinte (L, L*2/3 et L/3) préalablement identifiés. Dans l'ordre, ont été exécutés:

- la série des octaves de la note de base prolongée jusqu'à L/16;

- une 'spirale des quintes' (L*2/3, L*4/9, L*8/27, L*16/81);

- une seconde série d'octaves à partir de la première quinte (L*2/3, L*1/3, L*1/6);

- la première quarte (L*3/4) et la première tierce (L*4/5).

7. La génération algorithmnique de gammes et de mélodies

Au terme de leur expérimentation, les deux élèves du groupe dont les résultats ont été présentés ci-dessus ont chacun créé une petite mélodie en utilisant les notes qu'ils avaient définies:

```
pour TAREQ
repete 4 [essai 75 essai 75 * 4 / 5 essai 75 * 2 / 3 essai 75 / 2]
fin

pour THUY :l
si :l < 4 [ecris "fini stop]
note :l
note :l * 4 / 5
note :l * 2 / 3
THUY :l / 2
fin
```

L'une des prochaines étapes de notre travail consistera à structurer cette démarche à travers une étape de construction par les élèves de gammes simples, par tri selon leur acuité des notes créées à l'intérieur d'un intervalle d'octave.

La programmation en LOGOwriter se prête fort bien à la création algorithmique de gammes engendrées par des sons consonnants. Bruno VITALE (3) l'a utilisée pour en définir différents types.

La spirale des quintes, en particulier, est une suite de sons consonnants dont chacun est la première quinte du précédent. Il peut engendrer une gamme en appliquant l'algorithme suivant: si une note sort du premier intervalle d'octave de la note de base, elle est remplacée par son octave inférieure, soit une note correspondant à une longueur double de la corde. Cet algorithme est transcrit dans la procédure SPIRALE (voir annexe). La suite doit alors être triée vers le bas, relativement aux longueurs de la corde, ce que réalise la procédure TRIER_VB :gamme :gamme2 (4), avec ELIMINE et PLUS_PETIT.

La figure 3 montre les intervalles créés en prenant les 12 premières notes engendrées par la spirale.

8. Conclusions

Notre travail a montré qu'une programmation rudimentaire par les élèves eux-mêmes en LOGOwriter peut être un outil très efficace pour leur faciliter la généralisation d'un modèle partiel. La figure 4 illustre les relations entre les phases expérimentale et simulée du processus d'organisation par l'élève de son espace sonore.

Une programmation plus avancée en LOGOwriter permettrait, par ailleurs, de montrer à des élèves plus âgés la multiplicité des facteurs de leur expérience musicale et de les rendre

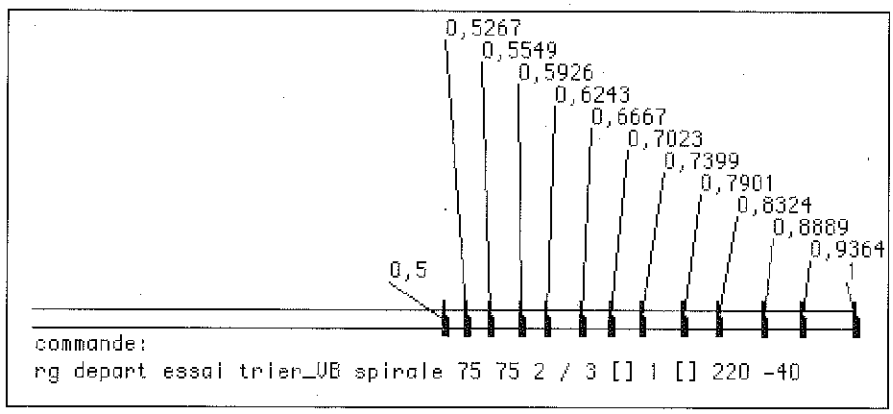


Figure 3: image d'écran des longueurs de corde correspondant aux 12 premières notes engendrées par la spirale des quintes

attentifs au fait que certains de ces facteurs (en particulier, la gamme choisie, c'est-à-dire, les 'notes légalement acceptées' dans un certain espace musical) sont conventionnels, contrairement à ce que leur éducation musicale (études du piano, du solfège, etc.) pourrait leur laisser croire avec son cadre rigide (qui défini une fois pour toute 'ce que la musique est et doit être').

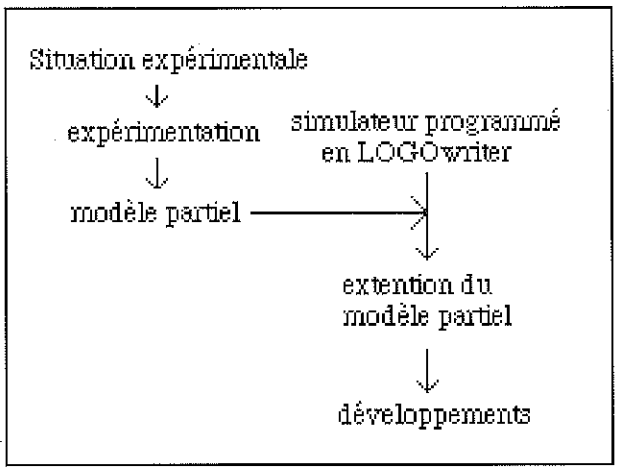


Figure 4: la relation expérimentation - programmation en LOGOwriter

9. Annexe

Procédures pour engendrer une gamme à partir des 12 premières notes de la spirale des quintes. Leur utilisation requiert les procédures `DEPART`, `CORDE1`, `CORDE2`, `CHEVALET` et `AVANCE_LE_CHEVALET :l` présentées ci-dessus.

```
pour SPIRALE :l° :l :r :gamme :compte
si :compte > 12 [ra :gamme]
si :l < :l° / 2 [ra spirale :l° :l * 2 :r :gamme :compte]
ra  SPIRALE :l° :l * :r :r md :l :gamme :compte + 1
fin

pour TRIER_VB :gamme :gamme2
si vide? :gamme [ra :gamme2]
ra TRIER_VB elimine :gamme mp plus_petit :gamme :gamme2
fin


pour ELIMINE :gamme
si vide? :gamme [ra :gamme]
si egal? dernier :gamme PLUS_PETIT :gamme [ra sd :gamme]
ra phrase dernier :gamme ELIMINE sd :gamme
fin


pour PLUS_PETIT :gamme
si vide? :gamme [ra "]
si egal? compte :gamme 1 [ra dernier :gamme]
sisinon (premier :gamme) < (dernier :gamme) [ra PLUS_PETIT sd :gamme][ra PLUS_PETIT
sp :gamme]
fin


pour ESSAI :l :x :y
si vide? :l [stoptout]
AVANCE_LE_CHEVALET premier :l
bc estampe
NOTE premier :l
```

```
fpos liste :x :y
etiquette (premier :l) / 75 lc fpos [-225 -60] fcap 90
ESSAI sp :l :x - 20 :y + 12
fin
```

Note
Cette recherche a été partiellement financée par le contrat N° 11-30259.90 du Fonds National Suisse pour la Recherche Scientifique

BIBLIOGRAPHIE

[1] BEGUIN, Claude, Edmée DUNAND, Jean-Rodolphe FURER, Claude GRUNER et Olivier de MARCELLUS. *Séquences didactiques pour la 7ème année avec LOGOwriter*, Département de l'instruction publique, Genève 1992.

[2] DREYFUSS, Pierre-André, Francis CHAMBORDON et Claude BEGUIN. *Possibilités d'application de la programmation en LOGOwriter dans les activités scolaires* (en préparation).

[3] VITALE, Bruno. *Intégration de l'informatique à la pratique pédagogique*, volume 2, cahier 4, publication du Centre de Recherches Psychopédagogiques du Cycle d'Orientation de Genève (CRPP), 1992.

[4] MATHIL, Jacques. Communication personnelle.

An Introduction to Database with Logo

Eduardo Calabrese[1], Gabriella Chiavarini[2], Angelo Rossi[2]

[1] Dip.to di Ingegneria dell'Informazione
Università di Parma
43100 PARMA, Italy
e-mail: Calabrase@prvxin.cedi.unipr.it

[2] A.S.I.
B.go Tommasini, 8
43100 PARMA, Italy

Abstract

A Logo program to manage a small database consisting of only one table is presented. The table is kept and updated on disk, but a copy of it is kept in memory using property lists in order to speed up queries. Workspace memory is therefore the only limitation to the number of records in the table and to the number of fields in each record. The program can be used to manage small collections of items, such as classroom libraries or data about a class. This program evolved from one presented in an advanced teacher training course in Parma. The goal is to present Logo as a general purpose programming language, not just turtle graphics.

Keywords: Logo, database, teacher training

Eduardo Calabrese *is a physics Ph.D. who switched to computer science. He is now teaching an introductory course in informatics for engineering students at the University of Parma. Since 1983 he has been involved in experimenting with the use of computers in primary school and in teacher training. Logo is the major topic of this activity, along with other educational uses of computers. Presently he is chairman of ASI (Associazione Scuola e Informatica), which is a group of teachers involved in the use of informatics in education.*

Gabriella Chiavarini *is a teacher of mathematical and natural sciences in secondary school. Since 1982 she's using informatics in education through the use of Logo. She's a committee member of the ASI (Associazione Scuola e Informatica).*

Angelo Rossi *is a teacher of technological sciences in the secondary school. He is presently working on the application of informatics and new technologies, mainly through the use of Logo. He's a committee member of ASI (Associazione Scuola e Informatica) and is also vice-president of Parma's school district.*

An Introduction to Database with Logo

Preface

The program shown in this paper, written in IBM-Logo, is one of a series of programs presented and analyzed by E. Calabrese in an advanced Logo course for secondary school teachers in 1992. This course was focused on the deepening of knowledge of Logo as advanced programming language showing mainly non-graphic applications. We believe that knowledge of a programming language and of the principles of informatics should nowadays belong to the cultural background of any teacher, and in particular of those in a scientific or technical field.

Logo is a good language for learning such topics as control structures, procedural abstraction, data abstraction, recursion, local and non-local environments, etc. Besides, a good knowledge of Logo as a general purpose language lets teachers appreciate it more even if they are just using it to do turtle graphics. This feeling of appreciation can be transmitted to the pupils, who must feel that they are learning to 'program', not just playing another computer game.

The program presented here can be used to manage a small database, such as data on the pupils of a classroom, a classroom library, a collection of photographs, etc.

Introduction

The concept of DataBase (DB) can be explained as a collection of items [1] that can be organized on different tables and that can be managed by a program called a DataBase Manager. These tables are organized in lines (records) and columns (fields) and the information there contained can be updated, modified, or deleted. It is also possible to carry on queries satisfying particular criteria

In this paper we are showing how Logo can be used in order to organize and manage a small DB. We will simply consider a DB consisting of one single table and we will suppose that this table contains some data about a class of pupils (see Table 1).

Table 1: Some data on the pupils of a class

	FAMILY	NAME	SEX	AGE	HEIGHT	WEIGHT
1	ROSSI	MARIO	M	12	158	54
2	BIANCHI	GIORGIO	M	12	161	57
3	VERDI	MARIAROSA	F	11	152	48
4	NERI	FRANCESCA	F	13	160	49
5	BRUNI	CARLO	M	12	165	63
:	:	:	:	:	:	:

As you can see from the table, each pupil corresponds to a record and the information about each single pupil is divided into fields. Each record is identified by a line number and each field is identified by a field name.

Since we don't want to loose the data between executions, they must reside in a file. In order to easily handle this file with a Logo program, it could be organized as follows:

```
6                       number of fields

FAMILY
NAME
SEX                     field names
AGE
HEIGHT
WEIGHT

"ROSSI
"MARIO
"M                      Record no. 1
12
158
54

"BIANCHI
"GIORGIO
"M                      Record no. 2
12
161
57
:
```

(The empty lines above divide the different blocks and are not part of the file.)

The first line contains the number of fields (`:NFIELDS`). The following `:NFIELDS` lines contain the names of the fields. These are followed by an arbitrary number of groups of `:NFIELDS` lines; each group corresponds to a record and contains the information about that single pupil. If `:NRECORDS` is the number of records in the table, the number of lines in the file is `1+(:NRECORDS+1)*:NFIELDS`.

In order to consult our table more easily it should be more convenient to have all data in memory. In general, this is not possible since most of the time files are notably larger than the memory capacity of a PC. Considering the exclusively didactical objectives of our problem, we will therefore consider a file small enough to be able to put all data in memory. First of all, our program will have to read the file and transfer the contents in memory. Of course, when data are modified or new records are added, it is wise to immediately bring the file on disk up to date in order to avoid loss of information.

We will use "Property Lists" (PL) for the organisation of data in memory [2]. We'll build a PL for each line of table 1, calling it `RECORD`*n*, where *n* is the line number. For example, the first line of the table will result in the following PL called `RECORD1`:

RECORD1	
Property	Value
FAMILY	"ROSSI
NAME	"MARIO
SEX	"M
HEIGHT	158
WEIGHT	54

Loading the table

If global variable "FILENAME contains our file's name, the following procedure READTABLE reads the file and creates a PL for each record.

```
TO READTABLE
MAKE "NRECORDS 0
OPEN :FILENAME
SETREAD :FILENAME
IF NOT READEOFP
      [MAKE "NFIELDS FIRST READLIST
      FIELDNAMES :NFIELDS
      READRECORDS]
SETREAD []
CLOSE :FILENAME
END
```

The global variables "NRECORDS and "NFIELDS will contain respectively the number of records read from the file and the number of fields in each record (6 in our example). The procedure READTABLE, after initializing :NRECORDS to 0 and :NFIELDS from the first line of the file, calls the procedures FIELDNAMES and READRECORDS.

```
TO FIELDNAMES :NFIELD
IF :NFIELD = 0 [STOP]
FIELDNAMES :NFIELD - 1
MAKE WORD "FIELD :NFIELD FIRST READLIST
END
```

Calling FIELDNAMES 6 produces the same effect as defining the following six global variables:

```
MAKE "FIELD1 "FAMILY
MAKE "FIELD2 "NAME
MAKE "FIELD3 "SEX
MAKE "FIELD4 "AGE
MAKE "FIELD5 "HEIGHT
MAKE "FIELD6 "WEIGHT
```

The recursive call before the MAKE instruction allows us to reverse the numbering of the fields: from 1 to 6 instead of from 6 to 1.

```
TO READRECORDS
LOCAL "RECORD
IF READEOFP [STOP]
MAKE "NRECORDS :NRECORDS + 1
MAKE "RECORD WORD "RECORD :NRECORDS
READRECORD :NFIELDS
IF EQUALP "& FIRST FIRST GPROP :RECORD :FIELD1
  [REMOVE :RECORD MAKE "NRECORDS :NRECORDS - 1]
READRECORDS
END
```

Procedure READRECORDS keeps reading records until it reaches the end of the file. The reading of each record is carried on by the procedure READRECORD, that will read a line for each field for a total of :NFIELDS lines.

```
TO READRECORD :NFIELD
IF :NFIELD = 0 [STOP]
READRECORD :NFIELD - 1
PPROP :RECORD THING WORD "FIELD :NFIELD READLIST
END
```

The PPROP instruction will result more clear if we evaluate its inputs in the situation in which we are reading the first field of the first record (:RECORD = "RECORD1, :NFIELD = 1, READLIST -> ["ROSSI]). The resulting instruction

```
PPROP "RECORD1 "FAMILY ["ROSSI]
```

will assign the value ["ROSSI] to the property "FAMILY of the PL "RECORD1. The IF EQUALP "& ... instruction in READRECORDS will allow us to ignore a record that will contain the symbol & as the first character of the first field. The use of this symbol will let us logically delete a record, as we will show later in the procedure DELETE. GPROP :RECORD :FIELD1 yields the contents (a list) of the first field of the given record. Applying the operations FIRST FIRST on this list will result in the first character of the first word of the list. If this character is &, the procedure REMOVE will cancel the corresponding PL and the total number of records is diminished by one.

```
TO REMOVE :RECORD
REMOVEPL :NFIELDS
END

TO REMOVEPL :NFIELD
IF :NFIELD = 0 [STOP]
REMPROP :RECORD THING WORD "FIELD :NFIELD
REMOVEPL :NFIELD - 1
END
```

The REMOVEPL procedure, called by REMOVE, deletes the PL :RECORD by removing one by one all values with the use of the REMPROP primitive.

Main procedure

As we have mentioned before, the main functions performed by a DB are:

- the insertion of a new record
- the removal of an existing record
- the search of data satisfying particular criteria

All these options are present in the main procedure DBM. We can select the file that we want to work with, perform queries, add or delete a record, and obtain a list of all records.

```
TO DBM
LOCAL "OPTION
SETREAD [] SETWRITE [] CLOSEALL
MAKE "FILENAME []
LABEL "BEGIN
CLEARTEXT
PRINT [DATABASE MANAGER]
PRINT [_____]
PRINT []
```

```
PRINT [1\) SELECT FILE]
PRINT [2\) QUERY]
PRINT [3\) ADD RECORD]
PRINT [4\) DELETE RECORD]
PRINT [5\) LIST RECORDS]
PRINT [0\) QUIT]
PRINT []
PRINT [Choose option:]
MAKE "OPTION RC PRINT :OPTION
IF :OPTION = 0 [STOP]
IF :OPTION = 1 [SELECT]
IF :OPTION = 2 [QUERY]
IF :OPTION = 3 [ADDRECORD]
IF :OPTION = 4 [DELETE]
IF :OPTION = 5 [LISTALL]
PRINT []
PRINT [Press a key to continue]
MAKE "OPTION RC
GO "BEGIN
END
```

At this point, we wish to analyse the various options of the menu presented by procedure DBM in order of increasing complexity of the corresponding procedures.

SELECT FILE (option 1)

This option lets us choose the file with which we want to work by calling procedure SELECT.

```
TO SELECT
TYPE [File name:]
MAKE "FILENAME READWORD
IF NOT FILEP :FILENAME [ (PRINT [File] :FILENAME [not found])
      MAKE "FILENAME [] STOP]
READTABLE
END
```

This procedure assigns the filename given by the user to variable "FILENAME. It checks the existence of the file and then calls procedure READTABLE, which reads the file into memory.

LIST RECORDS (option 5)

LIST RECORDS is the option that gives the listing of the whole table. The procedures are as follows:

```
TO LISTALL
IF EMPTYP :FILENAME [PRINT [No file selected!] STOP]
LISTRECORD :NRECORDS
PRINT SENTENCE [Total records:] :NRECORDS
END

TO LISTRECORD :NRECORD
LOCAL "RECORD
IF :NRECORD = 0 [STOP]
LISTRECORD :NRECORD - 1
MAKE "RECORD WORD "RECORD :NRECORD
```

```
PRINT :RECORD
SHOWFIELDS :NFIELDS
END

TO SHOWFIELDS :NFIELD
LOCAL "NAME
IF :NFIELD = 0 [STOP]
SHOWFIELDS :NFIELD - 1
MAKE "NAME THING WORD "FIELD :NFIELD
SPACES 5 TYPE WORD :NAME ":
SPACES 12 - COUNT :NAME
PRINT GPROP :RECORD :NAME
END
```

LISTRECORD shows the records one after the other by calling SHOWFIELDS, which displays all the fields of a given record. Procedure SPACES puts :N spaces on the screen:

```
TO SPACES :N
REPEAT :N [TYPE CHAR 32]
END
```

ADD RECORD (option 3)

To add a new record we make use of procedure ADDRECORD, which calls NEWRECORD.

```
TO ADDRECORD
IF EMPTYP :FILENAME [PRINT [No file selected!] STOP]
NEWRECORD
END

TO NEWRECORD
MAKE "NRECORDS :NRECORDS + 1
GETFIELDS :NFIELDS
OPEN :FILENAME
SETWRITE :FILENAME
WRITEFIELDS :NFIELDS
SETWRITE []
CLOSE :FILENAME
END
```

NEWRECORD updates the number of records. It calls GETFIELDS in order to request the values of each field and then adds the new record to the file by calling WRITEFIELDS :NFIELDS.

```
TO GETFIELDS :NFIELD
LOCAL "NAME
IF :NFIELD = 0 [STOP]
GETFIELDS :NFIELD - 1
MAKE "NAME THING WORD "FIELD :NFIELD
TYPE WORD :NAME ":  SPACES 12 - COUNT :NAME
PPROP WORD "RECORD :NRECORDS THING WORD "FIELD :NFIELD READLIST
END
```

The MAKE instruction in procedure GETFIELDS assigns to variable "NAME the name of field number :NFIELD. The last instruction deserves further explanation. PPROP takes three inputs: a PL name, a

property name and a value to assign to the property. The first input is WORD "RECORD :NRECORDS, which outputs the name of a record (e.g.: "RECORD3). The second input is THING WORD "FIELD :NFIELD, which outputs the name of a field (e.g.: "WEIGHT). The third and last input is the list returned by READLIST.

Coming back to NEWRECORD, this procedure opens the file for writing and then calls WRITEFIELDS, which saves the fields of the new record on the file.

```
TO WRITEFIELDS :NFIELD
IF :NFIELD = 0 [STOP]
WRITEFIELDS :NFIELD - 1
PRINT GPROP WORD "RECORD :NRECORDS THING WORD "FIELD :NFIELD
END
```

GPROP takes two inputs which correspond to the first two inputs of PPROP and outputs a value. This value corresponds to the third input of PPROP.

DELETE RECORD (option 4)

Option DELETE RECORDS allows us to logically delete a record. By 'logically' we mean that the record is not physically eliminated from the file, but simply marked so that it can be ignored by the READTABLE procedure. By simply employing a text editor it would be possible to remove the mark and recover the record.

```
TO DELETE
(LOCAL "NUM "FPOS "RECORD)
IF EMPTYP :FILENAME [PRINT [No file selected!] STOP]
LABEL "REQUEST
TYPE [Record N.]
MAKE "NUM READWORD
IF NOT NUMBERP :NUM [GO "REQUEST]
IF OR :NUM < 1 :NUM > :NRECORDS [GO "REQUEST]
MAKE "RECORD WORD "RECORD :NUM
PRINT :RECORD
SHOWFIELDS :NFIELDS
PRINT []
IF NOT AFFERMATIVE [Delete this record?] [STOP]
OPEN :FILENAME
SETREAD :FILENAME
MAKE "NFIELDS FIRST READLIST
REPEAT :NUM [SKIPDELETED SKIPRECORD]
SKIPDELETED
SETREAD []
SETWRITE :FILENAME
SETWRITEPOS :FPOS
TYPE "&
SETWRITE []
CLOSE :FILENAME
READTABLE
(PRINT [Record N.] :NUM [deleted])
END
```

This procedure asks for the sequential number of the record to be deleted. After visualizing the contents and obtaining a confirmation, it substitutes the first char of the first field (field "FAMILY in our example) with the symbol &. In this way procedure READTABLE will ignore the corresponding record.

AFFERMATIVE is a general purpose predicate that can be used in many contests:

```
TO AFFERMATIVE :QUESTION
LOCAL "INPUT
PRINT []
PRINT SENTENCE :QUESTION [\(Y\/N\)]
MAKE "INPUT READLIST
IF :INPUT = [Y] [OUTPUT "TRUE]
IF :INPUT = [N] [OUTPUT "FALSE]
PRINT [Please, reply "Y" or "N"]
OUTPUT AFFERMATIVE :QUESTION
END
```

Procedure SKIPDELETED lets us skip logically deleted records, whereas procedure SKIPRECORD skips one record which consists of :NFIELDS lines.

```
TO SKIPDELETED
LOCAL "FLAG
MAKE "FPOS READPOS
MAKE "FLAG FIRST FIRST READLIST
SETREADPOS :FPOS
IF NOT EQUALP :FLAG "& [STOP]
SKIPRECORD
SKIPDELETED
END

TO SKIPRECORD
LOCAL "LINE
REPEAT :NFIELDS [MAKE "LINE READLIST]
END
```

Note that SKIPDELETED modifies the non-local variable "FPOS which will contain the position in the file of the first character of the record to be deleted. This character is the one that will be substituted by &.

QUERY (option 2)

QUERY is the most important option in the managing of a database. It allows us to quickly identify those records of the table that satisfy given conditions. We can have one single condition. For example, we could ask the list of all the pupils older than a certain age. We can also give more conditions involving one or more fields. For example, we could ask the list of girls younger than a certain age and heavier than a certain weight, or the list of pupils whose height is within a given range.

When our query contains more than one condition, these can be linked between them with AND/OR operators. For example, if we want to know how many girls weigh more than 55 kg the conditions are:

```
SEX = "F  AND  WEIGHT > 55
```

In this case we have to find the intersection between two groups: the group of the girls and the group of the pupils that weigh more than 55 kg.

In another case, we could be interested in the union of two groups: the pupils that weigh less than 40 kg and those with a height less than 145 cm. In this case, the conditions are:

```
WEIGHT < 40   OR   HEIGHT < 145
```

In the program that we are presenting we will limit ourselves to simple queries — one condition — and to queries with two conditions connected with AND/OR operators.

We now examine the procedures.

```
TO QUERY
IF EMPTYP :FILENAME [PRINT [No file selected!] STOP]
PRINT []
PRINT [1\) ONE CONDITION]
PRINT [2\) TWO CONDITIONS]
PRINT [0\) TO MAIN MENU]
PRINT []
PRINT [Choose option:]
MAKE "OPTION RC PRINT :OPTION
IF :OPTION = 1 [SINGLEQUERY]
IF :OPTION = 2 [DOUBLEQUERY]
END
```

Procedure QUERY presents a submenu from which we can choose between single or double conditions.

```
TO SINGLEQUERY
(LOCAL "PROP "COND "HOWMANY)
LISTFIELDS :NFIELDS
PRINT []
TYPE [Field No:]
MAKE "PROP THING WORD "FIELD FIRST READLIST
TYPE SENTENCE [Condition on field] WORD :PROP ":
MAKE "COND READLIST
MAKE "HOWMANY 0
LISTSQ :PROP :COND :NRECORDS
PRINT SENTENCE [Total records:] :HOWMANY
END
```

Procedure SINGLEQUERY calls LISTFIELDS to list the names of all the fields. It asks the number of the field involved in the query and then asks for the condition. The latter must be written as

```
<relational operator> <value>
```

where <relational operator> can be =, < or >, and <value> can be a word (preceded by a quotation mark) or a number. The following are sample conditions:

```
= "F
> 45
< 150
= "ROSSI
```

Once the field name and condition are obtained, procedure LISTSQ is called which searches and lists all records satisfying the condition.

```
TO LISTFIELDS :N
LOCAL "NAME
IF :N = 0 [STOP]
LISTFIELDS :N - 1
```

```
MAKE "NAME THING WORD "FIELD :N
TYPE WORD :N "\)\  PRINT :NAME
END
```

Procedure LISTFIELDS lists the field names preceded by a sequential number.

```
TO LISTSQ :PROP1 :REL1 :NRECORD
LOCAL "RECORD
IF :NRECORD = 0 [STOP]
LISTSQ :PROP1 :REL1 :NRECORD-1
MAKE "RECORD WORD "RECORD :NRECORD
IF RUN SENTENCE GPROP :RECORD :PROP1 :REL1 [PRINT :RECORD
     SHOWFIELDS :NFIELDS MAKE "HOWMANY :HOWMANY + 1]
END
```

Procedure LISTSQ, by calling itself recursively, examines one after another all the records in the table. When a record satisfies the specified condition it is displayed on the screen and counted. The key instruction is the one starting with IF RUN . . . Let's examine it.

The commands between square brackets are executed only when the RUN operation outputs TRUE. RUN outputs the result of the execution of its input list, which in our case is the output of the following operation

```
SENTENCE GPROP :RECORD :PROP1 :REL1
```

The first input of SENTENCE is GPROP :RECORD :PROP1, which yields the value of the property :PROP1 of the PL :RECORD. This is the value of the searched field. The second input is :REL1, which is the condition given by us. For example, if the values of :RECORD, :PROP1, and :REL1 are respectively "RECORD1, "WEIGHT, and [> 60], the output of SENTENCE is (cf. Table 1)

```
[54 > 60]
```

and RUN outputs FALSE.

```
TO DOUBLEQUERY
(LOCAL "PROP1 "COND1 "PROP2 "COND2 "OPER "HOWMANY)
LISTFIELDS :NFIELDS
PRINT []
TYPE [No. of first field:]
MAKE "PROP1 THING WORD "FIELD FIRST READLIST
TYPE SENTENCE [Condition on first field] WORD :PROP1 ":
MAKE "COND1 READLIST
TYPE [No. of second field:]
MAKE "PROP2 THING WORD "FIELD FIRST READLIST
TYPE SENTENCE [Condition on second field] WORD :PROP2 ":
MAKE "COND2 READLIST
TYPE [1\) AND 2\) OR]
MAKE "OPER IF RC = 1 ["AND] ["OR]
PRINT []
MAKE "HOWMANY 0
LISTDQ :OPER :PROP1 :COND1 :PROP2 :COND2 :NRECORDS
PRINT SENTENCE [Total records:] :HOWMANY
END
```

Procedure DOUBLEQUERY asks for the field number and relation for each of the two conditions involved in the query. It then asks whether AND or OR should be used. Finally, it calls procedure LISTDQ to show

and count all the records satisfying the given conditions.

```
TO LISTDQ :OPER :PROP1 :REL1 :PROP2 :REL2 :NRECORD
LOCAL "RECORD
IF :NRECORD = 0 [STOP]
LISTDQ :OPER :PROP1 :REL1 :PROP2 :REL2 :NRECORD-1
MAKE "RECORD WORD "RECORD :NRECORD
IF RUN (LIST :OPER WORD "" (RUN SENTENCE GPROP :RECORD :PROP1
      :REL1) WORD "" (RUN SENTENCE GPROP :RECORD :PROP2 :REL2))
      [PRINT :RECORD SHOWFIELDS :NFIELDS MAKE "HOWMANY :HOWMANY + 1]
END
```

Procedure LISTDQ is similar to LISTSQ, but with more inputs. Besides, RUN has the following more complex input:

```
(LIST :OPER WORD "" (RUN SENTENCE GPROP :RECORD :PROP1 :REL1) WORD ""
      (RUN SENTENCE GPROP :RECORD :PROP2 :REL2))
```

LIST outputs a list of three elements. The first element is :OPER, which is "AND or "OR. The second element is WORD "" (RUN SENTENCE GPROP :RECORD :PROP1 :REL1), which puts a quotation mark in front of the output of RUN yielding "TRUE or "FALSE. The third element is similar to the second, but related to the second condition involved in the query. Therefore, it will also output "TRUE or "FALSE. The resulting list could be for example: [OR "TRUE "FALSE].

Use of the program

In order to start the program we must type DBM at the Logo prompt. We are not showing an example here due to the fact that being organized as a menu, its use should be immediate. We only need to be careful while inputting data to place a quotation mark before nonnumeric words.

Improvements

The program showed in the present paper is only a first step towards a database managing program. Many improvements can be suggested. For example: - modifying only part of a record; - checking the existence of the record in the option ADD RECORD; - recovering a logically deleted record; - permanently removing a logically deleted record; - printing the whole table or the result of a query; - including NOT in the query conditions; - checking the validity of the input data; - creating new table structures.

Conclusions

Logo is certainly not the language that one would choose to develop a professional program for database managing. However, it is powerful enough to let us manage a small table. Principally, it allows pupils to detect some basic problems that do emerge when one deals with a database. Furthermore, due to the fact that they are already familiar with the language, it can stimulate their curiosity about how the procedures work and they could therefore be led to try to modify them or add others of their own.

REFERENCES

[1] Tsichritzis D.C., Lochovsky F.H.: *Data Models*, Prentice-Hall, 1982.

[2] Harvey B.: *Computer Science Logo Style*, Vol.1, MIT Pess, 1985.

La Tortue, le Formalisme Mathematique, et les Connaissances Intuitives...

Dapontes Nikos

Papazachariou 27
Agios Dimitrios
Athens, Greece
tel: (301) 99 63 864

Abstract

The aim of this paper is to present an alternative method for problem solving in "geometrical constructions" without the use of special mathematical background. The alternative process consists of four guiding ideas concerning knowledge, method, "sensibilisation" of geometrical constructions and the structure of LOGO language. We have chosen four tasks of geometrical construction to show how we can apply our heuristic method "angular and linear scanning". Our intention is to introduce a similar strategy for geometry instruction into the LOGO environment.

Keywords: Logo, geometry, programming, problem-solving

Nikos Dapontes *was born in Ermoupolis (Syros) in 1948. He studied Physics at the University of Patras and obtained DEA (Didactique de la Physique) at the University of PARIS VII. He is the author of a book concerning the teaching of LOGO. Now, he is working in Secondary Education.*

La Tortue, le Formalisme Mathematique, et les Connaissances Intuitives...

1. Avec la tortue, il faut penser différemment

Les constructions des figures dont on impose certains éléments et les problèmes sur les "Lieux Géométriques" sont deux types de problèmes que les élèves envisagent dans l'enseignement de la géométrie. Ces exercices de construction, autorisent l'usage de la règle et du compas, exigent de la part des élèves une approche par analyse-synthèse et en même temps la maîtrise des connaissances spécifiques au problème concerné.

Dans la perspective d'aborder les activités de construction sur l'écran d'un moniteur, en milieu de LOGO, on est conscient qu'il faut penser différemment. Un essai pour résoudre ces problèmes en utilisant le formalisme mathématique comme connaissances des références, s'est avéré une chose facile et sans aucun intérêt particulier. Cette démarche n'est qu'un transfert direct des connaissances et des méthodes d'un domaine à un autre.

En essayant de faire le tracé d'une perpendiculaire sur une droite donnée, dans l'environnement LOGO, j'ai trouvé plusieurs solutions (programmes) en utilisant les connaissances de la géométrie Euclidienne, la trigonométrie et la géométrie analytique. Mais, comme je n'étais pas satisfait, il m'a paru intéressant d'abandonner à la fois le formalisme mathématique et la méthode privilégiée, l'analyse et la synthèse. Après cela, je suis arrivé à une heuristique gui est basée sur des connaissances intuitives. Cette heuristique s'est avéré propice pour construire d'autres figures et pour trouver la solution aux problèmes sur les "lieux géométriques". Son efficacité et sa généralité quand on envisage les problèmes de construction simples, sont les indices suffisants pour avour toute la confiance en cette heuristique.

L'objectif de mon travail est de présenter un processus alternatif pour la résolution des problèmes de construction géométriques sans l'utilisation de connaissances spécifiques. L'alternative consiste en quatre idées directrices qui concernent les connaissances, la méthode, la "sensibilisation" de la construction sur l'écran, et la structure du langage LOGO.

PREMIERE IDEE.
Nous abandonnons volontiers le formalisme mathématique "prêt-à-porter". A sa place, nous utiliserons les connaissances intuitives que l'homme peut avoir avec son corps (il se met en mouvement et en rotation, il se dirige vers un point) [7, 8] et avec son "regard" (perception d'un object, estimation des angles et des distances). A partir de ces connaissances on peut saisir des idées pour procéder à un problème de construction dans l'environnement LOGO.

DEUXIEME IDEE.
Nous abandonnons la méthode d'analyse-synthèse telle qu'elle est appliquée en géométrie Euclidienne [1, 2, 3]: on fait une figure d'étude et on analyse la configuration à laquelle on doit aboutir. A sa place, j'ai préferé faire une analyse et un raisonnement en même temps qu'un processus de construction "incomplet". J'ai pensé, alors, en terme d'état initial et d'état final auquel on arrive en appliquant des "opérateurs" [5].

TROISIEME IDEE.
Je désire que la programmation soit telle qu'on fasse apparaître clairement sur l'ecran, le processus de construction. Cela veut dire qu'on veut "sensibiliser" la méthode de construction, à partir d'une figure "incomplète" jusqu'à la figure désirée.

QUATRIEME IDEE.
La structure du langage informatique est centrée sur les propriétés et les "capacités" de la tortue.
De mon point de vue, les instructions et les fonctions de LOGO, conçues pour le mouvement et la rotation ont une structure bien déterminée:

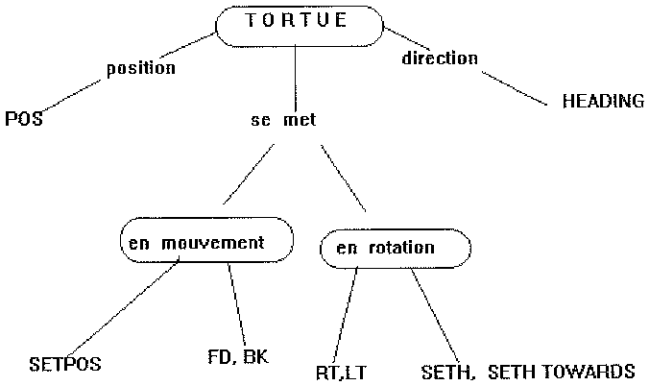


Schéma 1

Ainsi, se trouvent apparemment mêlées deux géométries, l´une de type "Global" (le plain - c´est-à-dire l´écran - est muni d´un repère XOY) et l´autre de type "Local" (le plan n´est pas orienté a priori). [6]
La tortue dispose d´autres capacités importantes pour sa programmation. Ainsi, la tortue peut:

- calculer (+. -./. *. ROUND)
- accepter des données (RW, RL)
- écrire sur l´écran (TYPE, PR, SETCURSOR)
- décider (IF <condition> <action>)
- mémoriser (MAKE)

Pour mon travail, j´ai ajouté une seule procédure avec laquelle on peut calculer la distance de la tortue de n´importe quel point de l´écran, à un point P (x,y).
Nous avons appelé, DISTXP: P, la macroprocédure qui accomplit cette tâche. Avec cet ensemble de primitives LOGO, nous sommes prêts à programmer la tortue, à construire des figures simples et à rechercher des "lieux géométriques".

2. Le"balayage": une heuristique pour la détextion d' une direction ou d' unpoint

Nous avons choisi quatre problèmes de construction pour démontrer comment s´applique l´heuristique du balayage angulaire et linéaire. Comment programmer la tortue:

(I) faire la trace d´une perpendiculaire sur une droite?
(activité de la "perpendiculaire")
(II) construire un triangle rectangle quand on connaît l´hypoténuse et un côté perpendiculaire?
(activité du "triangle rectangle)
(III) construire un triangle quand on connaît les trois côtés?
(activité du "triangle-1")
(IV) construire un triangle quand on connaît le côté (a), la hauteur (ha) et l´angle(B), du triangle ABC?
(activité du "triangle-2")

L´ACTIVITE DU "PERPENDICULAIRE"

Les connaissances intuitives seront le point de départ. Quand je veux avancer perpendiculairement vers une droite sur le sol (ou faire la perpendiculaure à une droite dans l´espace), je dirige mon "regard" vers un point fixe de la droite. Ensuite je balaie cette droite avec mon regard. Pendant ce balayage, j´estime l´angle (φ) entre la droite et mon "regard", celui-là étant une droite partant de mes yeux jusqu´à un point de la droite donnée. Je constate que mon regard se dirige perpendiculairement vers la droite si l´angle $\varphi = 90^{\circ}$
Ainsi, je détecte et trouve la direction désirée. Puis, j´avance, et comme je dispose de la vision je sais à quel moment il faut terminer mon mouvement. Cette métaphore, n´a rien à voir avec la géométrie scolaire.

Pourtant, celle-ci m´a donné quelques idées pour programmer la tortue pour accomplir sa tâche sur l´écran. On distingue les trois phases suivantes:

- les conditions initiales
- la détection d´une direction et
- la détection d´un point sur une droite fixe.

Soit, <PERP> le nom du programme "perpendiculaire".

PREMIERE PHASE: Les conditions initiales.

Nous appelons <CIPERP> la procédure qui accepte les données du probléme: deux point. P1(x1.y1) et P2(x2,y2) qui déterminent la droite et un point P0(x0,y0) que l´on considère à droite de P1, comme le schéma 2 le montre.

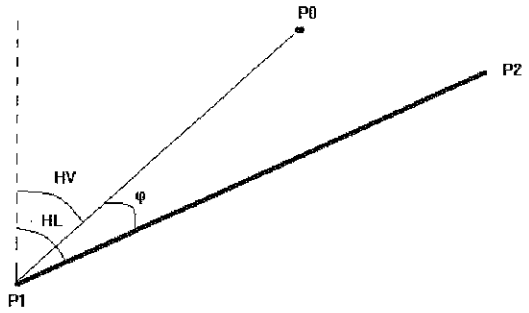


Schéma 2

La tortue fait la trace P1P2, et en même temps calcule l´angle, φ = HL - HV
φ = direction (P1P2)-direction (P1P0)
Celui-ci se traduit en langage de la tortue:
φ = HEADING (P1P2) - HEADING (P1P0)

DEUXIEME PHASE. La détection d´une direction.

Du point P0, la tortue se dirige vers le point fixe P1. Evidemment P0P1, n´est pas la perpendiculaire sur P1P2. Mais nous pouvons considérer cette situation comme un état "incomplet" par rapport à celui qu´on demande (schéma 3, 1). Si la tortue tourne un peu (d´1º, par exemple) vers la gauche, nous constatons facilement la propriété suivante:
< le nouvel angle φ = le précédent φ + 1º>

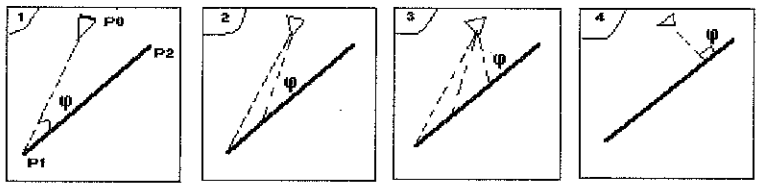


Schéma 3

La tortue va terminer sa rotation lorsque φ = 90º
Nous appelons "balayage angulaire", l´heuristique qui nous permet le passage de l´état (1) à l´état (2) et (3). Cette heuristique aide la tortue à détecter une direction, en suivant la propriété: φ -> : φ + 1
Nous appelons <BALPERP> la procédure qui joue le rôle du détecteur d´une direction.

TROISIEME PHASE: La détection complète de la perpendiculaire. La tortue avance d´un pas et calcule la direction de la droite P1P (5). Cette direction est l´angle HV.

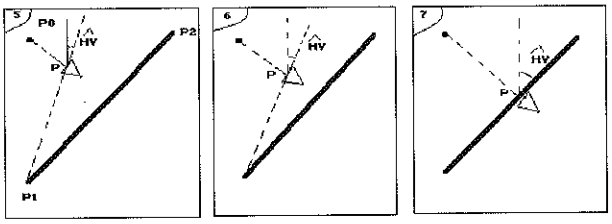


Schéma 4

Nous appelons "balayage linéaire" l'heuristique qui nous permet le passage de l'état incomplet (5) à l'état final (7). Cette heuristique aide la tortue à détecter un point sur une droite fixe, en suivant une propriétè:
<P1P coïncide avec P1P2, si leurs directions sont les mêmes>
Nous appelons <LINE> la procédure qui programme la tortue pour tracer la perpendiculaire.

Le programme de la construction prend la forme suivante:

```
TO PERP
CIPERP    -> Introduction des données (P1,P2,P0)
BALPERP   -> Balayage angulaire et puis linéaire
END
```

Supposons qu'on veuille tracer P0P à partir de P1P2 et P0, avec P0PP2 = P (connue).
Cette tâche est très facile si on met en fonction un balayage angulaire et puis un balayage linéaire, comme nous l'avons déjà réalisé lors du problème de la "perpendiculaire".
La seule différence se trouve dans la condition d'arrêt:
au place de $\varphi = 90^{\circ}$ on doit poser $\varphi = P$.
Une approche formelle se heurte à des difficulités.

L'ACTIVITE DU "TRIANGLE RECTANGLE"

Nous allons commencer à partir des données, a (la longueur de l'hypoténuse) et b (la longueur d'un côté perpendiculaire). On commande à la tortue de tracer un segment BC = a, qui réprésente l'hypoténuse du triangle. Avec le côté de longueur b, on peut dessiner un triangle qui ne satisfait pas à toutes les conditions de la construction. Bien sûr, il s'agit d'un état incomplet:
l'angle BAC n'a pas pas 90º

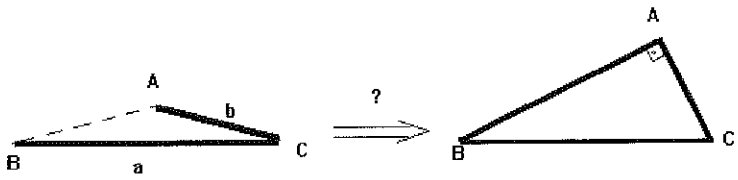


Schéma 5

Qu'est-ce que nous pouvons faire pour arriver à l'état final? Une idée est de pivoter le côté b (de longueur constante) autour du point fixe C, c'est-à-dire, mettre en fonction l'heuristique du balayage angulaire.
Une série de bandes dessinées montre clairement le processus de construction.

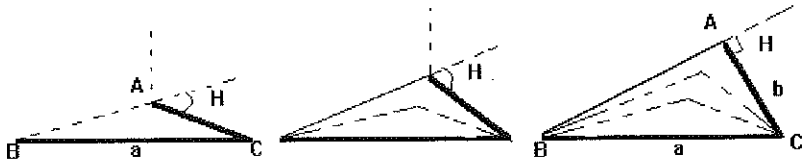


Schéma 6

Nous identifions comme variable convenable, l'angle H, que la tortue peut calculer facilement:
H = direction (AC) - direction (BA)
Le programme de la construction d'un triangle rectangle à partir des données {a. b. A = 90o} prend la forme suivante:

```
TO TRIOR
CITRIOR    -> Introduction des données
BALTRIOR -> Balayage angulaire d'un rayon constant, calcul d'un ang variable
END
```

Comment peut-on construire un triangle ABC si on connaît les deux côtés a, b et l'angle A=90º? Il suffit d'un changement dans la condition d'arrêt:
(poser H = 180º - A, à la place de H = 90º)

L'ACTIVITE DU "TRIANGLE -1"

On dessine un côté du triangle par exemple, le côté (a). Avec le deuxième côté, de longueur (b), on peut construire un triangle "incomplet". Pendant le balayage angulaire du rayon constant (b), autour du point fixe C, la distance AB change. Si cette distance devient égale à la longueur du troisième côté (c), le balayage doit s'arrêter.

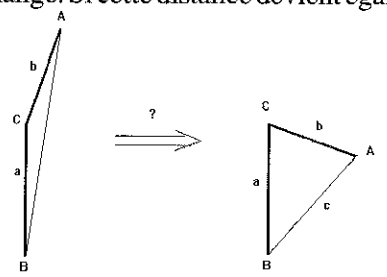


Schéma 7

Une série de bandes dessinées montre le processus de la construction.

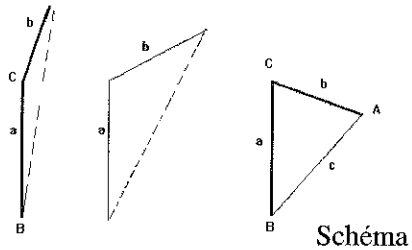


Schéma 8

Le programme de construction prend la forme suivante:

```
TO TRI1
CITRI1      -> Introduction des données (a. b. c.)
BALTRI1     -> Balayage angulaire d'un rayon constant, calcul d'une distance variable
END
```

L'ACTIVITE DU "TRIANGLE -2"

A partir des données (a) (longueur du côté BC) et (h) (hauteur du triangle) nous pouvons construire un triangle "incomplet" ABC. Ce triangle satisfait aux obligations BC = a, CA = h, mais l'angle B n'est pas exact.

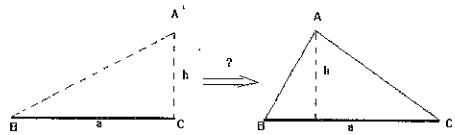


Schéma 9

Alors, le problème est de trouver la position exacte du point A tel que l´angle ABC = B (donné).
Nous avons deux chemins à suivre. Soit un balayage linéaire du point A´, parallèlement à BC, soit un balayage linéaire du point B dans une direction qui fait un angle B avec BC (schéma 10). On va montrer le deuxième chemin.

Du point B, la tortue prend une direction d´angle B. Ensuite, elle avance et calcule l´angle H. Auparavant, la tortue mémorise les coordonnées du point A´. La suite apparaît clairement en bandes dessinées:

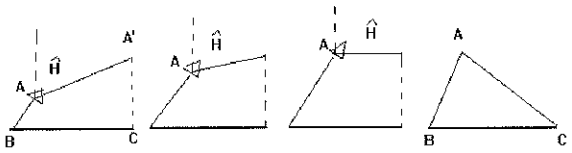


Schéma 10

Le programme de construction du "triangle -2":
TO TRI2
CITRI2 -> Introduction des données {a. h. B}
BALTRI2 -> Balayage linéaire à partir du point B.
END

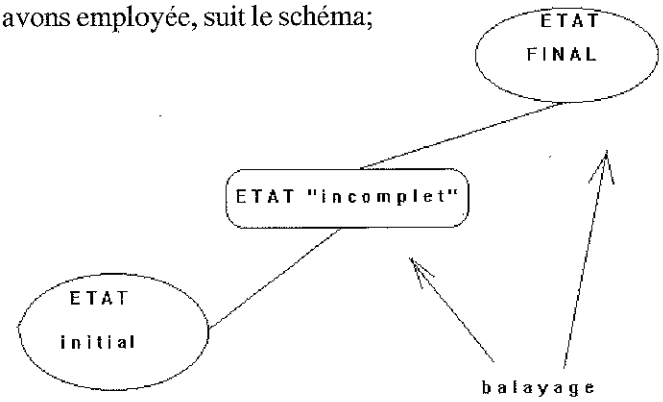
3. Epilogue

Tous les programmes qu´on peut créer en utilisant les connaissances mathématiques et la méthode "analyse-synthèse", ont une structure "sequentielle". Ils ne présentent pas la moindre difficulté du point de vue de la programmation en LOGO. D´autre part, tous les programmes de constructions, PERP., TRIOR, TRI1 et TRI2, ont une structure commune. Lorsqu´un tel programme est activé, il présente, pas-à-pas, le processus de la construction sur l´écran.
La structure est la suivante:

TO<NOM> -> Nom du programme
CINOM -> Conditions initiales
BALNOM -> Balayage angulaire et/ou linéaire
END

La procédure <BALNOM> consiste en une structure ITERATIVE.
Elle comprend, un ensemble des instructions qui doivent être répétées, et une condition d´arrêt qui sert à assurer que le programme "ne boucle pas indéfiniment".

La stratégie que nous avons employée, suit le schéma;



En premier lieu, nous cherchons un état "incomplet" en partant d'un état initial du problème. Il s'agit d'une transformation des conditions initiales. L'"incomplet" caractérise un état dans l'espace du problème, qui satisfait partiellement aux exigences de la construction. Ensuite, on cherche une quantité variable (angle ou distance) qui convient au prolème. A la fin, il est nécessaire de trouver une condition d'arrêt au processus. On peut dire que le passage des données à la construction s'est opéré en faisant "fonctionner" l'heuristique de balayage.

·La géométrie de type "Papertien", qui favorise une géométrie de type "Local" ne suffit pas pour **explorer** les constructions de figures géométriques simples. On a choisi de travailler dans l'ensemble des instructions et des fonctions de type local et global du système LOGO (voir schéma 1). D'ailleurs, il n'y a pas de raison de limiter les possibilités du système LOGO, quand on veut travailler avec les élèves du collège [8].

·Dans l'approche alternative que nous avons proposée, on considère les problèmes de construction géométrique simple comme des **problèmes de transformations** [4].

·La technique de balayage consiste en un outil de transformation d'une configuration en une autre. Elle peut offrir un moyen pour la résolution des problèmes sans l'utilisation des connaissances spécifiques.

REFERENCES

[1] DISSESA A. (1977): On "learnable" representations of knowledge: A meaning for the computational metaphore. *M. I. T. LOGO MEMO* No. 47.

[2] DISSESA A. (1987): Artificial worlds and real experience. In: R. Lawler, M. Yazdani, *"Artificial Intelligent and education"*, Volume one, Ablex publishing.

[3] HALBWACHS F. (1981): Significations et raisons dans la pensée scientifique. *Archives de Psychologie*, XLIX (190), 1981.

[4] GREENO G. (1978): Nature of problem-solving abilities. In: W. K. Estes (Ed.), *Handbook of learning and cognition processes*.L. E. A.

[5] MAYER R. (1981): *The promise of cognitive Psychology*. Freeman and Co, San Francisco

[6] MYX A. (1988): *Logo géométrique et l'école élémentaire*. "petit x", no 18.

[7] PAPERT S. (1980): *Mindstroms: Childrens, Computers and powerful ideas*. New York: Basic Books.

[8] PAPERT S. (1987): Microworlds: Transforming education. In: R. Lawler, Yazdani (Ed), *Artificial intelligence and education*, volume one, Ablex Publishing.

Exploring Recursion: from Random Walks to the Design of Mazes

Gerald Futschek[1], **Helmut Schauer**[2]

[1] Institute for Software Technology
Vienna University of Technology
A-1040 Vienna, Austria
e-mail: futschek@eimoni.tuwien.ac.at

[2] Institute for Informatics
University of Zurich
CH-8057 Zurich, Switzerland
e-mail: schauer@ifi.unizh.ch

Abstract

The purpose of this paper is to illustrate experimental learning in combination with evolutionary program design using an example in which we generate and draw random mazes. We begin by discussing a simple program that draws random walks. We then experiment with this program by adding interesting features step by step until we have designed an efficient program that generates and draws mazes. In the course of designing this program we will obtain some interesting programs that draw realistic trees and plants. The Logo programming language proves to be an excellent and adequate tool to support this approach since the resulting programs are short and comprehensible.

Keywords: recursion, turtle geometry, experimental learning, mazes

Gerald Futschek *is an assistant professor in the Department of Computer Science at the Vienna University of Technology where he is responsible for Software Engineering education. His research interests include program verification and teaching methods in information technology. He studied mathematics and computer science and received his Ph.D. from the Vienna University of Technology in 1984. For the last 10 years he and Helmut Schauer have organized many computer camps for children where the Logo language was used to provide children with a comprehensible and creative means to explore computer technology. He also held several computer science courses for teachers where he dealt with subjects related to Logo.*

Exploring Recursion: from Random Walks to the Design of Mazes

1. Introduction

Finding paths in complicated mazes is a challenging task for developing search strategies. Hence mazes are used to test different learning strategies of humans, animals and robots. The design of good mazes also provides a challenging task. Furthermore we want to describe the procedure of designing random mazes that can be drawn by a robot. The LOGO system is an adequate tool for describing the corresponding algorithm and for drawing the mazes using turtle geometry [1], which simulates the robot.

At first it is not obvious how a procedure to generate random mazes works. Hence we try an experimental and evolutionary approach. We start with a simple program that draws random curves and add more and more features that we need for generating a maze. This way of presentation can be used in programming courses to demonstrate experimental program development and to show the efficiency and comprehensibility of recursive programs.

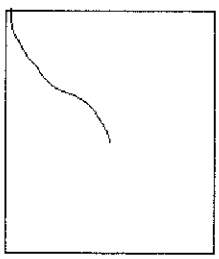
A maze consists of many randomly formed curves which are connected with each other in a systematic way. We start with a program that draws random curves.

2. Random walk

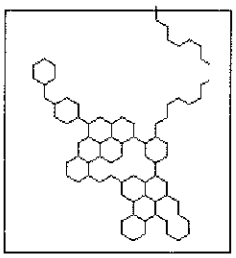
A randomly formed curve can easily be drawn with the following procedure `MOVE` which is often programmed in a similar fashion e.g. in [4, 5].

```
TO MOVE :CHOICE
  rt first :choice
  fd 10
  if not border [move shuffle :choice]
END
```

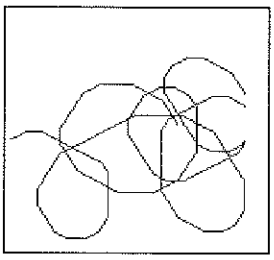
The turtle turns its heading before each forward step. We use a list `CHOICE`, which contains the values of angles, that can be used to change the heading of the turtle. The function `SHUFFLE :CHOICE` outputs the randomly ordered list `CHOICE`. The function `BORDER` outputs `TRUE` if the turtle is outside a certain border. `SHUFFLE` and `BORDER` are described in the appendix. Let us have a look at some results of calls of `MOVE` with different lists of angles. The turtle starts in the center of the screen. The recursive moves terminate when the turtle crosses the border.



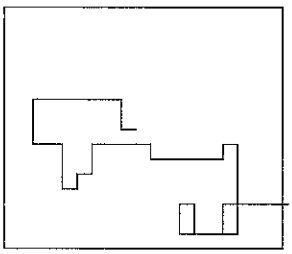
`MOVE [-15 0 15]`
small angle values produce a smooth curve



`MOVE [-60 60]`
60 degree angles produce hexagonal walks



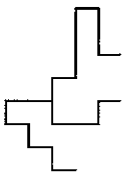
MOVE [-30 0]
an asymmetric list of angles produces spiral shaped walks that tend to the preferred side



MOVE [-90 0 0 0 90]
a multiple occurrence of an angle (here 0) produces walks with a preferred angle

Executions of MOVE with different parameters show that we obtain some interesting curves, with some lines crossing each other. In order to avoid line crossings we introduce a list TRACE which contains the points of the curve that have been drawn already. We then proceed to draw the curve only if the current position of the turtle is not a member of the list TRACE.

```
TO MOVE :CHOICE
  RT FIRST :CHOICE
  FD 10
  IF NOT OR BORDER MEMBERP POS :TRACE
     [MAKE "TRACE FPUT POS :TRACE MOVE SHUFFLE :CHOICE]
END
```



```
CS MAKE "TRACE (LIST POS)
MOVE [-90 0 90]
SHOW :TRACE
[[-10 40] [0 40] [0 50] [0 60] [0 70] [10 70] [10 60] [10 50] [20 50]
[20 40] [20 30] [10 30] [10 20] [0 20] [-10 20] [-10 30] [-20 30]
[-30 30] [-30 20] [-20 20] [-20 10] [-10 10] [-10 0] [0 0]]
```

Note that the simple check for line crossing with the primitive function MEMBERP works only if we use appropriate lists of angles which ensure a walk on a certain grid of points. Hence the following lists and all their sublists may be used: [-90 0 90], [-120 -60 0 60 120]. Multiple occurrences of values are also possible.

If we want to use other angle values we can write a new function CROSSING instead of MEMBERP. CROSSING should output TRUE if the turtle is not more than a certain small distance away from a point in the list TRACE.

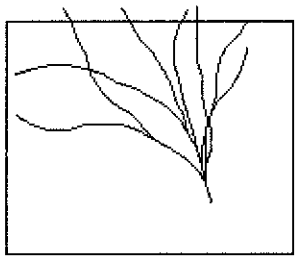
3. Drawing realistic plants

In order to study the properties of random walks we want to see the results of many executions of MOVE with the same list of angles. The execution of the inverse turtle operations at the end of the procedure moves the turtle back to the starting point:

```
TO MOVE :CHOICE
  rt first :choice
```

```
    fd 10
    if not border [move shuffle :choice]
    bk 10
    lt first :choice
END
```

Note that the second half of the procedure MOVE reverses the first half.
The repeated execution of this modified MOVE-procedure draws a realistic plant.



REPEAT 8 [MOVE [-15 0 15]]

Fascinated by the possibility of drawing plants we proceed to modify MOVE so that it draws a realistic tree. Two successive recursive calls of MOVE do the trick.

```
TO MOVE :CHOICE
  IF EMPTYP :CHOICE [STOP]
  RT FIRST :CHOICE
  FD 10
  IF NOT BORDER [MOVE BF SHUFFLE :CHOICE
                 MOVE BF SHUFFLE :CHOICE]
  BK 10
  LT FIRST :CHOICE
END
```



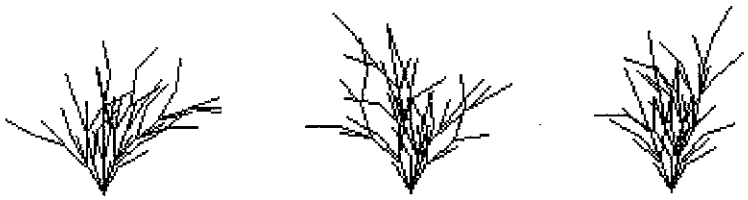
4 calls of MOVE [-20 -10 0 10 20 30]

4. Plants with multiple branches

A real plant has sometimes more than two branches at a branching point. Since a tree with multiple branches consists of a line and many subtrees we use a recursive call for every angle of CHOICE. This is done by the recursive call MOVE BF :CHOICE at the end of MOVE. The function BF in the line MOVE BF SHUFFLE :CHOICE is used to obtain different heights of the plant.

```
TO MOVE :CHOICE
    IF EMPTYP :CHOICE [STOP]
    RT FIRST :CHOICE
    FD 10
    MOVE BF SHUFFLE :CHOICE
    BK 10
```

```
  LT FIRST :CHOICE
  MOVE BF :CHOICE
END
```

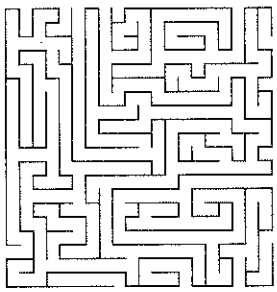


3 different calls of `MOVE [-30 -20 0 10 20 30]` produce bushes or tafts of grass.

Please note that the number of branches decreases with the depth of recursion. Since the number of items in the list `CHOICE` decreases by 1 with each recursive call, the end condition `EMPTYP :CHOICE` becomes true after a finite number of steps. Hence the condition `NOT BORDER` is not used in this version of the program.

5. Drawing mazes

Our aim is to design and draw random mazes. A desired maze could look like this:



A closer look at the previous maze shows that a maze is a (graph theoretical) tree without crossing lines. Since such a tree is free of cycles and a maze has no crossing lines we can assert that there is a way out of the maze from every point inside the maze. The reader may find it interesting that we can draw mazes with a program that was designed to draw trees.

The complete maze program:

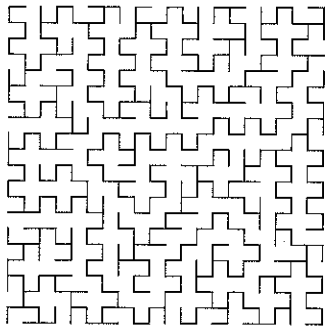
```
TO MAZE :ANGLES
  CS
  MAKE "TRACE (LIST POS)
  MOVE SHUFFLE :ANGLES
END

TO MOVE :CHOICE
  IF EMPTYP :CHOICE [STOP]
  RT FIRST :CHOICE
  PU FD 10
  IF NOT OR BORDER MEMBERP POS :TRACE
  [PD BK 10 FD 10 MAKE "TRACE FPUT POS :TRACE
```

```
    MOVE SHUFFLE :ANGLES]
  PU BK 10
  LT FIRST :CHOICE
  MOVE BF :CHOICE
END
```

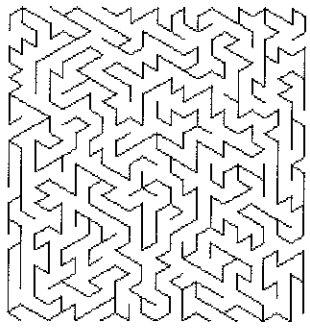
While drawing a maze the procedure MOVE tries to draw a line into all directions which are given in the list ANGLES at the current position of the turtle. This is achieved by the tail recursion MOVE BF :CHOICE at the end of MOVE. Since we do not want to cross the border or cross an existing line we first execute every step forward with pen up (PU FD 10). Then we check for a border or existing line at the new position, and we actually draw the line with pen down (PD BK 10 FD 10) only if the new position is valid. Then we draw the remainder of the maze with the recursive call MOVE SHUFFLE :ANGLES at the new position.

The values of the items in the list ANGLES remain unchanged during the execution of MAZE, but the function SHUFFLE puts them in a random order which represents the only random factor of the program.

Experiments with different parameters draw a variety of different mazes:

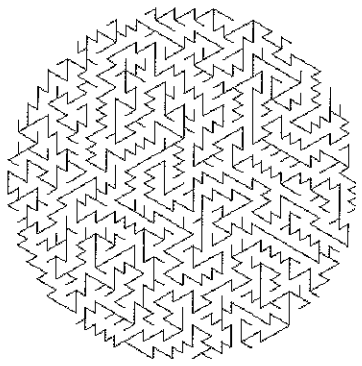


MAZE [-90 90]



MAZE [-120 -60 0 60 120]

2 examples of mazes with a quadratic border

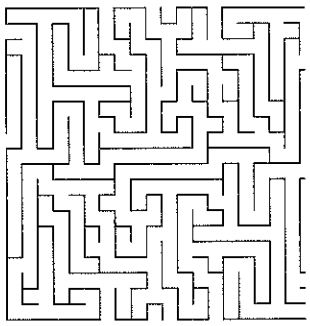


MAZE [-120 0 120]

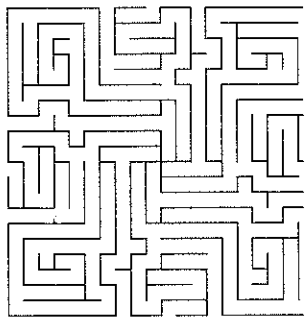
A maze with a cyclic border

6. Further ideas

Inspired by the results and motivated by the simplicity of the program the programmer may spend some time to explore more challenging tasks. The drawing of symmetrical mazes may be one of these tasks. The idea is that two or more simultaneously walking turtles can do the job. A logo system that provides more than one built-in turtle makes this approach substantially easier. At the beginning the turtles start at point symmetrical positions. If we use a symmetric border every turtle has the same possible steps to go. After every step of the turtles we have again a point symmetric maze with the same possible steps for all turtles. At the end we achieve a point symmetric maze.



point symmetric maze drawn by 2 synchronously walking turtles



point symmetric maze drawn by 4 synchronously walking turtles

Further ideas may include path finding algorithms for robots, plant drawing algorithms or computer art.

7. Conclusions

The project of designing random mazes presented in this paper illustrates how Logo supports creative problem solving. Experimental program development leads to efficient programs. The presented framework can be used in programming courses to explore the power of recursion.

Appendix

Here is a short description of our SHUFFLE and BORDER functions.

A. Shuffling a list

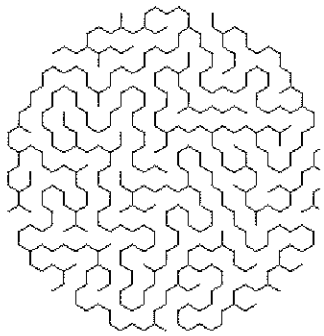
For the generation of a randomly shuffled list we use an algorithm which is very similar to quicksort. First each element of the list L is randomly appended to one of two lists X and Y. After recursive shuffling of X and Y, these lists are sentenced to the desired result.

```
TO SHUFFLE :L
  IF (COUNT :L) < 2 [OP :L]
  OP RNDM.LIST [] [] :L
END

TO RNDM.LIST :X :Y :L
  IF EMPTYP :L [OP SE SHUFFLE :X SHUFFLE :Y]
  IF 0 = RANDOM 2 [OP RNDM.LIST FPUT FIRST :L :X :Y BF :L]
  OP RNDM.LIST :X FPUT FIRST :L :Y BF :L
END
```

B. Border

The BORDER function outputs TRUE, if the turtle position is outside the drawing area of the maze. The drawing area can be bordered by any closed convex curve (e.g. a cycle, square or triangle etc.). In the following example the border is a cycle with a radius of 100 units.

```
TO BORDER
  IF XCOR * XCOR + YCOR * YCOR > 10000 [OP "TRUE]
  op "FALSE
END
```



MAZE [-60 60]

REFERENCES

[1] Abelson H., diSessa A.: *Turtle Geometry*, MIT Press Cambridge MA (1981).

[2] Dicheva D.: How to introduce recursion in Logo, *Proceedings of the Third European Logo Conference*, A.S.I.-Parma, 641-650 (1991).

[3] Futschek G.: *Computer camps for children*, internal paper (1985).

[4] Holroyd P.: Random Logo: a random look at random processes, *Eurologos Vol. 1*, 39-44 (1992).

[5] Schauer H.: *Logo, Jenseits der Turtle*, Springer-Verlag Wien New York (1988).

MicroBridge: a Combinatorical Microworld

Tammy Lapidot[1], Rinat Rachman[2]

Department of Education in Technology and Science
Technion - I.I.T., Haifa 32000, Israel
e-mail: [1] ttr0148@technion.bitnet
[2] Rachman@techunix.technion.ac.il

Abstract

In this paper we introduce a combinatorical microworld. This new microworld is based on a Bridge-like card game called MicroBridge.

Keywords: Logo, combinatorics, microworld, game

Tommy Lapidot *and* ***Rinat Rachman*** *graduated Computer Science from the Israel Institute of Technology in Haifa, Israel. At the present, they are in the department of Education in Technology and Science, Technion, and their main interest is Computer Science Education. Rinat is an M. Sc. student. Tammy has graduated M. Sc.*

MicroBridge: a Combinatorical Microworld

1. Introduction

Combinatorics is usually considered a difficult subject for learners. In this paper, we try to apply the microworld concept to the area of combinatorics. This is done by using a Bridge-like card game called MicroBridge.

In the Logo community, there is a consensus about the advantages of microworlds. Moreover, Papert [1] suggests "to create a network of microworlds, each one focusing on differnt areas of knowledge". Here, we introduce a new segment for such a network.

In creating our microworld, we followed two guidelines: representing the studied area (combinatorics), and keeping microwolrd's principles in mind.

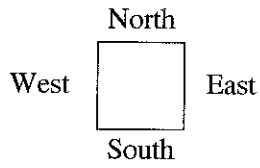
MicroBridge, which is a variant of the popular Bridge game, is a very sophisticated game. In order to play the game successfully, one must employ certain "mathematical considerations", which are mainly of a combinatorical type. We can use these combinatorical aspects of MicroBridge in order to reflect combinatorical issues.

MicroBridge is suitable to serve as a microworld, having the following features: the game is very attractive and exciting; it offers a variety of intellectual activities; and having more than one winning-strategy, contributes to its being open. Consequently, students are likely to be enthusiastic and cooperative.

The rest of the paper is organized as follows: in Section 2 we present the MicroBridge card game. In Section 3 we give didactical recommendations for using MicroBridge as a microworld for combinatorics. We conclude with a discussion in Section 4.

2. MicroBridge

MicroBridge is a card game for four players, that are divided into two teams. The players are identified by the names East, South, West and North, and the two teams are North-South and East-West. The game takes place around a squared table, where the players are seated according to the following diagram:



As the game starts, the 52 cards are shuffled, and dealt among the four players so that each player holds a set of 13 cards. In general, the game consists of 13 rounds called "tricks". In each trick, each player puts one card on the table, and according to a certain evaluation of these cards, the winner of the trick is determined. The four cards that participated in the trick are then removed from the table, and they do not participate in any of the following tricks.

In Bridge, the game begins in a stage called "Bidding". This stage ends when the players agree on some "Contract". In MicroBridge, since the bidding stage has no relevance to our combinatorical microworld, we omit the actual bidding, and assume that the teacher supplies the contract directly to the players.

The contract has three roles in the game:
1. It identifies the players with their positions around the table.
2. It determines the minimal number of tricks that North-South must win in order to win the game.
3. It defines the "trump", that is, a suit which is considered the most valuable suit of the game. Possibly, the contract may define that all the suits are equally valuable. The "trump" of the game may be one of the following: Spades, Hearts, Diamonds, Clubs, or No-trump. (The role of the trump suit is explained later.)

As an example, a contract may be:
1 - Ron = South; Don = North; Ronna = West; Donna = East;
2 - North-South (in this case Ron and Don) must win at least 10 tricks.
3 - The trump is Hearts.

We now describe how the tricks of the game proceed. In each trick each player puts one card on the table. The players put their cards one after the other, in a clockwise order. The player who wins the last played trick, is the first to put a card in the next trick. (Conventionally, West plays first in the first trick.)

When choosing a card to play, a player must obey certain rules:
The first player in a trick may put any card. This first card is called the "lead" of the trick, and its suit is called the "suit of the trick". The other players, in their turn, must put a card from the suit of the trick, unless they have no such card. In this case, they can put any card (no cheating is accepted!).

After the four cards of the trick have been placed on the table, the winner of the trick is determined. This is done according to the following rule: if there are no trumps on the table, then the winner is the highest card in the suit of the trick. Otherwise, the winner is the highest trump.

So far, we have described the game as if the players conceal their cards throughout the game. However, this is not true for all four players. At the beginning of the game, just after West has put the first card of the first trick, North's cards are revealed by laying them on the table, so that the rest of the players can see them. From that moment on, North serves as a "Dummy" player, and South is actually the one that decides which cards to put for both North and South. The role of the Dummy in the game is just to obey South, and to put exactly the card that South orders. Thus, South is the player who tries to win the game, by winning at least the number of tricks that are specified in the contract. Consequently, South is called the "Declarer" of the game, and West-East, who try to prevent South from achieving the specified goal, are called "Defenders".

As an example, assume that the players receive the following sets of cards at the beginning of the game, and that the contract given to the players is the one specified above.

Notrh (Dummy)
Spades 7 6 4 3
Hearts K Q 3 2
Dianonds A Q
Clubs 10 7 4

West (Defender)
Spades Q J 10 9
Hearts 10 6 5 4
Diamonds 2
Clubs J 9 8 6

East (Defender)
Spades A K 8 5
Hearts -
Diamonds K 9 8 7 5 3
Clubs 5 3 2

South (Declarer)
Spades 2
Hearts A J 9 8 7
Diamonds J 10 6 4
Clubs A K Q

The first trick starts when West puts the Q of Spades, for example, on the table. (Thus, Spades is the suit of the trick.) Now, North's (the Dummy's) cards are revealed on the table, and South asks North to put a

card, for example, the 3 of Spades. (Recall that since North has Spades, the card that South chooses to put must be a Spade.) East, holding the K and A of Spades, knows that South does not hold a higher Spade than the Q, and unless South puts a trump, West will win the trick. Therefore, East chooses to put the 5 of Spades. South, who holds only the 2 of Spades, must put it. Since all the cards of the trick are of the suit of the trick (Spades), the highest card wins. In this case, West wins the trick with the Q of Spades, and proceeds by playing first in the second trick.

In the second trick, West chooses to lead the J of Spades, South asks North to put the 4 of Spades, East puts the 8 of Spades, and South, who does not hold any Spades, may put the 9 of Hearts. Since Hearts is the trump suit, South wins the second trick.

Looking at North's cards on the table, South counts that together they hold eight Hearts. Adding the 9 of Hearts that was put in the second trick, South realizes that the defenders hold together exactly four Hearts. South knows, that the distribution of the Hearts between the defenders can be one of the following: four Hearts to West and none to East, three Hearts to West and one to East, etc. Using this and similar counting techniques, South tries to plan the tricks in a way that will guarantee winning at least ten tricks in the game, as specified by the contract.

3. Didactical Recommendations

In this section, we suggest a way of using MicroBridge for combinatorics. Our suggestion consists of three stages: entering the MicroBridge world; combinatorics in the MicroBridge world; and stepping outside the MicroBridge world.

3.1 Stage 1: Entering the MicroBridge world

The first stage is an introduction to MicroBridge. The teacher presents the game and its rules to the students, and then, in order to exercise the rules of the game, the students spend a certain period of time playing.

A recommandation for conducting the first stage is to let the students play "home-made" games. That is, instead of letting the students deal their cards freely, the teacher prepares the games in advance. Moreover, it is recommended that the class will play the same game simultaneously. Playing "home-made" games simultaneously has two significant merrits: First, since all the students play the same game, the class can discuss each game right after it is played. Second, and more important, by using "home-made" games, the teacher can encourage the development of the combinatorical thinking towards the desired direction. This is achieved by creating situations in which the combinatorical questions arise naturally from the game.

3.2 Stage 2: Combinatorics in the MicroBridge world

The second stage consists of a well chosen set of assignments within MicroBridge. In this stage, the students continue to play MicroBridge, however, combinatorics, and not the game itself, is the main focus. The students play "home-made" games that are oriented to deal with specific combinatorical topics. After the game (or games), the discussion in class is conducted on the combinatorical questions that arise from the game, rather than the game itself.

In general, the core of the combinatorical world is the process of counting. Usually, when posing a question in combinatorics, one describes a real life situation, in which it is required to count the number of permutations, variations, combinations, etc.

In MicroBridge, as in combinatorics, counting plays a major role. Players often apply counting considerations in order to decide their next move. For example, it is highly important to count how many cards remain in the game at any stage, and then, to try and guess the distribution of the remaining cards among the players. Looking at the subjects of a traditional curriculum for combinatorics, one may notice that it starts with

permutations, continues with variations, and then concludes with combinations. From these three subjects, combinations is the crux of combinatorics, and permutations and variations stand as its special cases. However, since combinations are considered harder, they are usually taught last. In MicroBridge, we can skip permutations and variations and concentrate on the main issue, combinations, almost from the beginning. This idea of focusing on the main issue right from the start, resembles the way in which working with Logo enables us to focus on major methodologies of computer science (like abstraction or modularity,) right from the start, contrary to traditional teaching (through Pascal, for example).

As was previously mentioned, the emphasis of the second stage is to give rise to combinatorical questions through MicroBridge games. Thus, we suggest that the structure of a typical lesson in the second stage will consist of: First, letting the students play "home-made" games that concentrate on a specific topic. Second, discuss the problems that arise from the game (still in the game level). Then, the teacher introduces assignments that the class solves together, and finally, the teacher formalizes the problem. This follows the Constructivism philosophy, in which the formalism comes at the end.

Following is a representative sample of assignments. Naturally, each of the assignments is given to the students after they have played several games that emphasize the need for solving these assignments.

Assignment 1:
Assume you are South. You hold A K Q J 10 of Spades. In how many different orders can you play your Spades?

Assignment 2:
Assume you are South. You hold A K Q J 10 8 of Hearts, which is trump, North holds no Hearts. No Hearts have been played so far. Now, you plan to play the A in the first trick, the K in the second, etc. Find the number of possible distributions of the rest of the Hearts among East and West which will guarantee that you will win all six tricks in Hearts?

Assignment 3:
Assume you are West, and you hold the 9 7 6 3 2 of Hearts. Moreover, assume you see on the table that North holds A K Q of Hearts. Now you realize, that North (or actually South) is planning to play the A K Q of Hearts in the next three tricks. In how many different ways can you throw three Hearts from your hand in these tricks?

Notice, that each of the above examples hides a well defined combinatorical question. In the first asignment, the question is to find the number of permutations of five elements. In the second assignment, the question (which is not trivial) is to find the number of partitions of the set 2 3 4 5 6 7 9 (the remaining Hearts) into two boxes (West and East), in a way that the box, to which 9 is inserted, contains no more than five elements. In the third assignment, the question is to find the number of variations of three elements out of five.

3.3 Stage 3: Stepping outside the MicroBridge world

The didactical process is completed by stepping out of the MicroBridge world. The two major goals in this stage are:
1. Extending terminology and formalism.
2. Expanding the combinatorical thinking of the students to other areas.

It is obvious that whatever method of teaching combinatorics is used, the student must be familiar with all the terminology and formalism in the area. Thus, in the third stage, we complete and generalize the formalism that was introduced in the second stage. Naturally, this stage may still use the MicroBridge world for examples, though in this stage it is done without actually playing the game.

In the first and the second stages, we try to motivate the students towards combinatorical thinking. However, this is done only in the narrow aspect of the MicroBridge world. Now, we must perform a "transfer"; the students will learn to apply the combinatorical knowledge they acquired to other areas of life. As a result,

the students will obtain an abstraction of the combinatorical thinking. Needless to say, this stage of the didactical process is crucial.

4. Conclusions

We introduced the MicroBridge game, which is a simplified variant of Bridge. We suggested that MicroBridge can serve as a microworld for combinatorics. The didactical recommendation of employing Microbridge as a combinatorical microworld starts with teaching the game to the studens. After the students are familiar with the rules of the game, they acquire, guided by the teacher, certain techniques and strategies that are essential for the game. These strategies encourage the students to employ "combninatorical thinking", which in turn, is used by the teacher as the basis for extending the combinatorical knowledge of the students. An additional stage in the didactical process, consists of stepping out from the MicroBridge world, and proving to the students that the same kind of combinatorical situations, may arise in other areas as well.

By utilizing an attractive and popular game as the basis for our combinatorical microworld, we believe that the students will be enthusiastic and most cooperative. Moreover, students who start to play Bridge (or MicroBridge) in school, will probably continue to play after school hours, and when they grow older. Thus, using the suggested microworld, will not only make them good combinatorics learners during the school period, but also life-long learners.

Students are often having difficulties in developing combinatorical thinking. The traditional way of teaching, tries to demonstrate combinatorics through the called "real-life" problems. Allas, these problems are artificial and forced. We believe MicroBridge poses the combinatorical questions naturally, encourges an intuitive approach to the field and therefore is excellent for the development of the combinatorical thinking.

Another advantage of using MicroBridge as a microworld, is that it can be very naturally extended to the teaching of Probability Theory. Roughly speaking, choosing the best strategy in a certain MicroBridge game, can be translated into choosing the strategy with the best probability to succeed.

Aknowoledgements
We wish to thank Orit Hazzan, Dalit Levy, Tal Rabin and Ophir Rachman for their careful reading and helpful remarks.

Bibliography

[1] Papert S. Microworld: Transforming Education. In Lawler, R.W., Yazdani, M. (eds). *Artificial Intelligence and Education*, Vol. 1, Ablex Publishing Co., pp. 79-94.

Analysis of the Musical Domain in the Logo Environment

Maria Cecília Martins

Center of Computer Science Applied to Education-NIED
State University of Campinas-UNICAMP
Campinas, S.P - Brazil

Abstract

The purpose of this paper is to survey and analyse the aspects involved in a computer environment in the musical domain. This paper addresses what is involved in the learning environment and also the individual performance dynamic in these idealized surroundings. The analysis is based on a workshop where the music-Logo is explored, with basis on what has been observed during some moments of the individual performance in this context. The results of this experiment discloses the moments of the individual involvement in the creative process and point out the need for developing new resources in the music-Logo program.

Keywords: musical learning, musical composition, Logo, programming, sounds

Analysis of The Musical Domain in the Logo Environment

1. Preliminary Considerations

When observing the work done in the ambient of a Logo program, the intense interest of the student to act in the musical domain with the use of the computer is clear, however, the difficulties faced when exploring the sonorous resources offered by the language are clearly observed.

There are various aspects to be considered in relation to the use of the computer in music. In this paper however, the focus of the analysis is directed to the musical composition, here considered as an activity in which the individual performs some musical development through a process of "materialization" and transformation of his ideas.

The experiments allowing musical creation in the music-Logo context is more interesting as the affinity of the Logo philosophy to musical learning becomes more evident.

Both contexts have the same supposition, that is, that knowledge is acquired through the individual interaction in the environment in which operates.

The work in the musical domain can be highly advantageous if the Logo language is conceived as a programming language apt to promote a description, depuration and pondering process, besides being a certain kind of learning ambient. Thus, the work with music and Logo leads us to a deeper reflection overcoming the considerations about the computing possibilities, since the environment conception has a relation with the musical composition conception as well as with learning as a whole.

This paper has the specific objective of pointing out and discussing the possibilities and limitations of Logo language for the performance of the individual and for establishing the musical concepts by the individual in this context. In order to illustrate these aspects, a music-Logo workshop has been developed by UNICAMP Center of Informatics Applied to Education.

The workshop established a proposal to increment the Logo language with commands to facilitate the development of activities in the musical domain by persons who know the Logo language without necessarily having musical experience.

The proposed activities in this workshop had the objective of allowing persons to use the computer as an auxiliary tool in the musical composition process, being exposed to and reflecting about the musical concepts involved in the context offered by this language.

However, before describing the workshop we shall go over the elements comprising the setting in which the experiment was developed.

2. The Setting of the Experiment

The setting where the activities with music-Logo takes place includes the individual who operates in a given computing surrounding and uses a programming language.

The work of this individual is set on the musical domain. However, the development of the knowledge

involves other domains such as, for example, computation.

In this working context, the individual interaction takes place both in relation to other individuals (students and constructors) and in relation to his own thinking as well.

In order to place the elements involved in the setting, it is important to consider what Papert (1987) quite properly said about the computer and other technologies in our lives, in a conscious assessment of what "we will make with the computer" and not what "the computer will make to us". Thus, we can better think how to recompose the world on the face of the technology. And as a consequence, we shall also think what "learning" is in order to, under this conception, rethink education in itself.

Papert, addressing the learning-teaching environments comments that, even before the birth of the computer and its use in education areas, educators had different views in relation to the objectives of education, defining it as either the learning of facts and skills or as a process for the development of individuals. The role of the computer today brings back this discussion on the conceptions related to teaching-learning.

In relation to learning, some theoretical conceptions point to the active interaction in which the individual is led to experiment, adjust and modify the object of learning, analyzing his own action. In order to achieve the above, there are necessary activities that stress action, reflection and communication about the acquired knowledge and the environments that promote such activities.

Therefore, learning environments are characterized as those offering the occurrence of a dynamic action between the making and the reflection about making, and the communication to oneself and to others the experience of making.

This dynamic action - which helps the student to acquire his knowledge through manipulation of objects of the learning - gives the student the opportunity for developing his own theories and investigations, besides allowing him to explore, explain and describe such theorizing subjects as well as informing others about them.

When preparing description of his explorations, the student is assisted on his discoveries about the world, since, while maintaining the records of his discoveries, making them more clear to himself and to others, he develops and acquires knowledge. In the activity of making and describing what he does, the student can gain knowledge about himself and about the environment in which he performs.

In this context, the role of the educator or cobuilder of knowledge will be to provide significant environments to the student. These surroundings should make the contact with the intended knowledge easier, offering situations having problems in which he could be involved during the process over such situations. The educator should also be involved in these experiments.

2.1 Musical Learning and Computer based Environment

In relation to musical learning, there are alternative educational approaches that allow the individuals an effective performance on the musical composition.

A proposal that exemplifies this educational approach is the situation presented to the student, in which he develops a musical composition starting from sounds of nature or from the daily life. For the development of such compositions are provided activities in which the student must:

- explore various sound sources or different ways to produce sounds, for example, from objects in the room.
- select sounds generated on his initial work and record them.
- make graphic representation that characterizes the chosen sonorous elements.
- organize the selected material.
- produce and analyze the created material

With this approach, the music is conceived as something to be tried and developed by individual actions. "Learning by doing" with a wide array of tools through activities that assure retrieval, description and representation of the musical elements experienced is stressed.

Papert reminds us that learning is reconstruction, not transmission of knowledge. Thus, learning is effective when immerse in the activity in which the student tries, by producing, modifying and thinking about a learning object that is relevant to him. At this point, an opportune tool comes to our mind - the computer - and specially the Logo environment.

To establish the additional uses that the computer may have in relation to other available tools for musical learning is of utmost importance to provide the additional required elements in the Logo language in the musical context.

The computer, as considered in this work, is seen as a machine that allows the creation, handling and symbolic transmission of objects. Through the computer, the individual participates in a cycle of actions and their feedback. The individual, with this tool, models the object of his understanding and with basis on the generated feedback , reviews and depurates his ideas.

The computer is an instrument that displays a double function: one is to promote the involvement of the individual in the construction of objects of knowledge or ideas, leading him to share in this construction, and the other is to make possible to observe the modification of the object of knowledge as a function of the individual and of the individual as a function of the object.

The computer may permit to "visualize" the construction process, in this particular case, the construction of musical "objects" or ideas, since it is part of a favorable environment.

The possibility - provided by the computer - of retrieving the composition process permits the individual to establish the way developed for the design of an idea. On the retrieval of this process, the computerised environment experienced allows the interested party to "shape up his ideas" and expand his perspectives about the subject of his experience through an analysis of his actions.

2.2 Setting up the Computing Environment on the face of the Programmed Activity

It is important to think about the development of the environment for programming, especially when it is intended to offer, to the individuals, operation in domains which are not familiar to them. In this context, the concern is how to design computing tools adequate to the use and purpose of the individual, that is, the concern for the creation of tools which allow the individual to be the conductor of his process of acquisition of the musical language as vehicle of expression.

Considering the production of computing systems for the musical field, we see, generally, that there are a variety of computer tools for the musical domain intended to allow access of professionals to the resources of the computer in music and very little has been considered in relation to the needs of the student in this area. The Logo language, due to its educational format - centered in the construction of knowledge - appears as a propitious way to the beginner. There is in this language a command that allows the emission of sounds, however, since it is not directed to specific musical purpose, it offers some difficulties to individuals operating in the musical domain.

The identification of this difficulty led us to take steps to enrich the Logo ambient within the musical domain.

With this in mind, it was tried to analyse the activities of the beginner in the music-Logo environment as a strategy to find out what type of difficulty takes place, how they are overcome and what type of tools could help them in the process of using this language as a means for representation and generation of musical ideas.

Two other significant aspects to be considered for the expansion of the music-Logo environment are the

musical concepts to be offered for use and the actions involved in this process of developing knowledge. Actions such as hearing, executing, improvising, recuperating, analyzing, depurating, ordinating, are part of the process for the development of musical concepts, being all of them actions offered by the computer.

Understanding the actions of the individual in the music-Logo environment depends upon the perspective under which we see the activity of programming.

Baranauskas (1993, p.3), addressing the activity of programming, stresses that "the knowledge of the programming language in itself, in its semantic and syntax aspect, does not exist isolated or a priori of the knowledge of how to "express" a given problem in that language". In Baranauskas point of view, "the knowledge of the language in its semantic and syntax aspects and the knowledge of the manner how to express/represent a problem with this language are developed together, along a process". This author uses the expression "appropriation of the programming language" to represent the solution of difficulties to suggest the indissociability between knowledge of the language and knowledge in how to express and represent through it.

With regard to this indissociability of language and its use in the context of the musical learning-teaching, it is inferred that the knowledge of the subject (music) and the language itself (Logo) are built together, along the interacting process of the individual in the environment of musical learning.

Other conceptions, in which the knowledge of a language is acquired isolated of its use, are equivalent to existing educational belief in which it is accepted that is sufficient for a student to know a language to be a "composer" or that the knowledge of a language would be a "prerequisite" for a beginner to initiate composing activities.

After having reviewed the elements involved in the conceptions that comprise the educational environment of experimentation based on the computer, we shall go next to the description of the experiments that allowed a reflection on the use of the Logo language in music.

3. Workshop Music-logo

The music-Logo workshop here described was set up in 1990 and was a part of an assembly of other workshops with the scope of providing an analysis of powerful processes for description, depuration and reflection through different experiments in the Logo environment, such as Lego-Logo, tridimensional Logo, listing-Logo.

12 Educative Informatic professionals with experience in Logo, both in programming and in the use of the language for education, but not necessarily in music, participated in the music-Logo workshop..

The supporting material used in the workshop consisted of one written instruction and a mini-software. The mini-software was developed with basis on the primitive PLAY that operates with the computer sound manager.

The mini-software was implemented to offer greater musical flexibility, allowing the person to program small musical pieces and apply some variations on them. The mini-software was composed of musical notes, commands and musical operations.

The existing musical commands would allow the production of:

- a musical note with a given duration, as for example:

```
PLAY.NOTE    [A1  50];
```

- a sequence of notes, as for example:

```
PLAY.PHRASE  [[G1  10]  [E1  10]  [D1  10]  [C1  20]];
```

- or, in addition, a sequence of a chain previously stored in the memory, as for example:

```
PLAY.PHRASES [phrase3  phrase1].
```

The musical operations allowed handling of listings of notes stored in the memory. From the stand point of composition, the operations are mechanisms that allow people to work dynamically in a set of musical information to achieve a musical development.

One example of operation available is the inversion, that consists in recompose one phrase in the inverted order in which was composed. This possibility doubles the number of phrases available for selection, with no need to add any "new" musical element, as for example:

```
PLAY.PHRASES [phrase4  phrase3] PLAY.PHRASE INVERT [phrase4]
```

3.1 Results

In regard to the use of the proposed material, we found that all participants, independently of having more experience in programming or in music, developed various composicional exploratory ideas. All of them used suggestions of activity from the support material, some having basically used the resources offered by the mini-software and other having expanded them.

The analysis of the activities allows us to pinpoint the aspects of the experience that disclose the dynamic of the exploratory process followed by the individuals.

Activities centered in the initial exploration of the material.

At the start of the workshop, the participants were at the exploratory level of the mini-software, the major interest being to find out how the available commands operated in this context. In this phase, most participants explored different developments of sonic sequences, using musical phrases available in the material supplied. Setting of musical phrases in the material aimed to have the participants to position themselves in the context and, with these phrases, start an initial exploratory move, applying transformation upon such sound sequences, and generating different groupings between phrases.

It can be stated that, in this phase, the participants were allowing themselves an auditive disconditioning, with elements from the individual and with his freedom to choose, without requirements or determination of "prerequisite" therefore minimizing the inhibition to act in this domain of the knowledge. The important point to be stressed in regard to what takes place at this moment is that, in the act of exploring the sonic possibilities, regardless of the final product, it is of significant importance to the person who produced it, as part of an acting process of the individual in this domain.

Activities centered in the "translation" of music to the proposed environment.

Another moment that reflected the dynamics of the exploratory process was in the interest of the participants being able to reproduce through the computer a known tune, specifying their notes and the respective duration.

During this phase, the participants displaying better control of the "translation" activity, were those with some musical knowledge; the ones with more difficulties were those who did not have any knowledge. To reproduce a tune in this new context, the participants should have some notion of the sonic sequences, sound intervals and their duration. In this case, the major work in this activity was centered in "translating" the data in terms of programming language. This approach was effective for those who had the necessary "prerequisite". In the case of a newcomer in music, the environment should supply greater feedback so that his performance in the activity were not interrupted.

In a subsequent moment, those who were able to achieve the "translation" of a known melody, reused the

elements of the composition in a differentiated form, in accordance with their own criteria. For these users, the "translation" was the entrance gate for the acquisition of the music-Logo environment.
Activities centered in the development of their own musical ideas.

Another moment of interest experienced both by individuals with musical knowledge and by those without this knowledge was characterized by activities in which the individual was trying to develop his own musical idea.

What is interesting to point out is that their effort reflects an action of the individual to expose himself musically, risking the search and elaboration of a musical idea through the application of contrasts, variations and repetition of sonorous elements. The process of elaborating musical ideas developed by these students show that the action of composing, more than a process with support from the auditory perception, is a process that involves activities of description, depuration and reflection, which are actions inherent to the programming activity.

4. Final Considerations

This pilot experience offers an example of how a computer, together with a language and available operations (despite still be offering insufficient assistance, requiring improvement in various aspects), appears as a tool that helps the student to operate musically in this context.

The importance of the computer is due to the fact that it is a tool with characteristics that permit the implementation of ideas and allow these ideas to be retrieved and depurated by the student. The computer, set in educational surroundings and allowing experimentation, becomes a tool that permits the involvement of the individual in the construction and in sharing objects/ideas.

An overall analysis on the development of this experience points out the need to add new resources on the environment of programming, as we will discuss as follows:

The results pointed out the need for different forms to represent what is being performed, in relation to both the musical and computation aspects. The textual representation of the music (with Logo words) did not appear to be enough as a feedback from the composition elaborated. Thus, the need for a graphic representation attending the typical characteristics of the musical activity to help the exploratory action on the attributes of the sonic events.

On the other hand, the music-Logo environment showed the possibility of approaching computing concepts through the music. Also showed to be an adequate environment to equip people to think about programming.

Finalizing this paper, we think that educational approaches in music with the use of computers require an environment with special characteristics in regard to the communication interface with the computer environment, with sound output of good quality, with adequate graphic representations for the sonorous events and a computer language in addition to the commands directed to music.

REFERENCES

BARANAUSKAS, M.C.C. (1993). *"Criação de Ferramentas para o ambiente PROLOG e o acesso de novatos ao paradigma da Programação em Lógica"*, PHD Thesis, Electrical Engineering Faculty, State University of Campinas

GARGARIAN, G. (1990). *"A Logo Microworld on Musical Form"*.

PAPERT, S. (1980). *"Mindstorms: Children, computers and powerful ideas"*. New York

PAPERT, S. (1987). "A critique of technocentrism in thinking about the school of the future. Conference presented at *Children in an Information Age: Opportunities for Creativity, Innovation & New Activities*, Sofia, Bulgaria.

Discovering Angles with Logo

Christos Minopoulos[1], Tefcros Michaelides[2], Alkistis Patrineli[3]

[1,3] Ziridis School
45 Kifissias av.,
151 23 Marousi, Greece,
tel: 6898720-24, fax: 6898732

[2] Athens College
2 Iconomou Street
115 25 Athens, Greece
tel: 6714621, fax: 6478457

Abstract

The paper presents a method of initiating young students to fundamental geometric concepts such as angles and their operations using LOGO. The method is actually applied experimentally in classes of children from 11 to 13at Ziridis School, Athens. Worksheets of specific activities are included.

Keywords: angle, geometry, Logo

Christos Minopoulos *was born in 1967 and graduated from the University of Crete, Department of Mathematics in 1989. Teaches Computer Science at Ziridis School since 1992.*

Tefcros Michaelides *was born in 1954 and graduated from the University of Paris VI, Department of Mathematics in 1977. Obtained a Doctorate in Number Theory from the same University in 1980. Teaches Mathematics at Athens College since 1981. Author of four school manuals on Mathematics and Computer Science.*

Alkistis Patrineli *was born in 1965 and graduated from the University of Athens, Department fo Mathematics in 1991. From 1990-1992 worked as a software developer in Unisys Hellas S.A. Teaches Computer Science at Ziridis School and Campus Arts and Siences College since 1991. Is a member of the Greek Mathematical Society and of the Scientific committee of Informatics of OEEC.*

Discovering Angles with Logo

Euclidian geometry has been the "persona non grata" of mathematical education for the last thirty years. Ever since the famous revolution of the so called "Modern Mathematics" the role of Geometry in education is more and more minimised. The principle of teaching "useful" mathematics, presented in the form of dogmatic revelation in view of their future applications, progressively substitutes theories permitting a heuristic approach and stimulating trial and error, experimentation and deductive reasoning. On the other hand the voices of eminent mathematicians and educators underline the error of this process pointing out that Euclidian Geometry is unique in providing a non trivial learning object, constantly referring to an underlying intuition, rich in examples and problems and adequate to introduce strict reasoning, and initiate young students to axiomatic methods [4].

Moreover, the recent technological revolution gives us powerful new tools for creating and exploring rich interactive environments which may help more students to experience the joy of discovery as compared to the boredom of memorisation and blind application.

Using LOGO as our main tool we have organised a series of activities which help the students discover fundamental geometric concepts. The activities presented here are actually applied experimentally by the authors, to classes of Demotic 6 (equivalent to grade 6, 11-12 years old) having a good LOGO background and to classes of Gymnasium 1 (equivalent to grade 7, 12-13 years old) who have not taken any other courses in LOGO. The whole experiment takes place at Ziridis School, Athens. An hour per week is dedicated to these activities independently of the main school curriculum.

The centre of the activities included in this presentation is the concept of "angle". The first activity is organised in class without using a computer. The other two activities are carried out in the computer lab. Throughout the activities the students are guided by worksheets which direct them and help them draw the necessary conclusions. (It is a general conclusion that students have a tendency to treat more seriously activities in which a written report is required than activities in which all their work is corrected and evaluated through the computer).

The setting of the problem is the following: Students are requested to assist a tourist office in drawing on a map the itinerary to be followed by a helicopter on a sightseeing tour. Children are provided with a map indicating the positions of the departure point (Toy City) and all the interesting places to visit. (see worksheet number 1). They also have a protractor drawn on a transparency (their "path finder") which they use to measure direction angles.

During the first session students get acquainted with the notion of angle which they conceive in three different ways: as a classical Euclidian angle, as a rotation and as a path indicator. They also have an intuitive approach to the angle bisector as well as to angle addition and subtraction.

During the second session the students are asked to create the same map on the computer, using LOGO. Using the classic LOGO commands RT and LT students practise in angle addition and subtraction and get a more concrete notion of the bisector.

During the third session the students are asked to follow the reverse path and take the turtle back to its starting point. In this session students get acquainted with the notions of supplementary angles and get a first intuitive view of modulo arithmetic. During the sessions students use also a small LOGO programme, prepared by the authors to facilitate their task.

In the following sessions, not included in this presentation for space reasons, students are lead to discover the sum of the angles of a triangle, of a quadrilateral and finally of a polygon with n sides.

Each worksheet is followed by exercises for drill and practice and some questions aimed to introduce more gifted kids to additional concepts.

The sessions described above and the related worksheets that follow are just an example of the way we are working. Similar activities have been organised to investigate congruent triangles, properties of similar triangles and the Pythagoras theorem.

We are actually waiting for feedback as to how students having followed these activities respond when they are first formally initiated to Geometry in higher classes.

References

[1] Abelson Harold: *Turtle Geometry*, MIT Press,1986

[2] Clemens Herbert & Clemens Michael: *Geometry for the Classroom*, Springer Verlag, New York, 1991

[3] Seymour Papert (1980), *Mind-Storms, Children, Computers and Powerful Ideas*, Basic Books, New York, (Greek Edition: *"Νοητικές Θύελλες"*, 1991, εκδ. Οδυσσέας)

[4] Thom René: Les Mathematiques "Modernes". Une erreur pedagogique et philosophique? , in *"Pourquoi la mathematique?"*,10/18, Paris 1974

Activity 1

Introduction

Your **path finder** is a circular disk divided into 360 equal parts, numbered from 0º to 360º. Each subdivision is called a **degree**. A small º to the top left of a number stands for "degree". The number corresponding to a subdivision will be called a **reading**. Each reading determines a **direction** on the path finder. Several readings have special names:

0º correspond to NORTH (N)

90º correspond to EAST (E)

180º correspond to _____

270º correspond to _____

What is the reading corresponding to NORTHEAST (NE) ? _____
What is the reading corresponding to NORTHWEST (NW) ? _____
What is the reading corresponding to SOUTHEAST (SE) ? _____

360º correspond to _____

The problem

A tourist office in Toy City, the capital of Pleasure land, is organising sightseeing tours, using a helicopter.. A map of the area around Toy City showing the principal monuments is provided in next page.

The pilot needs a good method for planning and drawing his trip on the map.

The first monument to visit is a mediaeval monastery due 40 miles northeast (NE) of Toy city. On your map, draw the line joining Toy City to the mediaeval monastery. Also draw a line from Toy City to the North. Place your path finder on the map with its centre on Toy City identifying the North line of the path finder to the North line you have just drawn.

What is the reading on the line going to the monastery? _____

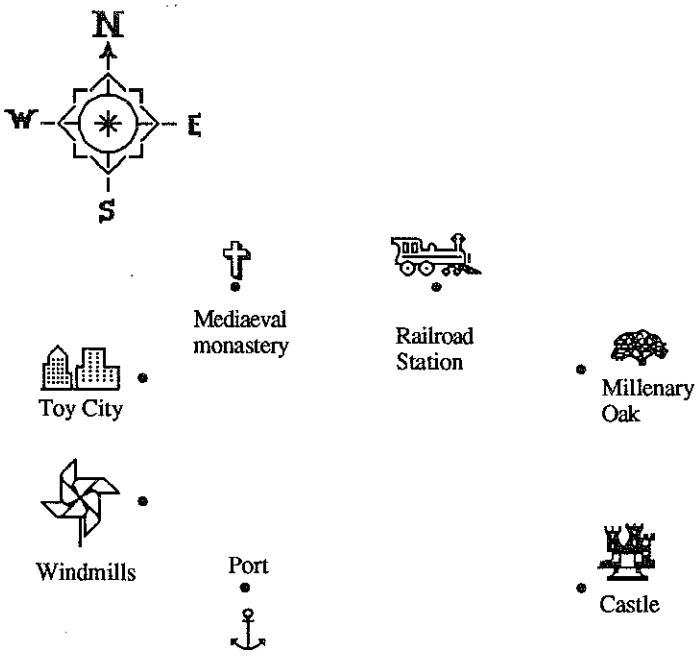
The reading on the path finder is called a 45º **course**.

The plane figure formed by the two lines drawn is called a 45º angle.

On our way from Toy City to the monastery at a distance of 20 miles from Toy city there is an old farmhouse not mentioned on the map. Mark its position on your map.
To reach the mediaeval monastery we follow a ____ course for a distance of ____ miles.

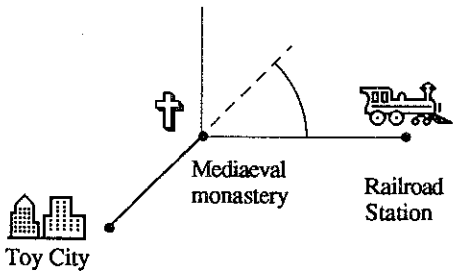
To reach the old farmhouse we follow a ____ course for a distance of ____ miles.
Draw a line from the monastery towards North and then join the monastery to the railroad station. Use your path finder as before to determine the course to follow from the monastery to the railroad station.
It is a _____ course. The two lines drawn form a _____ angle. The distance from the monastery to the railroad station is 60 miles.

Continue in the same way, each time drawing a North line and joining each spot to the next one. Your path finder will give you each time the reading corresponding to the course to follow and to the angle formed.



Departure from	Arrival at	Course	Angle
Toy City	Monastery	45	45
Monastery	Railroad station		
Railroad station	Millenary oak		
Millenary oak	Castle		
Castle	Port		
Port	Windmills		
Windmills	Toy City		

Now produce the line joining Toy city to the monastery. This line, together with the line joining the monastery to the railroad station form an angle. You may measure this angle by putting the centre of your path finder on the monastery, identifying the north line of your path finder to the line produced and getting the reading of the railroad station. The angle measures _____. This new measurement indicates the turn that the helicopter should make if it is heading towards the monastery in order to assume a new course heading towards the railroad station.



Repeat the same exercise at the railroad station. If the helicopter is heading towards the railroad station what is the turn it should make when it arrives there in order to assume the new course heading to the millenary oak? _____

Repeat the same operation at each new spot of the map and complete the table:

Arrivng to	New destination	Old Course	New Course	Turn
Monastery	Station	45	90	45
Station	Oak Tree			
Oak Tree	Castle			
Caslte	Port			
Port	Windmills			
Windmills	Toy City			

Could you find the angle to turn, without measuring, knowing the old and the new course of the helicopter?

_____

Exercises

1. A pilot flying from Athens to Ioannina is following a course of 270º. After several miles of flight he is turning clockwise 75º. What is his new course?

_____

2. A pilot flying from Athens to London. After several miles of flight he is turning clockwise 50º. His new course is 85º?

_____

Activity 2

In this activity we are going to use LOGO to draw the sightseeing map studied during the previous activity.

Load LOGO and prepare to draw with the turtle at HOME and your screen clear.

Each turtle step will represent one mile. The initial HOME position of the turtle will be Toy city. The turtle is initially heading North. In order to turn towards Northeast we must give the command RT 45. Determine the turn needed to assume each one of the following courses:

From North to East RT _____. From North to Southeast RT _____
From North to South RT _____. From North to Southwest RT _____
From North to West RT _____. From North to Northwest RT _____

From Northeast to East RT _____. From East to Southeast RT _____
From Southeast to South RT _____. From South to Southwest RT _____
From Southwest to West RT _____. From West to Northwest RT _____
From Northwest to North RT _____.

Therefore to turn from North to East we may either turn once RT _____ or take two successive turns of RT _____ and again RT _____.

Now let us draw the map. Clear the screen and put the turtle at HOME (Toy city).

Northeast of Toy city at a distance of 40 miles is the mediaeval monastery. So make the turtle assume a NE course _____ and advance 40 miles _____.

The railroad station is 60 miles due east of the monastery. To assume an E course _____ and to go to the station _____. Complete the map using the table:

Departure from	Arrival at	Course	Distance	Command to turn
Toy City	Monastery	45	40	RT 45
Monastery	Railroad Station	90	60	RT 45
Railroad Station	Millenary Oak	120	50	
Millenary Oak	Castle	180	65	
Castle	Port	270	100	
Port	Windmills	310	40	
Windmills	Toy City	0	37	

To assume its new course the turtle each time takes a turn equal to_____
The office offers a second tour. The destinations are indicated in the following table. Draw the map:

Departure from	Arrival at	Course	Distance	Command to turn
Toy City	Aritfiacial lake	315	50	RT
Artificial lake	Television Tower	240	70	RT
Television	Tower Airport	120	67	RT
Airport	Toy City	50	50	RT

Give the commands PU HOME PD to make the turtle assume its departure position heading North.

Can we make the turtle head towards the artificial lake with a LEFT (LT) command? ___
Therefore RT 315 has the same effect as LT ____ .

Departure from	Arrival at	Course	Distance	Command to turn right	Command to turn left
Toy City	Aritfiacial lake	315	50		
Artificial lake	Television Tower	240	70		
Television	Tower Airport	120	67		
Airport	Toy City	50	50		

Exercises

1. What is the sum of all the turns that the turtle made in order to complete the round trip from Toy city to the monastery, the railroad station,... and back to Toy City? (Add the numbers you filled in the last column of the first table in this worksheet. _____

What is the sum of all the right turns that the turtle made in order to complete the round trip from Toy city to the artificial lake,... and back to Toy City? (Second table) _____

What is the sum of all the left turns that the turtle made in order to complete the round trip from Toy city to the artificial lake,... and back to Toy City? (Third table) _____

2. Load the file ANGLE1 which contains several additional LOGO commands. Give the command POSITION then the commands FD 90 BK 90 RT 50 FD 90 BK 50 to draw an angle. It is a 50º angle. Now draw a 40º angle. Give the command CIR to draw a circle around the two angles.. Use the FILL command to colour the area occupied by the two angles. This coloured area is a new angle of ___°. Repeat the same process drawing successively angles of 70° and 60°. The new angle is ___°.

Activity 3

With the turtle at Toy city, heading North we give the commands _____
in order to go to the mediaeval monastery. To come back to Toy city, the turtle should make a turn of _____. Observe that such a turn may be done either to the left or to the right with the same effect. Try it!

From any course to assume the return course the turtle must turn by _____ .

Set the turtle at HOME, heading North. Turn it towards NE by RT _____. From this position, which command will turn it towards South? RT_____. The two angles have a sum of _____.

Set the turtle heading West. Give the command LT 30. From this position, which command will turn it towards East? LT_____.The two angles have a sum of _____.

Angles having a sum of 180° are called **supplementary**.

Give the following commands:
RT 50 FD 40 HOME

What angle should you give in the first command to obtain the same result with a left turn? _____
Same question for the following commands:
RT 80 has the same effect as LT _____ RT 120 has the same effect as LT _____
RT 250 has the same effect as LT _____ RT 310 has the same effect as LT _____
LT 80 has the same effect as RT _____ LT 120 has the same effect as RT _____
LT 250 has the same effect as RT _____ LT 310 has the same effect as RT _____

Conclusion: If RT a and LT b have the same effect then a+b=_____
Here is the table you used during the last session for the round trip from Toy city to the monastery etc.

Departure from	Arrival at	Course	Distance	Command to turn
Toy City	Monastery	45	40	RT 45
Monastery	Railroad Station	90	60	RT 45
Railroad Station	Millenary Oak	120	50	
Millenary Oak	Castle	180	65	
Castle	Port	270	100	
Port	Windmills	310	40	
Windmills	Toy City	0	37	

You must now describe the trip the other way round. From Toy city to the windmills then to the port etc. Before the trip starts the turtle must look north.

Departure from	Arrival at	Course	Distance	Command to turn
Toy City	Windmills			
Windmills	Port			
Port	Castle 120			
Castle	Millenary Oak			
Milenary Oak	Railroad station			
Railroad station	Monastery			
Monastery	Toy City			

What is the sum of the angles for the round trip? _____

What will be the sum if we give the turtle a last command to make it look north as it did initially? _____

Exercises

Try to answer the following questions without using the computer. Then test your answers with the turtle.

RT 40 followed by RT 90 is the same as RT _____ or as LT____.
RT 210 followed by RT320 is the same as RT _____ or as LT____.
RT 40 followed by RT 340 is the same as RT _____ or as LT____.
LT 310 followed by LT220 is the same as LT _____ or as RT____.

Each command in the left column has the same effect as a command in the right column. Match them:

RT 50	RT 240
RT 280	LT 570
RT 600	RT 140
RT 500	RT 210
LT 150	RT 410
LT 210	RT 640

Catch Infinity by its Tail: Infinite Sequences in Logo, Revisited

Yehoshafat Shafee Give`on

The Centre for Informatics, Beit Berl College, Kfar Saba,
Israel and also Tel Aviv University, Tel Aviv, Israel
e-mail: shafi@ccsg.tau.ac.il

Abstract

In a previous paper I presented a rather unnatural method for the representation of infinite sequences in Logo, using the idea of streams as implemented in Scheme. In this paper I want to present a simpler, more direct, even though slightly weaker method of working with infinite sequences in Logo. The main idea of this paper is to treat sequences as integral objects studied through their extrinsic properties. That is, in addition to treating sequences as structures determined by the properties of their elements, it is suggested here that students will study such sequences through their relationships to other sequences. As it turns out, this seemingly abstract approach opens the door to an abundance of exploratory activities which becomes concrete through a relatively simple programming method of LOGO functions.

Keywords: Logo, infinite sequences, recursion, microworlds

Yehoshafat Shafee Give'on *graduated in 1963 from The Hebrew University of Jerusalem in Mathematics, Philosophy, and in History and Philosophy of Science. Received M.Sc. degree (1964) and Ph.D. degree (1966) in Communication and computer Sciences from The University of Michigan. Did research in algebraic automata theory during 1965-1967 at Harvard University and during 1967-1969 at Stanford University. In 1969 returned to Israel and joined the staff of the Ben-Gurion University, to establish, together with the late Prof. H. Hanany, the department of Mathematics and Computer Science there. In 1983 went into the field of teacher training in Informatics. In 1986 established the Centre for Informatics in Beit Berl College.*

Catch Infinity by its Tail: Infinite Sequences in Logo, Revisited

Infinite numerical sequences treated as objects

Since in this paper we shall discuss only infinite sequences of numbers, the term "sequence" will mean, always, such a sequence.

Consider constant sequences such as

1, 1, 1, 1, 1, 1, 1, ... or 5, 5, 5, 5, 5, 5, 5, ...

How can we express their structure by means of a reference to other sequences?

For instance, we may use the fact that these sequences are special cases of arithmetic progressions. This leads us to consider the sequence of their adjacent differences. In the case of a constant sequence, the sequence of adjacent differences is always the zero-sequence: 0, 0, 0, 0,.... Or, we may drop the first element of any of the constant sequences, only to find out that the resulting sequence is the given sequence itself.

Obviously, each of these two properties is sufficient for the characterization of the class of constant sequences.

We may reverse the last observation and find out that the specific constant sequence 1, 1, 1, 1,... can be uniquely characterized as the sequence S which is equal to the sequence derived by joining 1 at the head of S itself. In a similar manner, we can treat the arithmetic progressions or the geometric progressions.

The idea is to try to find as many as possible properties of given types of sequences, or of specific sequences, in terms of relationships to other sequences. This approach leads naturally to operators on sequences. In fact, three specific operators were implicitly employed in the above discussion.

First, we have the adjacent difference operator DELTA. If S is any sequence, then DELTA S is the sequence of the differences of adjacent pairs of elements of S.

We also used the operator BUTHEAD that removes the head of the sequence. BUTHEAD S is the sequence derived from S by removing the first element of S. Thus, *all* the constant sequences are characterized by the equation S = BUTHEAD S, and therefore, all the arithmetic progressions can be characterized by the equation DELTA S = BUTHEAD DELTA S. It states that the sequence of adjacent differences of an arithmetic progression is a constant sequence.

Finally, in the characterization of the specific constant sequence of 1-s, we used the HPUT operator which puts a number as a new head onto a given sequence. If S is any sequence and c is any specific number them HPUT c S is the sequence made up of c followed by the elements of S. The characterization of the specific constant sequence of 1-s can be expressed now by the equation S = HPUT 1 S.

The operator HPUT has the following important property. For all numbers a_1 and a_2 and for all sequences S_1 and S_2:

If HPUT a_1 S_1 = HPUT a_2 S_2 then $a_1 = a_2$ and $S_1 = S_2$.

The analogy with lists is intended and obvious. This analogy calls for another operator that applies to sequences but has values in the domain of the elements of the sequences.

Let HEAD be the operator that yields the first element of a given sequence. That is, if S is any sequence, then HEAD S is the first element of S.

We may add a constant number c to a sequence S, by adding c to every element of S. Denote this operator by ADD and its result by ADD c S and also by c+S. We may add two given sequences by adding their corresponding elements. Denote this operator by ADDSEQ and its result by ADDSEQ S_1 S_2 and also by S_1+S_2. Their multiplicative and subtractive counterparts will be denoted by MULT, MULTSEQ, DIF and DIFSEQ and their results, in a corresponding manner, MULT c S or c*S, MULTSEQ S_1 S_2 or S_1 *$S2_2$, DIF S c or S-c, and DIFSEQ S_1 S_2 or S_1 -S_2 .

This host of operators provides a rich setup for various exploratory activities concerning such sequences. They can be employed in formulating a variety of characterizations of the classes of sequences known to the students: the constant sequences, the periodic sequences, the ultimately periodic sequences, the arithmetical and the geometrical progressions and their arithmetic compositions such as the Fibonacci series. Searching for characterizations of classes of sequences in terms of operator equations opens the door to the algebraic study of logical relationships that may hold between such equations.

Now we may try to characterize other classes of sequences. This can be done by choosing interesting equations and by asking what are the classes of sequences that are determined by them. This can be done also by restricting the types of operators used in the definitions of sequences to certain specific sets.

For example, what is the class of sequences that is defined by equations using only the BUTHEAD and HPUT (with any numerical constant) operators? What is the class of sequences that is defined by equations using only the BUTHEAD, HPUT and ADDSEQ operators? These are not trivial questions.

As shown above in the case of the constant sequence of 1-s, operator equations can be used to characterize specific sequences as well.

For example, the Fibonacci sequence: 1, 1, 2, 3, 5, ... may lead us to the equation:

S + BUTHEAD S = BUTHEAD BUTHEAD S

which holds not only for this sequence, but also for all its BUTHEADs and for other sequences as well.

The fact that all the "tails" of the Fibonacci sequence satisfy this equation can be verified by direct observation. It can be verified also by observing the linear nature of BUTHEAD with respect to ADDSEQ. That is, the following equation

BUTHEAD ADDSEQ S_1 S_2 = ADDSEQ BUTHEAD S_1 BUTHEAD S_2

obviously holds for any sequences S_1 and S_2, and therefore, we can apply the operator BUTHEAD to both sides of the previous equation.

How can we derive a unique characterization of the Fibonacci sequence FIB? We know that FIB starts with two specific elements. That is, if we apply HPUT 1 twice to BUTHEAD BUTHEAD FIB we get FIB. Therefore, the following two equations:

S = HPUT 1 HPUT 1 BUTHEAD BUTHEAD S

and

S = HPUT 1 HPUT 1 ADDSEQ S BUTHEAD S

are satisfied by S = FIB. The first equation characterizes all sequences that start with two 1-s. The second equation, however, has a unique solution S = FIB, and therefore it characterizes FIB.

A didactical intermezzo

What are types of sequences that can be characterized by such equations? Are there any methods for solving such equations? Are there decision procedures for the solvability of such equations, and, in particular, for the possibility of unique solution?

The treatment of infinite numerical sequences as "first-class" objects and, in particular, as operands, raises also some didactical issues. There is no doubt that this approach is very abstract and sophisticated. At the same time, it seems to be more interesting than the conventional approach used in high school.

Can these ideas, nevertheless, be developed into an exploratory activity suitable for high school students of mathematics? Can, say, senior high school students be led, by a proper didactical mediation, to pose pertinent mathematical questions concerning sequences, and to study them?

There is a slight chance, that in this context, some students can reach for the idea of mathematical induction. For example, if they are asked to find convincing arguments that may validate some of the unique characterizations of the sequences, they may bring up inductive arguments. At least, it seems that the study of operators and operator equations, in this context of the sequences, can serve as a good soil for learning about the principle of mathematical induction. Is it possible that the study of equational characterizations of sequences can serve as a didactical opener also for the concept of recursion? Similarly, is it possible that the notions of a computable sequence, and of a computable real number can be intrigued while doing these activities? A thorough examination of these questions goes far beyond the scope of this specific paper.

Several didactical techniques can be devised in order to cushion the abstract and algebraic treatment of sequences presented in the previous section. In particular, sequences and their operators can have a somewhat concrete representation as programs. This can be exploited for didactical purposes. The idea is to use programs as manipulatives (cf. [6]) in constructivistic learning processes (cf., [7]).

A functional microworld of infinite numerical sequences

The conventional approach toward sequences teaches us to define them in terms of their elements. In particular, sequences can be defined by the functions that yield, for each position n, where n is any non-negative integer, the n-th element of the sequence. For example, the first element of the sequence is given as the value of the function at 0 etc..

Clearly, a program for the function, which computes the specific elements of the sequence, can be regarded as a *representation* of the sequence itself.

Thus, DOUBLES, defined as the function

```
TO DOUBLES :N
IF :N = 0 [OP 1] [OP 2 * DOUBLES :N - 1
END
```

can be considered to be a representation of the sequence of all non-negative powers of 2.

Similarly, the function:

```
TO ONES :N
OP 1
END
```

can be considered to be a representation of the constant sequence of 1-s.

The following meta-procedure LINE will help us to "see" sequences in a more concrete (or *elementary)* manner. It accepts two "inputs" - (1) a positive number; and (2) the *name* of any function, defined for all non-negative integers. It prints out the "elementary representation" of the sequence abstractly represented by the named function:

```
TO LINE :M :FUN
PRINT.LINE 0 :M :FUN
END

TO PRINT.LINE :I :M :FUN
IF :I = :M [PR [...] STOP]
     (TYPE RUN SE :FUN :I  CHAR 44 CHAR 32)
```

```
PRINT.LINE :I + 1 :M :FUN
END
```

For example, the instruction LINE 8 [ONES] can be used to print out a line of 8 1-s.
We turn now to the programming of operators on sequences, as meta-functions, i.e., as functions of functions.

The three basic operators HEAD, BUTHEAD and HPUT are defined by:

```
TO HEAD :S
OP RUN SE :S 0
END

TO BUTHEAD :S :N
     OP RUN SE :S :N + 1
     END

TO HPUT :C :S :N
IF :N=0 [OP :C]
     OP RUN SE :S :N-1
     END
```

These are not plain functions. They must receive as "inputs", names of functions and they evaluate them as functions of the appropriate numerical arguments. Therefore, all the variable S (i.e., the variable that stands for infinite sequences,) must be provided with values that are either *words* or *lists*. Furthermore, these words or lists must be terms that need a single argument value (as a syntactical complement,) in order to form *a complete expression that has a numerical value*.

This means, for example, that the expression HEAD [DOUBLES] is correct and it denotes the first element of DOUBLES, which is 1. The expression BUTHEAD [DOUBLES] can be used as the name of the function that represents the sequence: 2, 4, 8, 16, ... That is, BUTHEAD [DOUBLES] 0 is 2, BUTHEAD [DOUBLES] 1 is 4, etc.. The expression HEAD [BUTHEAD [DOUBLES]] denotes the second element of DOUBLES, which is 2. Thus, BUTHEAD [DOUBLES] represents a sequence and so is HPUT 7 [BUTHEAD [DOUBLES]]. Hence, the instruction LINE 9 [HPUT 7 [BUTHEAD [DOUBLES]]] results in printing the elementary representation of the following sequence: 7, 2, 4, 8, 16, 32, 64, 128, 256, ...

We turn now to the more algebraic operators:

```
TO ADD :C :S :N
OP SUM :C RUN SE :S :N
END

TO MULT :C :S :N
OP :C * RUN SE :S :N
END

TO DIF :S :C :N
OP (RUN SE :S :N) - :C
END

TO DELTA :S :N
OP (RUN SE :S :N+1) - RUN SE :S :N
END
```

```
TO ADDSEQ :S1 :S2 :N
OP (RUN SE :S1  :N) + RUN SE :S2 :N
END

TO MULTSEQ :S1 :S2 :N
OP (RUN SE :S1  :N) * RUN SE :S2 :N
END

TO DIFSEQ :S1 :S2 :N
OP (RUN SE :S1  :N) - RUN SE :S2 :N
END

TO SUMSUP :S :N
IF :N=0 [OP HEAD :S]
OP (SUMSUP :S  :N-1) + RUN SE :S :N
```

The syntactical rule for the use of these meta-functions, as mentioned above, still applies. Try LINE 10 [SUMSUP [ONES]] or LINE 10 [DELTA [DOUBLES]].
These operators and sequences make up a microworld in which one may explore sequences through several types of learning activities. The simplest manner would be working with specific sequences, functionally defined, such as DOUBLES or ONES, and studying their properties. The very activity of defining various specific sequences by programmed functions is enlightening. However, if instead of specific sequences, the students will study types of sequences, (such as arithmetic sequences, geometric sequences, periodic sequences, algebraic sequences, etc.,) this exploration can develop into a rich mathematical activity. The students can be asked to invent and define additional operators, say by using composition of the above. Besides observing the direct effect of such operators on sequences, the students may examine the effect of composition of operators and compare them to the effect of other operators and of other compositions of operators.

Recursive programming of infinite sequences

Equations that characterize specific sequences may be translated into LOGO definitions of these sequences in a very direct manner. For example, the students may discover the previously mentioned equation that characterizes the Fibonacci sequence. If we call this sequence FIB, then, using the syntax of our microworld, the equation can be written as follows:

```
FIB = HPUT 1 [HPUT 1 [ADDSEQ [FIB] [BUTHEAD [FIB]]]].
```

It may be surprising to realize that the following definition of FIB is valid in LOGO:

```
TO FIB :N
OP HPUT 1 [HPUT 1 [ADDSEQ [FIB] [BUTHEAD [FIB]]]] :N
END
```

We are dealing with operator equations that have the form

S = T[S]

where T is any given operator (or a composition of operators). Of course, such an equation must have a unique solution in S if it is used for a computation, or a construction, of a specific sequence. Note well that we have here a special case of what seems to be a general situation of recursive programming (e.g., [2]). In most circumstances, recursive programming actually applies to infinite entities such as infinite sequences of related structures. Attempts to characterize an infinite structure of a given object Obj can lead to the discovery of a part SubObj of Obj that is similar to Obj itself. This situation is true independently of programming considerations (e.g., [3] and [4]). If we denote by T, the transformation that reconstructs Obj from its copy SubObj then we end up with an equation of the form

T[Obj] = Obj

where T is an operator.

Now, if Obj is the unique fixed point of the operator T, then we may try the following definition of Obj in LOGO:

```
TO Obj :N
OP T [Obj] :N
END
```

For example, it is not difficult to realize that DOUBLES hides a copy of itself, inside itself. Remove the HEAD of DOUBLES, i.e., take BUTHEAD [DOUBLES]. Then, divide each element of BUTHEAD [DOUBLES] by 2 and you get DOUBLES back again. This means that the equation X = MULT 1/2 [BUTHEAD [X]] is satisfied by DOUBLES. This equation does not characterize DOUBLES since it is satisfied also by all the BUTHEADs of DOUBLES. It does characterize, however, the class of all geometric progressions with q = 2. The right-hand side of this equation describes the process of finding a copy of DOUBLES inside itself.

Now, we reverse this process properly, step by step, as follows. First we derive MULT 2 [X] = BUTHEAD [X]. Then we HPUT 1 both sides and get HPUT 1 [MULT 2 [X]] = X which characterizes DOUBLES as its unique solution.

Indeed, DOUBLES can be defined in LOGO by either one of the following definitions:

```
TO DOUBLES :N                          TO DOUBLES :N
OP HPUT 1[MULT 2[DOUBLES]] :N      OP HPUT 1[2*DOUBLES] :N
END                                    END
```

As the above examples indicate, the equations that describe the path from the given sequence to its internal copy, yield, by reversing the path, only partial characterizations of the given sequences. These partial characterizations become complete with the specification of the first elements of the specific sequences (through the application of HPUT).

These observations lead to an approach that views a sequence as a parameter-dependent object, where, usually, especially in the sequences encountered in high school mathematics, the parameters represent the first elements of the sequence or are related to them.

Concluding notes

The microworld described in this paper can serve as an environment in which infinite sequences are studied through various kinds of mathematical activities. This microworld is based on treating functions as streams, using a few basic meta-functions as operators.

One of the intended outcomes of using this microworld of infinite sequences is the change of mind of the students, from viewing sequences only as composite structures with elements, to the possibility of viewing them as "chunked" objects. Hopefully, this is a skill that can be achieved by the transition from the study of sequences defined by their elements, to the study of sequences as operands defined by equations wit h operators. The activiti es within thi s environment may help students to attain a broader notion of recursion. It adds evidence to the complexity of the concept of r ecursion, and to the insu fficiency of the commo n definitions provided in the literature for this concept [4].

This particular microworld, as described in this paper, has been used extensively in Computer Science teacher training programs with some positive results. The ideas have not been tested yet on high school students.

REFERENCES

[1] Abelson, H., Sussman G. J. & Sussman J. (1985), *Structure and Interpretation of Computer Programs*, The MIT Press, Cambridge, Massachusetts.

[2] Burge, W.H. (1975), *Recursive Programming Techniques*, Addison-Wesley Publishing Company, Reading, Massachusetts.

[3] Give'on, Y .S. (1989) , "Teaching recursive program composition in procedural environments," *Machine-Mediated Learning*, 3(2), pp. 125-145.

[4] Give'on, Y.S. (1990), "Is Recursion Well Defined??," *Computers & Education.* 14(1), pp. 35-41.

[5] Give'on, Y.S. (1992), "Infinitely long streams in Logo," *Eurologos* 1 pp. 17-19.

[6] Perl, T. (1990), "Manipulatives and the computer: a powerful partnership for learners of all ages," *Classroom Computer Learning*, 10(6), March 1990, pp. 20-29.

[7] Schuyten, G., (1989) "Constructivism as a theoretical framework for LOGO-based environments," *Proceedings of the Second European Logo Conference 1989*, Gent, Belgium, pp. vii-xxii.

From Square to a Thousand Cranes

Márta Turcsányi-Szabó

Eötvös Loránd University, Faculty of Natural Sciences
Department of General Computer Science
Hungary
e-mail: Turcsanyine@ludens.elte.hu

Abstract

This paper hopes to prove that LOGO and ORIGAMI are brother and sister in the family of creativity.

Keywords: Logo, arts, recursion, origami

From Square to a Thousand Cranes

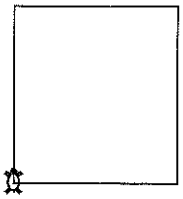
Introduction

Recently I have spent some time in Japan so I had the opportunity of getting acquainted with Japanese culture, the language, and the people. These three interact strongly with each other, so understanding the one urges the learning of the other. Getting acquainted with people and their thoughts meant learning their language, the beauty of the language is enhanced by the admiration of calligraphy, calligraphy is the companion of sumi-e, to learn sumi-e one has to learn more about paper, touching paper leads to folding paper, and folding paper is origami. I have always admired arts and I especially loved to do handicrafts myself. Origami gave me the delight of expression which led my research on this ancient art. Being an old LOGO-ist I often saw the world through the Turtle. One learns a lot about nature through playing around with fractals and vice versa. [1]. The patterns produced by some recursive procedures [2] give inspiration to origami structures which I would like to introduce.

LOGO-ORIGAMI

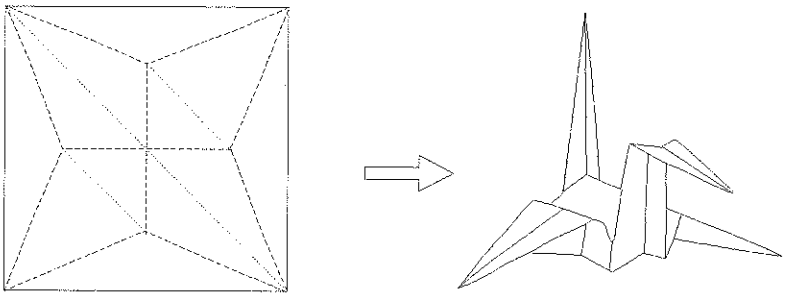
Let us start from the simple procedure that every beginner in LOGO tackles sooner or later.

```
TO SQUARE :SIZE
      REPEAT 4 [FORWARD :SIZE
                RIGHT 90]
END
```



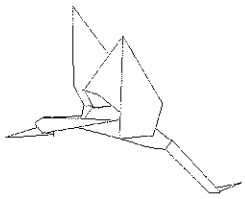
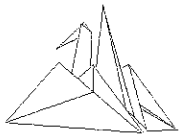
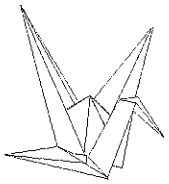
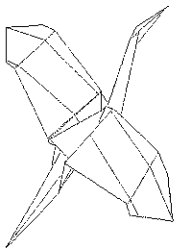
```
SQUARE 200
```

The result represents a 200 unit square piece of paper which is the starting point of every origami object. To produce a three dimensional piece of origami work is a child's job, of which Japanese children are very talented. A lot of objects are well known basics, while the rest depends on imagination and talent. The most famous piece of origami is probably that of folding a crane or *tsuru*, which is a lucky bird in Japan. Here are the basic folding lines of the crane:



...but the completion really depends on artistic view. As Kunihiko Kasahara, my favourite origamist wrote in his book [3]:

"For the devoted origamian, it is an object of affection and, at the same time, stimulus to the spirit of challenge."

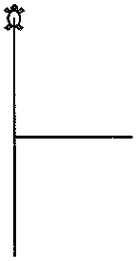





Akira Yoshizawa[4] Toshio Chino[5] Jun Maekawa[6] Kunihiko Kasahara[3]

Or see Eiji Nakamura's further variations [7].

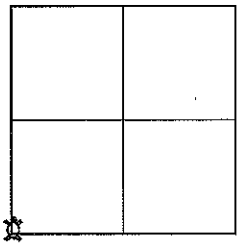
Let us go back to our piece of paper or in other words our **SQUARE**. Let us divide the sides of the paper into half and cut with a scissors till nearly the centre. We can do the actual cutting of one side with the Turtle itself by replacing **FORWARD :SIZE** with the following generalised Koch curve procedure [8] where N=4 (No. of segments) & r=1/2 (scale ratio).

```
TO ONE_SIDE :SIZE
       FORWARD :SIZE / 2
       RIGHT 90
       FORWARD :SIZE / 2
       LEFT 180
       FORWARD :SIZE / 2
       RIGHT 90
       FORWARD :SIZE / 2
END
```



Let us construct the **NEW_SQUARE** procedure with this **ONE_SIDE**.

```
TO NEW_SQUARE :SIZE
       REPEAT 4 [ONE_SIDE :SIZE
                 RIGHT 90]
END
```



where

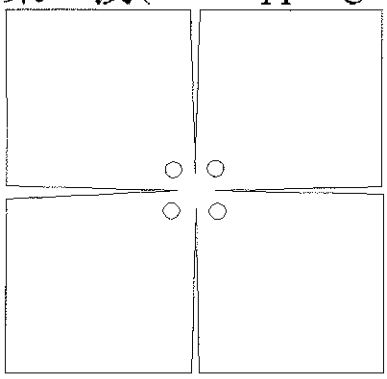
```
NEW_SQUARE 200
```

... results in 4 squares actually, but the road the Turtle takes represents the future edges of the paper.

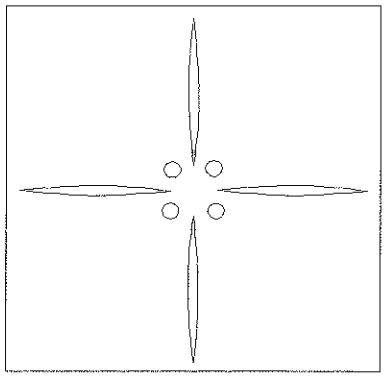
Do not cut all the way till the centre, leave out a few millimetres to keep the paper together.

Creativity solves the problem of choosing the corner for the head of the crane. Do not hold back on experimenting with other cutting lines or trying out combinations. In the following page, some examples of slit designs from the oldest manuscript Senbazuru (A Thousand Cranes) of traditional origami in Japan can be seen [9]. It was written in 1797 by Rokouan Gido Ichien, the chief Priest of a temple at Kuwana in Ise-no Kuni. It contains 49 kinds of crane-folding methods with the use of one square or rectangle sheet of paper with slits to produce a fairly good number of mutually connected cranes. Each amulet is accompanied by Akizato, the famous poet's Haiku to emphasise lyrical expression, and illustrations are drawn by Shunsensai. (A Haiku is a type of short poem, having lines of 5-7-5 syllables, in which the poet attempts to express his deepest emotions by describing nature in simple but beautiful language.)

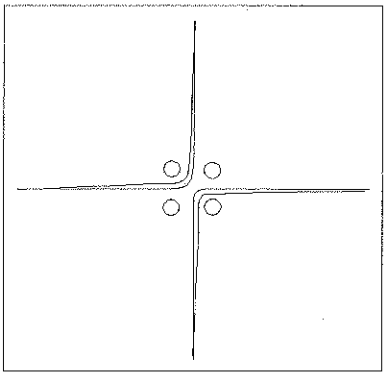
①楽々波(The Rippling Sea)



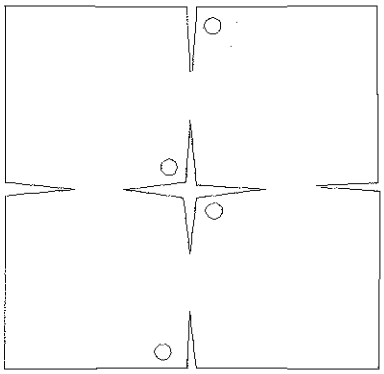
②風車(Windmill)



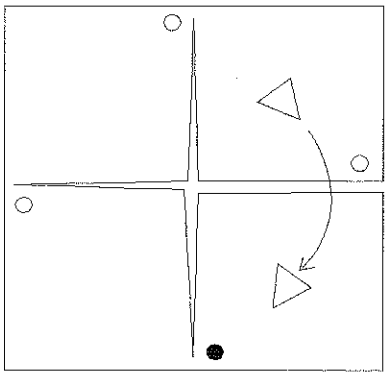
③四つの袖(4 Kimono Sleeves)



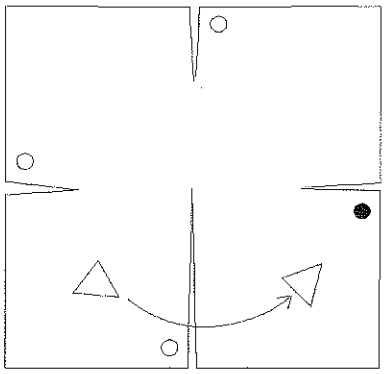
④ 瓢箪町(Pumpkin Town)



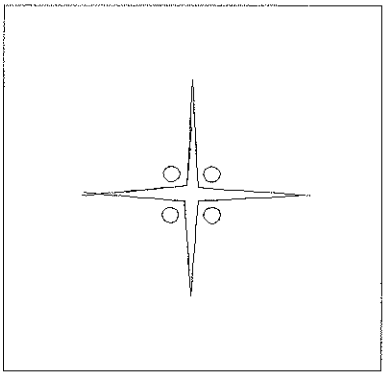
⑤鼎(Cast-iron Pot)



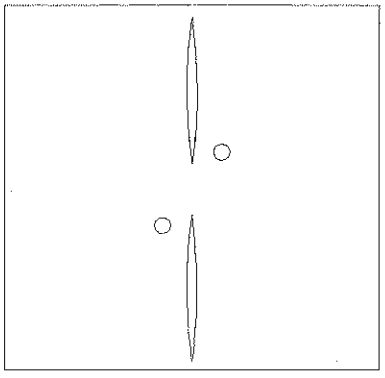
⑥三巴(Three Tangled)



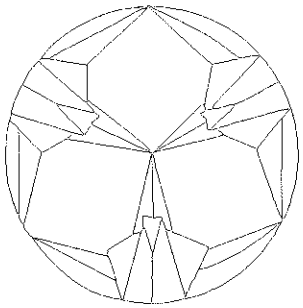
⑦龍膽車(Bellflower Cart)



⑧相生(name),比翼(Wings)

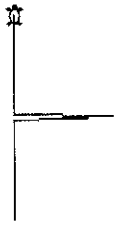


A lot of these origami emblems appear in family crests, which is possessed by every Japanese family. The origin of Japanese family crests (*mon*) goes back to the eleventh century. Each of the high-ranking countries in those days began using a specific textile design for his most formal costume worn at the Imperial court [10].

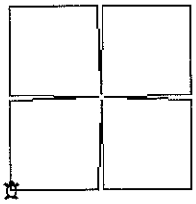


To increase the number of cranes you should go back to your LOGO procedure and experiment. First of all if you would like to see the actual cutting surfaces produced, it is better to substitute the **ONE_SIDE** procedure with a slightly different one:

```
TO REC_PATTERN1 :SIZE
       FORWARD :SIZE / 2
       RIGHT 88
       FORWARD :SIZE / 2
       LEFT 176
       FORWARD :SIZE / 2
       RIGHT 88
       FORWARD :SIZE / 2
END
```

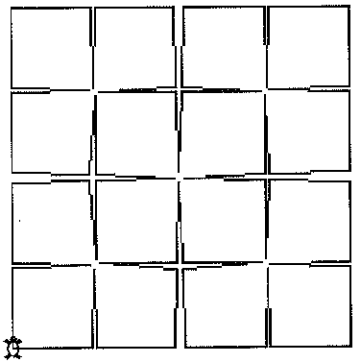


```
TO NEWER_SQUARE :SIZE
       REPEAT 4 [REC_PATTERN1 :SIZE
                 RIGHT 90]
END

NEWER_SQUARE 200
```



Using this as an element for your **NEW_SQUARE** procedure helps you visualise where and how to cut or vice versa guiding through the lines of cut gives you the LOGO procedure.
Understanding recursion is not so easy but paper cutting techniques can help a lot. Start cutting the resulting bits of squares in the very same way for every square to produce the basics of the next and next levels. At the same time try to put down the algorithm in LOGO too:

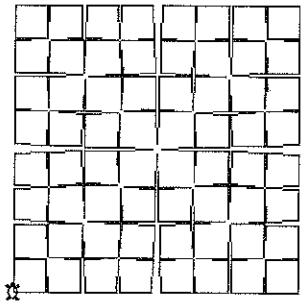
```
TO REC_PATTERN2 :SIZE
     REC_PATTERN1 :SIZE / 2
       RIGHT 88
       REC_PATTERN1 :SIZE / 2
       LEFT 176
       REC_PATTERN1 :SIZE / 2
       RIGHT 88
       REC_PATTERN1 :SIZE / 2
END

TO NEWER_SQUARE2 :SIZE
       REPEAT 4 [REC_PATTERN2 :SIZE
                 RIGHT 90]
END

NEWER_SQUARE2 200
```



This produces a new level of cutting. Continue the next level of cutting and algorithm writing.

Since each level of cutting is done after the previous cutting level, the procedures can be nested and the algorithm continues till we reach the desired level:

```
TO REC_PATTERN :SIZE :LEVEL
      IF :LEVEL = 0 THEN FD :SIZE STOP
     REC_PATTERN :SIZE / 2 :LEVEL - 1
      RIGHT 88
      REC_PATTERN :SIZE / 2 :LEVEL - 1
      LEFT 176
      REC_PATTERN :SIZE / 2 :LEVEL - 1
      RIGHT 88
      REC_PATTERN :SIZE / 2 :LEVEL - 1
END
TO REC_SQUARE :SIZE :LEVEL
      REPEAT 4 [REC_PATTERN :SIZE
                    :LEVEL RIGHT 90]
END
```



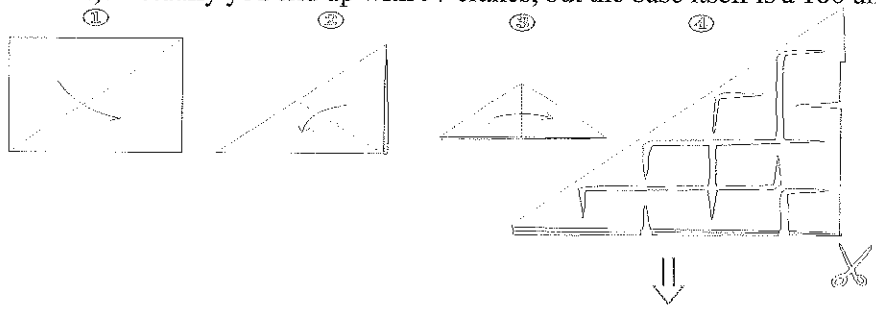
Try this new procedure of recursive squares and decide on the level of cutting. As you can see the :

```
SQUARE1 300 1
SQUARE1 300 2
SQUARE1 300  3
```
commands do the job perfectly.

You should try to do each origami masterpiece from a single piece of paper, so do take care while cutting to keep the paper together. When you are finished comes the more difficult part of folding the birds separately without tearing the paper.

Patience and gentleness should be your guide.

Choosing the corner for heads and tails enhances the product as well as choosing cutting slits.
This page describes another beautiful example from Rokouan. It is called Hyakkaku
(A Hundred Cranes). Actually you end up with 97 cranes, but the base itself is a 100 unit-square paper.

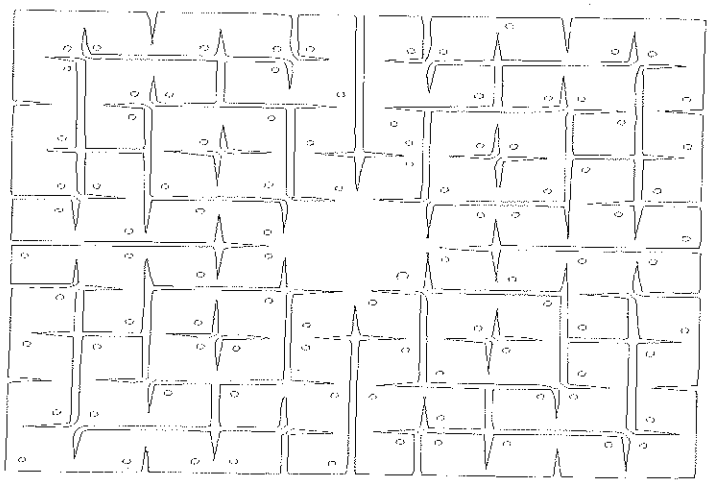


百鶴

朝比奈の
紋を十づっ十よせて
百人力の
鶴の紋なり

100 cranes

Asahina family crest
is represented by
10x10 that is
100 man power.



The Haiku accompanying this marvellous piece of origami is difficult to translate but it certainly gives the impression of force. Actually Asahina has a complex internal meaning and it can be a name of a place, a family or crest too.

If you try a number of easier variations, and manage a few of the complicated ones too you will definitely get some expertise. But I must tell you that, although Rokouan's book of origami is indeed very popular, most admirers remain at this stage and only a few persistent fellows complete the book. If keeping paper together is what hinders the project, you might as well complete the design from separate pieces of paper. I guess Rokouan's spirit would not be insulted by doing so.

Conclusion

Experimenting with fractals produces designs that one recognizes in nature or other sciences. Experimenting in origami reveals the fact that designs are based strongly on geometric forms. I hope this paper managed to convince you of the ties in LOGO-ORIGAMI and seeing the actual pieces of origami would arouse your artistic attention too.

REFERENCES

[1] Benoit Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman and Co., San Francisco, 1977.

[2] D. Senftleben - M. Turcsányi-Szabó, A LOGO programozási nyelv. (The LOGO programming language.), Edited by Zs. Votisky. Published by M_szaki Könyvkiadó, Budapest 1986.

[3] Kunihiko Kasahara, Origami Omnibus, Japan Publications Inc., 1988.

[4] Akira Yoshizawa, Origami Toku Hon II (Origami reader II), Kamakura Shobou Publications, 1986.

[5] Kunihiko Kasahara, Origami Shinseki (The New Era of Origami) of Viva Origami series Sanrio Publications, 1989.

[6] Jun Maekawa - Kunihiko Kasahara, Viva Origami, Sanrio Pblications, 1983.

[7] Eiji Nakamura,Henka Orizuru (Crane Origami Variations),1971.

[8] Benoit Mandelbrot, Fractals form, chance, and dimension, W. H. Freeman and Co., San Francisco, 1977.

[9] Rokouan - Akizato - Shunsensai, Sembazuru (A Thousand Cranes) 1797.

[10]Compiled by the Matsuya piece-goods store, Japanese Design Motifs, Translated by Fumie Adachi Dover Publications Inc., 1972.

The Use of Computational Representations in the Teaching and Learning of Programming Concepts

Heloisa Vieira da Rocha

Center of Computer Science Applied to Education
State University of Campinas - UNICAMP
Campinas, S. P. Brazil
e-mail: heloisa@dcc.unicamp.br

Abstract

The aim of this work is to explore the use of computational representations, in real time, as an aid to the learning of abstract concepts and processes. The knowledge domain chosen here was the computer programming. Computer programming was chosen because there are concepts and processes such as variables, execution flow, recursion, iteration, etc. that are difficult to be assimilated through the conventional means. A computational system of representations of the process of execution of a program was developed. The Logo language was considered, coupled with its learning methodology, enhancing then a system that amplifies the Logo programming environment. The system was implemented to facilitate the development of an adequate mental model which could guide the whole interaction of the user with the programming language. The results obtained with this work in the programming context can be extended to other areas of knowledge, such as Physics, Chemistry, Biology, etc... In such domains there is a series of abstract concepts such as speed, balance equilibrium, pressure, etc. which are also rather difficult to be understood. The comprehension of such concepts can certainly be facilitated by the use of other types of representations of phenomena where these concepts are present.

Keywords: teaching and learning environment, programming, programming concepts, Logo,

The Use of Computational Representations in the Teaching and Learning of Programming Concepts

1. Introduction

Psychologists of the Gestalt line [3] distinguish two types of learning how to solve problems: conducted learning and understanding learning. For example, regarding the learning of Mathematics, a distinction is frequently made, or at least it should be made between "getting the right answer" and "understanding what one is doing".

In a classical example, Wertheimer suggests that there are two basic ways of teaching a child how to find the area of a parallelogram.

The first method involved tracing a perpendicular line, measuring the length of the base and calculating the area using the formula `area = base * height`. That is what Wertheimer calls conducted learning, or yet, in a more assertive way, it is called the 'nonsense' method, because the student is induced to memorize a formula and a procedure to use it.

The second method asks the student to visually explore the parallelogram until he realizes that it is possible to cut a triangle from one of its ends, place it on the other end and form a rectangle. Considering that the students already know how to calculate the area of a rectangle, the problem is solved. Wertheimer calls this method "Structural Understanding" or "Apprehension of Significant Relations", because the student gets a vision of the structure of the parallelogram.

According to Wertheimer, if a test is given involving parallelograms of the same type of those used during the process of instruction, both groups of children will obtain good results. However, if a test is given which involves unusual parallelograms the "conducted learners" will say that they were not taught that subject, while the "comprehensive ones" will be able to derive the answer.

This example suggests that when the aim is the creative use of new pieces of information, it is important to use methods and environments which favor comprehension. As to the teaching of programming, it can be observed that the learner is being conducted and acquiring a series of solution patterns, without worrying about the concepts involved in such patterns. I believe that this is not what we want to do with programming.

Courses about programming will be offered at different levels: from primary school to the professional level. Considering such a context, many questions come up:

How to teach programming?

What is to be taught and in what level?

The answers to these questions are not automatically available; it is necessary to know more about how students learn how to program, which difficulties they find and how the instruction can help them overcome such difficulties.

It is important to make it clear that addressing the problem of how well the students are learning how to program is a subject that deserves much attention

2. Problematic of learning how to program with Logo

For beginners in programming, the computer does not exist at the level of binary system, ports, central processor and other basic units. The computer will be a definite machine as a function of the programming language constructions which they are learning. In that sense, one can be an extremist in the sense of affirming that if the beginner starts programming through the Logo language, the computer will be for him a Logo machine, that is, a machine that functions according to the instructions available in the Logo language.

Based on that, it is then defined that simplicity and visibility are two important characteristics of a programming language for beginners, because usually they start programming without any idea about the features of the notational machine defined by the language they are learning. The notational machine is the idealized model of the computer, defined through the construction of programming language; therefore, it is an idealized conceptual computer. For the beginner, the features of such a machine are dependent on the language and not on the hardware.

Logo is a programming language designed to facilitate the learning of programming; it is quite simple to be assimilated and used. It has a sub-set of commands which control the turtle, where by means of a small number of very clear actions, interesting results are quickly obtained by beginners who are writing simple programs. With a set of ten instructions plus the structure of procedures, it is possible to have an environment where rewarding effects are obtained in the first contact with the computer.

However, in order to learn/teach programming a facilitating language is not enough, because many concepts need to be fully understood in order to be correctly used. For example, from the work with Logo one can observe that people use procedures with parameters and cannot work with variables in a given assignment. In spite of using variables as parameters, they don't have the concept of what a variable is in a programming language. Innumerable examples could be mentioned, demonstrating that mere use does not imply understanding of the concept.

The concept faults, while working with Logo are felt when one advances from the graphic part to the symbolic manipulation. Several barriers occur, because the level of abstraction and the complexity of the activities increase considerably. Besides, the programming paradigm being used is changed, going from the procedural (graphic environment) to the functional symbolic environment. It is not possible anymore to follow the program execution which is somehow possible in the graphic part, because through the execution of programs that draw, the program behavior is shown step by step. Whereas in the symbolic processing what one gets is simply the result in the form of lists, sentences, words or numbers. It becomes difficult to understand how the computer got the result or not, because there are no more direct ways to follow the execution of the program.

What is the difficulty in understanding programming concepts?

These concepts are directly related to the computer operation, which in this case is defined by the language. In that sense the computer is an extremely opaque machine, because if we open up a computer and observe what goes on when a program is run, we won't see absolutely anything. The way we have to explain is through representations, that somehow show the operation. Allowing himself this "visibility" of the process of execution, the user has a better feedback of his interaction, which can help in the evolutive process of his mental model construction [1] about the computer function.

3. Objectives

There are then abstract concepts which are essential to the learning of programming. Those are concepts that cannot be acquired through concrete and static observations due to the computer opacity as a dynamic machine. A computational system that makes representations of the process of execution of a Logo program

was developed based upon the hypothesis that the supplying of alternate representations helps the understanding of concepts with such characteristics and, that in the case of computational concepts the dynamism of representations is essential.

As previously mentioned, even if Logo is a language that aims at facilitating the learning of programming, there is a barrier in the transfer from the graphic to the symbolic. How to facilitate this transfer and consequently to facilitate the learning of essential concepts, respecting the basic methodological aspect of Logo, that is, to render a learning environment where the knowledge is not passed along, but rather developed through the interaction with the objects of the environment, is what sets the basis for the system conception.

4. Basic characteristics of the implemented system

I have developed a system that gives several representations of a program execution written in non graphic Logo. One of the main aspects of the system is the multiplicity of representations, a result of the observation of people using computer where a diversity of learning styles is seen and, therefore, any teaching material should have diversity instead of uniformity. Another aspect is the dynamic characteristics of the representations, that is, they are given in real time, at the same time that the procedure is being executed.

The system's objective is not to teach the concepts, but rather to facilitate its understanding, it is "the see it to better understand it".

The process of choosing the implemented representations was, so to speak, "ad hoc" based on previous experience of programming teaching. I have defined a set of representations based on the representation used statically, adding primordially the real time characteristic. The computational concepts considered in the development of the system were execution flow, variables and recursion. I believe these are the fundamental concepts of programming, and all the others, such as iteration, procedures, etc. can be treated as a function of them.

The presentation form of the system can be seen as two basic sub-sets of representations which I call local and global.

The aim of the sub-set of local representations is to show at a certain point of execution what is happening and what the computer "knows" about the program. In such a representation, the past history of the execution is lost, that is, the process that conducted that moment of execution is not visible. Because I believe that visualizing the process as a whole is important to the understanding of the studied concepts, considering especially the relevant aspects of execution flow, at a macro level, I have defined another representation which shows that history of execution. This other representation I call global.

As far as the local representation is concerned, the system gives a screen subdivided in three windows.

The first window, named **"Procedure in Execution"**, contains the text of the procedure in execution and the cursor in reverse video, or in a different color, if the monitor is a color one, and it shows the instruction that is being executed. Through this representation I want to make it clear that, among other aspects, the sequence of the execution of instructions. This is an aspect which may seem irrelevant, but that causes many problems during the construction of procedures with several conditioned options, with repetitive structure and, fundamentally, with many sub-procedures. It is noted that for many programmers, the sequence in which the instructions executed in the return of sub-procedures is not clear.

In the system whenever a sub-procedure is called, the text for the calling procedure is substituted by the called procedure, and a message informs that a call is taking place.

The same occurs during the return of the sub-procedure, where a message informs that a return is taking place and the text of the calling procedure returns to the screen and the execution goes on. The same scheme is used in the case of recursive procedures.

In the second window, named **"Activation Tree"**, the dynamic structure of procedure calls is represented. In this window a rectangle is created with the name of the called procedure and containing the name and the value of the parameters at the moment of the call. These rectangles are constructed and eliminated according to the process of call and return of procedure.

The third window, named **"Variables"**, does contain the name of active variables and their current values. Each variable value alteration within the program is studied with special emphasis on this window. The purpose of this window is to show the relevant aspects to the conceptualization of variables, such as name, value, and validity goal.

Besides the windows, there exists an option menu, through which the user controls the whole process of execution of the system.

There is a fourth window which is not always on the screen, whereby the input values are requested and the output values are shown. This window is opened only when executing the input commands and the output commands and disappears at the end of the execution of the corresponding commands.

Another window, which is not always present on the screen, indicates relative outputs to the OUTPUT command, which indicates the return of functions or operations. In this window there exists the value which is being sent and the corresponding origin and address procedures. Also, this window does only appear when executing the OUTPUT command and disappears the moment the system is returned.

Through a function key the global representation of the execution is assessed. This representation is carried out on a single screen and shows the execution schemes of the procedures. On this screen one has the text of all the procedures that make up the working area, that is, the file used in the current execution. Through a scheme of colors, cursors are used to indicate which instruction is being executed and which executions are suspended, that is, call and return points in the other active procedures of the working area.

Every interaction with the system is done through the selection of options in menu or through the selection of previously defined function keys. The execution of a program can be interrupted at any moment, one can have access to another type of representation, redesign the screen, consult the file, etc.

5. Results from the use of the system

The system was used experimentally [2] in three situations and with three distinct objectives:

a) diagnose and remedy the concept of recursion

b) learning of recursion in the symbolic processing

c) to verify how much the computer specialists dominate the concept of recursion

In general, I concluded that the conceptual faults were not restricted to professionals from other areas other than Computer Science. Therefore, difficulties in understanding the concepts exist even for people whose perfect understanding is essential for the professional life.

Making the execution process explicit helps the understanding of concepts and the help extent is proportional to the type of mental structure that the user has. This mental structure is achieved from the user's experience with program-ming. In the case of the computer science specialists, they already have a good mental model and the system has facilitated the correction of the model and, consequently, the concepts. As far as the other students are concerned, the mental model was extremely defective and the system should allow two things: strong correction (almost construction) and acquisition of the concept.

The representations have proven to be adequate in the sense that people took hold of them, using similar forms even when outside the computer. And as in principle the objective of the system is not to teach, but rather to instrument so as to facilitate the learning, this result was quite significant.

Some of the specific conclusions can be pointed out:

a) The work in the graphic environment doesn't always favor the understanding of how the computer effectively processes a procedure. In the graphic environment one has more of a reflection about the results obtained than about the procedure that led to that result. Many times interesting and satisfactory results are obtained through absolutely empirical developments.

b) In order to understand the concept of recursion it is necessary to understand how the computer effectively processes a recursive procedure. The people need to be instrumented to be able to think about recursion and consequently understand the process. In that sense, the system has proven to be quite adequate, because in many occasions I could observe the people make use of representations similar to the ones of the system in order to follow the execution of a procedure outside the computer. It was observed that the people achieved a better structuring of the knowledge domain, which reflects an upgrading of a mental model of the computer function.

c) In order to understand the concept of variable in its widest sense, it is necessary to dissociate it from the concept of parameter. This is a serious fault from the programming teaching method with the Logo language. Because it is a language rooted on the functional paradigm, the introduction of variables as parameters is natural. However, as we have the graphic environment which is procedural, this concept should be enlarged. Also in this aspect the system has proven to be adequate, because during the dynamic presentation of the execution, it was clear the difference between parameters that existed during the execution of a procedure and the global variables that persisted and that could be used freely at any point of the execution.

d) In a formal course situation, which was one of the experimentation environments, there was a significant yield increase, reaching up to 90 %. This percentage by far surpasses the yield of any other course that has been given. These results are due in great part to the motivation and facilities provided by the use of the implemented system.

6. Conclusion

In the process of teaching programming the basic concepts of programming are not being focused. Such concepts are not many — variables, execution flow, and recursion.

Besides, when using the computer as an auxiliary in the process of abstract concepts acquisition in general, and programming in particular, one should keep in mind that the acquisition of concepts involves a cycle; to do, to reflect upon what has been done, to debug, to do, to reflect... The computer should be introduced in this learning process favoring, and not teaching that the people can form their own model of concepts through individual exploration. After all, we have never been taught about cats, and how to differentiate them from dogs. It is necessary to let the object of study be susceptible to manipulation, in such a way that the learner can interact with these computational objects in the same way that he interacts with animals and is able to learn the difference between a dog and a cat.

The set of representations also can be seen as a proposal for a debugging environment for Logo language. Actually, we are, at Center of Computer Science Applied for Education, developping a project, named SuperLogo, wich is a multimedia Logo-based environment for learning. In this project we implemented the system described in this work for debugging all kinds of Logo programs.

At last, the use of dynamic computational systems of representations can be applied to an other domain of

knowledge, which involves concepts resulting from the understanding of abstract processes. Studying new ways of representation that can be used in science areas where concepts such as equilibrium, pressure, temperature, etc are extremely dependent on the process where they occur, is a way to enlarge the current use that is made of computers in the teaching and learning process.

REFERENCES

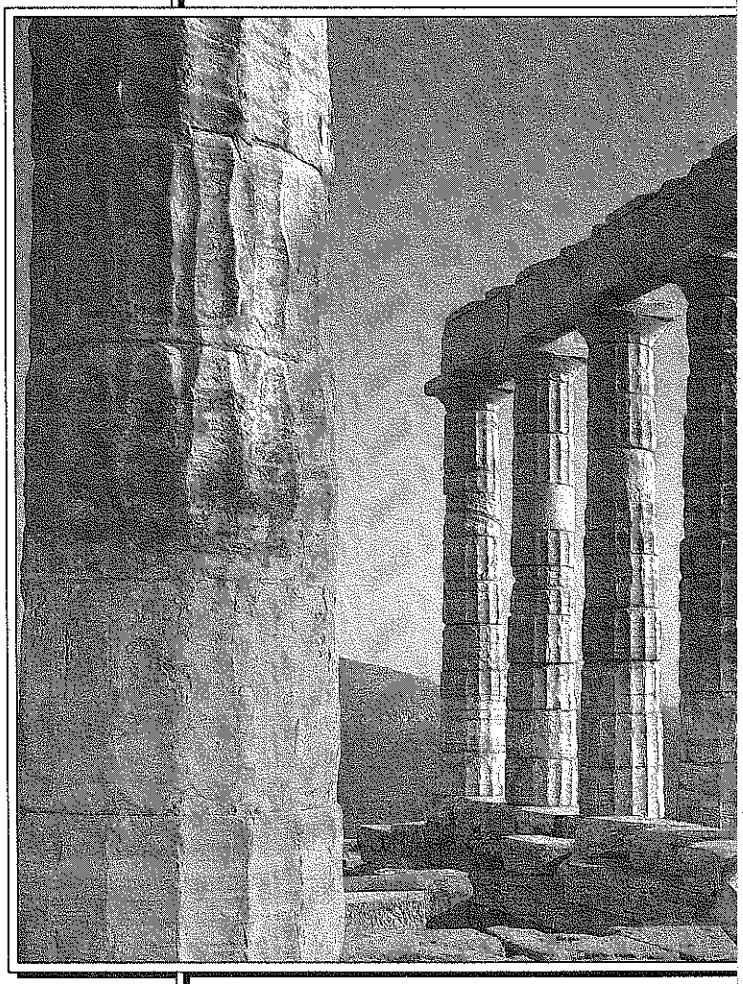
[1] Gentner, D. and Stevens, A. L.(ed.), Mental Models, Lawrence Erlbaum, 1983

[2] Rocha, H. V., "Representacoes Computacionais Auxiliares ao Entendimento de Conceitos de Programacao"(Auxiliary Computational Representations to the Understanding of Programming Concepts), Ph. D. Thesis, Electrical Engineering Department - UNICAMP, Campinas, Sao Paulo, BR, 1991

[3] Wertheimer, M., "Productive Thinking", Harper & Row, New York, 1959

Theme C

The School Interface



C. The School Interface

163 • *Bariamis G., Chaimantas S., Kotsanis Y., Papathomaidi L.*
A Microworld Oriented Approach in a Multi-Functional Logo-Based Curriculum

175 • *Koros-Mikis M., Farkas K.*
Informatics in Hungarian Public Education - Logo Environments in Primary Schools

181 • *Lapidot T., Levy D.*
From Programming to Computer Science: Opportunities and Pitfalls

189 • *Robinson C.*
Conversations with a Computer: Third Generation of Computer Assisted Learning

195 • *Rousseau-Gerard M.*
Pre-lecture par Logo en 3eme Maternelle

201 • *Turcsanyi-Szabo M.*
Where to Place Logo in Teacher Training

A Microworld Oriented Approach in a Multi-Functional Logo-Based Curriculum

G. Bariamis, S. Chaimantas, Y. Kotsanis, L. Papathomaidi

Doukas School
Messogion 36-40, Maroussi
151 25 Athens Greece
tel: (301) 6822500, fax: (301) 6818277
e-mail: ikotsa@leon.nrcps.ariadne-t.gr

Abstract

Through the microworld approach and the project-oriented activities we can enhance both the learning process and the educational applications. Our main aim, in this microworld environment, is to approach the problem solving process in a "human" way, without having to adapt our thinking skills to a computational aspect of view. The presented powerful ideas emphasize the use of Logo as a student-centered tool in a generative active environment.

This paper will focus on four open-ended microworlds (polyspin, alphabet, lego-logo, speaking-turtle) that can be successfully included in any kind of computer-based curriculum or curricular school program (geometry, algebra, physics, etc.). These microworlds (except the last one) have been tested in our Logo-based curriculum which is included as an appendix. This curriculum has been resulted from an over-five year effort to introduce and apply a (one-hour per week, in a condensed way) class of Informatics into the school time-table, covering 4th to 8th grade.

Keywords: Logo, curriculum, microworlds, alphabets, functional programming, multimedia

George Bariamis *was born in 1960. He received a degree in Electrical Engineering from Democritos University of Thrace. He has done post-graduate studies in Pedagogical and Technical School of Athens. He has also taken courses in Digital Electronics and Microcomputers at the Greek Center of Productivity. As a teacher of Informatics at Doukas School since 1987, he has been working with students from elementary level to high school level and also with adults interested in New Technology.*

Sotiris Chaimantas *was born in 1960. He received a degree in Informatics at the Fachhochschule of Konstanz Germany. He has been working in the field of Informatics in Germany and Greece for 4 years. As a teacher of Informatics at Doukas School since 1989, he has been working with students from elementary level to high school level designing and implementing educational software as well.*

John Kotsanis *was born in 1959. He has a degree in Electrical Engineering from the National Technical University of Athens (NTUA) and has done post-graduate research in Computational Linguistics. His interests include Educational Computing and Natural Language Processing. Logo Environments are of special significance to him. He is currently Co-ordinator of Department of Informatics at Doukas School. He is also active in software development and teacher training in various projects at Patras University.*

Lena Papathomaidi *was born in 1960. She received a degree in Mathematics from the Physics & Mathematics Department of Athens University. She has done post-graduate studies in teaching Mathematics at the University of Paris VII . She has also taken courses in Educational Computing at the Greek Computer Society. She is currently teacher of Informatics at Doukas School .*

A Microworld Oriented Approach in a Multi-Functional Logo-Based Curriculum

Introduction

The experience we gained from an over-five year introduction and application of Informatics in primary and secondary education has led us, apart from various subtargets, to a significant teaching target: To develop the learner's ability in handling problems by planning, implementing, testing, improving and finally comparing them to the desired result. This can be achieved by choosing certain ideas and applications that will motivate the learner.

The use of microworlds and the powerful ideas they contain, has proved to be one of the best means in this educational process. These microworlds can include the following features which have been previously referred. They are:

- clearly defined parts of reality
- visualising and modelling process
- highly interactive and friendly
- able to manipulate, explore and be explored
- open-ended, extensible, cooperative, combinable

In the following chapters we will describe four microworlds (polyspin, alphabet, lego-logo, speaking-turtle) that can successfully be included in any kind of curricular school program.

The Polyspin Microworld

The "polyspin" microworld is based on a very simple idea of designing complex patterns from a series of movements and rotations on an initial pattern.

The experimentation with this microworld includes the following stages:

- a *synthesis* process, that is the description of a complex pattern from an initial pattern and a "generation rule",
- an *analysis* process, that is the detection of the initial pattern and the "generation rule" of the given complex pattern, and
- a *human repetitive* process to achieve the desired goal through the trial & error approach.

We can use the microworld to design any kind of shapes (e.g. crystals, snowflakes, flowers, trees, etc.), to experiment with the changes that produce the variables, to also include recursively generated shapes and in this way simulate the nature.

The following procedures represent the basic structural elements of such a microworld (the initial pattern may be a procedure or an instruction list):

`POLY :NUM :SIZE :INSTR`, puts the initial pattern (3rd input) on the edges of the regular polygon made from the number of its sides (1st input) and the size of them.

`SPIN :NUM :INSTR`, rotates the initial pattern (2nd input) as many times as the number (1st input) indicates, until it returns to the starting position.

`TURN :ANGLE :INSTR`, rotates the initial pattern (2nd input) as many degrees as the number (1st input)

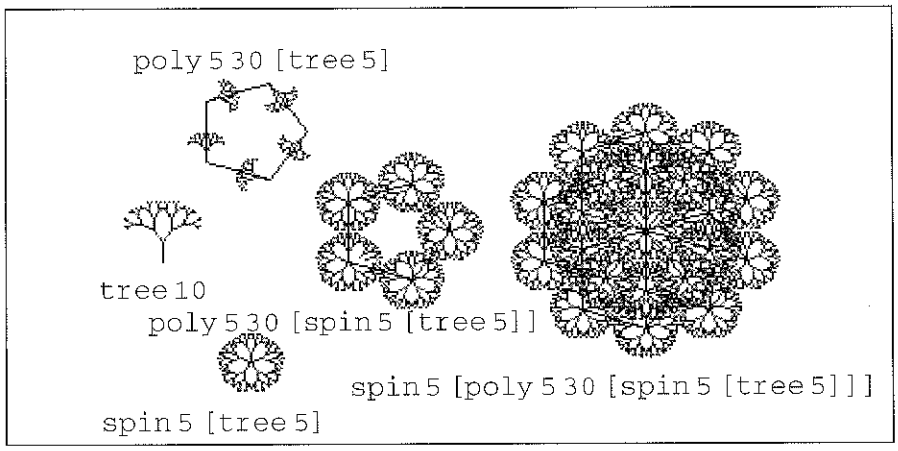
indicates, until it returns to the starting position.

The definitions of the procedures and sample examples are illustrated below (`lcm` function returns the least common multiple of two numbers):

```
to poly :n :side :instr
repeat :n
  [fd :side rt 360/:n run :instr]
end
```

```
to spin :n :instr
repeat :n
  [rt 360/:n run :instr]
end
```

```
to turn :angle :instr
repeat (lcm 360 :angle)/:angle
  [rt :angle run :instr]
 end
```



We can also include new inputs in the microworld (e.g. side and angle increments/decrements) or commands to change the colors of the shapes. The implementation of an "*angle guess game*" for randomly generated patterns should also be attractive for the students and beneficial for the teacher.

The fundamental challenge of the microworld is that in every new movement or rotation, the whole set of the previous patterns takes part and not only the first one. This reflects to the functionality of the microworld and to the interpretation of procedures as usual data.

We approach the problem solving process in a human way, giving the appropriate commands without having to adapt our thinking to the lower details of machine operation or a computational aspect of view. The previous possibilities emphasize the use of Logo as a powerful tool to which we can simulate and finally observe the way we are thinking.

The Alphabet Microworld

Dealing with the *description* and *designing* of various alphabets (our mother tongue or foreign ones) can turn out to be very rich in exploring ideas, experiencing and assigning projects, as well as a flexible way of combining formal and concrete process. It can become the reason for someone to refer to the history of Typography, the various family fonts, the way of recognising, processing, representing characters on the computer, up to designing letters with detailed predefined pieces (as well as real objects used by young kids) and the making of banners by the use of floor turtles.

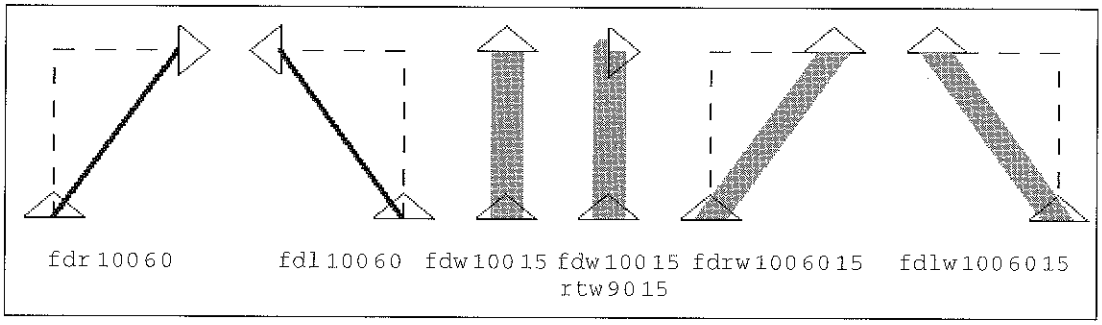
Using this microwold (and its combination with others, such as "*polyspin*" microworld) we can design letters and furthermore words and sentences of any heading and any size regarding their height (:h), length (:m), width (:w), (and depth if we tried to create 3-dimensional letters with perspective), color of the outline and filling of the letter, distance between letters and words, that can be placed in linear, curved or polygonal form. We can also include a "read" procedure to interactively input from the keyboard.

The following procedures represent the basic structural elements of such a microworld:

FDR/FDL: The turtle is moving on the hypotenuse of an hypothetical rectangular triangle (the two inputs define the size of vertical sides) preserving the vertical heading between starting and ending turtle position.

FDW/RTW: A moving/turning command that leaves a widened trace (if the used implementation of Logo does not support a "penwidth" primitive).

FDRW/FDLW: An FDR/FDL command with an additional width (it differs from FDR/FDL in that the ending turtle heading is the same of the start position).



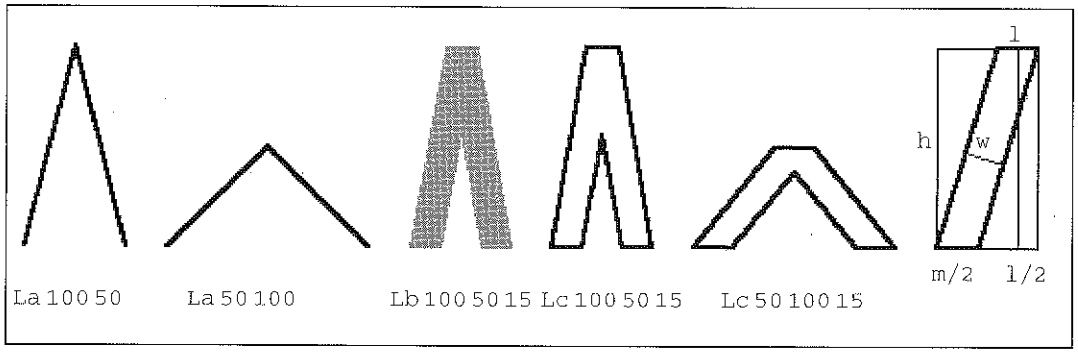
As an example we design the Greek letter "Λ" in 3 different ways.

```
to La :h :m
fdr :h :m/2
fdr :m/2 :h
pu rt 90 fd :m
  rt 90 pd
end
```

```
to Lb :h :m :w
lt 90 bk :w/2 rt 90
fdrw :h :m/2 - :w/2 :w
rt 180
fdlw :h :m/2 - :w/2 :w
pu rt 90 fd :m-:w/2
  rt 90 pd
end
```

```
to Lc :h :m :w
make "l Lambda :h :m :w
fdr :h :m/2 - :l/2 fd :l rt 90
fdl :h :m/2 - :l/2 bk :l lt 90
fdl :h*(:m/2 - :l)/(:m/2 - :l/2) :m/2-:l
lt 90
fdr :h*(:m/2 - :l)/(:m/2 - :l/2) :m/2-:l
fd :l rt 90
end
```

(the general function "Lambda" returns the size "l" of a letter that differs from the width :w).

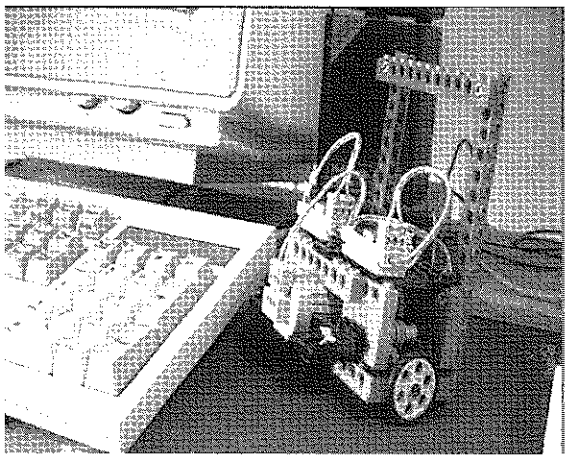


The Lego-Logo Microworld

Dealing with Lego-Logo could be a wonderful experience to study various theoretical and practical subjects. The construction of an original machine, as a final goal, with a combination of hardware & software, make the student feel like an "inventor - engineer". Every student could find, in every large or small project, something that suits his personal interests.

The understanding of automations, the clear distinction between hardware and software, the approach: experiment - evaluation - revision and finally the understanding of a part of our real world are some of the benefits that the student gains.

In small groups of 4-5 students, 9-13 years old, a set of projects are given from which they select one to implement. The available equipment is a portable specially formed work table, a computer connected to the school network, written material and manuals and control functions are also available in BASIC and PASCAL, so that the student can select their preferable programming language.



As an example we can refer to the task of implementing a Lego-Turtle construction, in a test-game form between the students of the same group, who check each other. Each student develops his own program on a computer of the lab and tests his work on the Lego-Logo work table, till he completes the whole job. The score refers to the total behaviour of the used model, containing also the handling process for the achievement of the final goal.

a. The Turtle Moves. The Lego-Logo turtle moves under the commands: XFD, XBK, XRT, XLT that are developed from the students themselves. The distances are calculated in centimetres, the angles in degrees. The vehicle should be moving in the most precise way and the test includes a difficult parking between obstacles that may fall with the lightest touch.

b. The Turtle Learns. The Lego-Logo turtle is being driven through two switches (one for each motor) only forward. The turtle must "learn" the path it has walked along and turn back (itself) to its initial position and heading.

c. The Turtle Avoids Obstacles. The Lego-Logo turtle has two touch sensors to detect the crashes. The test includes a speed race, with a procedure that drives faster to the exit. The initial position and heading was randomly set.

The following procedures (defined by students) are an example of the XFD command and the ESCAPE command that drives the turtle to an exit (it is an interesting problem to study the relations between the topology of obstacles and the turning angle after a crash)

```
to  xfd :distance
    fd :distance
    talkto [a b]
    seteven
    on
    wait :distance * 1.6667
    off
end
```

```
to  escape
  listento 6
  ifelse sensor? [xbk 2 xrt 50] [xfd 1]
  listento 7
  ifelse sensor? [xbk 2 xlt 50] [xfd 1]
  escape
end
```

Some of the microwold functions can be combined with other microworlds. For example in combination with the alphabet microworld the Lego-turtle can design letters on the floor.

The "Speaking" Turtle Microworld

Beyond the peripheral connections with specialized hardware (eg. floor turtles, lego-logo) working on Logo was limited until recently to its computational environment. With the development of multimedia this computational environment is expanded, enriched with the ability to include hypertext capabilities, to use colorful images, to drive any kind of devices (eg. CD-ROM, videodisks) and to control audio & video sources and functions.

Maybe, the first step in the construction of a multimedia microworld is to translate the written turtle commands, partly or wholy, into a verbal form. We create the sound files related to their respective commands or procedures and either Logo demands us verbally to execute turtle commands or we hear them being executed.

The following procedures represent the basic structural elements of such a microworld (based on MCI):

numtoword: A given number as input is put into words (eg. the command: `numtowords 123`, returns: [εκατόν είκοσι τρία] or `[100 20 3]`). This output can be related to respective sound files to produce the verbal form of the number.

say: A given instruction as input is converted in its verbal form (eg. the command `say [fd 10 rt 90 pu fd 100 fill]`, return its verbal form: [μπροστά δέκα δεξιά ενενήντα πάνω πένα μπροστά δέκα γέμισε]).

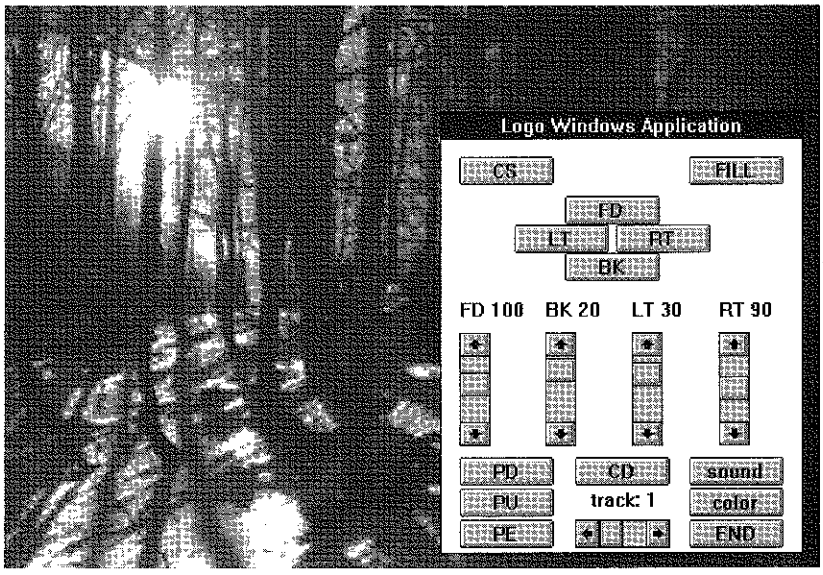
startcd/playcd/stopcd: Starts, plays or stops a specified track (or time lenght) as input, on an audio compact disk.

playwav: plays the sound sample of the specified file, as input (in "wav" format).

The above procedures can fully be driven from a graphic user interface (GUI), an example of it is shown in the figure below.

We can take an example of a "*multimedia turtle game*":

> A turtle, lost in the woods, is trying to find her nest. It can be moved by means of known commands (fd, rt, etc.) but can not see the location of the nest. With each move an overflying bird reveals the distance from turtle to nest but not the heading. The objective of the game is to lead the turtle to her nest by the fewest possible attempts (there is a unique solution with a precise number of attempts).



A Logo-Based Curriculum (4th - 8th grade)

Hours	Subject (4th grade)	Objective
1	Lab Operating Instructions	name parts of computer identify groups and codes, connect to the network, use basic keys
2	Basic Turtle Commands (I)	move, turn the turtle, estimate (elementary) distances, angles
2	Basic Turtle Commands (II) Picture Management	use penup, pendown and movement to any position color, fill the shapes and the background load, modify, save pictures
4	Drawing Project (I)	draw,maintain, print a complete picture analyse (elementary) the stages for the construction use different shapes and stamps
2	Basic Geometric Figures	draw squares and rectangulars, estimate distances, angles (of 90o) protect pictures from unwanted changes
2	Drawing Project (II)	draw freely a project related to a season, adding small text
3	Project with Preparation	prepare the desired drawing and shapes precalculation of distances (the EEC-Flag project)
4	Repetition	draw regular polygons and circles (using repeat)
4	Animation	create animation using different shapes as turtles create animation using pen exchange technique
2	Writing, Typing a Text	write, modify small texts on the pictures use editing keys and punctuation marks
2	Procedure	introduce to name, write and execute procedure for simple shapes

Hours	Subject (5th grade)	Objective
2	Turtle Commands	draw simple shapes, use pen-background colors erase, stamp, fill, shade and use of different shapes
2	File Management	load, modify, save, print pictures use function keys
2	Turtle Project	choose, describe, implement pictures
2	Drawing Geometric Figures	draw triangles, squares, rectangulars
4	Repetition	draw regular polygons and circles (using repeat) understand "total turtle trip theorem" choose and implement a given composition of polygons
2	Animation	create animated graphics (using different shapes and repeat)
2	Writing, Typing Text	use word processing capabilities to add texts on the picture
2	Production of Drawings	use interactive commands and record them on worksheets
2	Procedure Definition	use the editor to name, write and execute a procedure mark and copy the interactive commands into a named procedure
2	Debugging Process	detect "errors" of procedures, edit and correct them (using slowturtle)
2	Editor Exploitation	full use of editor capabilities to write and modify procedures
2	Modular Analysis	choose, describe a drawing on a worksheet implement by stages
2	Super/Subprocedures Building	write procedures that call others

Hour	Subject (6th grade)	Objective
4	Procedure-Subprocedure Building	create, test, modify a procedure use it as subprocedure within another procedure
4	Procedure Combinations	use student completed procedures and combine them to create regular geometric figures and free designs
4	Turtle Geometry (with Euclidean Geometry Elements)	familiarize with heading identify themselves with turtle-body discover the relation between number of sides and angles of shapes identify similarity, symmetry identify divisors, multiples, complementary/supplementary angles experiment with circles and arcs
4	Animation - Sound Project (Multiple Turtles)	draw using predefined shapes enrich drawing with animated graphics and sounds
4	Procedures with Variables	comprehend the use of variables and their effect on making flexible procedures
4	Variable Shapes Building	experiment with increments and decrements (grow and shrink) in parametric procedures

Hour	Subject (7th grade)	Objective
4	Alphabet Microworld (I)	use and combine elements of the Turtle and Euclidean Geometry find out the "symmetry" and the "similarity" as notions, by incrementing and decrementing patters, maintaining the proportions
4	Alphabet Microworld (II)	understand effortlessly mathematical notions and numerical correlations by varying and trying different variables in given procedures
4	Microworld Enrichement	create similar or different procedures by using the given Microworld and by having understood its philosophy extend its possibilities and enrich it according to their will and imagination
2	Analysis - Synthesis	analyse a given pattern, a group of patterns or a drawing, separating them in smaller pieces, create procedures which produce these pieces unite them properly to produce the whole (eg.theater project)
2	Approaching through Different Ways	use given or self made procedures compose a whole, being approached through different ways (imagination, personnal judjement) during the composition, even if they begin with the same procedures
4	Recursive Procedures	be introduced to the notion of recursion do research on the numerical and harmonious correlations of the produced graphics realize the level of one's own intervention in the variety of the produced patterns
4	Control Structures Position of Recursion	work on the continuous repetition, reproduction of patterns, understand the need of controlling repetition, recursion apply the recursion on other problems (such as the motion in the natural environment)

Hours	Subject (8th grade)	Objective
2	Basic Commands Procedure Building	review of Logo environment, commands and functions distinction between interactive and procedural environment
2	Procedures with Variables	create, test, modify a procedure with variables distinction between procedures that returns or not values
2	Tail Recursion N-gons (I)	understand the notion of tail recursion analyze and use the N-gons procedures, estimate the value of variables
2	Control Structures N-gons (II)	use tail recursion to produce complex shapes from elementary ones use if as control structure, rotate and center N-gons (proc: Spingon)
2	Parameters of Recursion	identify and recognize the basic N-gons that produces a complex one
2	The "Angle Guess Game"	estimate the angle of an incrementing recursive N-gons (with score 80% for the first prediction & 95% for the last one)
4	Polyspin Microworld	use the procedures of the microworld combine "initial pattern" & "generation rule" to produce a complex one choose a shape, describe the stages to produce it, implement the final procedure
4	Embedded Recursion Recursive Trees	identify the tail & embedded recursion (variables, control structures,...) create proper stop conditions describe in details the binary tree procedure describe the effects of using different parameters in tree procedures
2	Recursive Levels	describe the effects of using different parameters in recursive patterns create synthesis of recursive patterns
2	List Processing	be introduced to list processing, basic commands identify and perform elementary list operations
2	Lists & Recursion	use recursive procedures to define, search, sort, change a list
2	"Knowledge Trees"	describe, organize & exchange knowledge, having the form of binary trees

Remarks

- The computational environment consists of personal computers with color monitors (DOS & WINDOWS based PC-386) connected to a network via a server that manages and shares the available hardware & software resources (LogoWriter 2.0, TC-Logo 1.0, MSW-Logo 3.3, etc).
- Students work one hour per week in groups of three, aiming to a better result (rarely do their work in couples which has proved to have created difficulties).
- The presentation of learning material takes place in the first quarter of the teaching hour. Students use the remaining time to deal with the given tasks, using their worksheets accordingly.
- The evaluation of their progress is achieved through both their worksheets and the results of their work on the PC.
- The teacher takes an active part throughout the entire process, filling in, skilfully, the lack of personal contact while students get oriented to the PC.

Acknowledgements

We would like to thank Y. Agellakis for the literary care of this paper, E. Marouka for her contribution in the Alphabet Microworld, our students P. Christeas & V. Tsaousopoulos for the multimedia support and Th. Kontonatsios for the telecommunication support, during the preparation of Eurologo '93 Proceedings.

BIBLIOGRAPHY

Harold Abelson, Andrea DiSessa (1981), *Turtle Geometry, the Computer as a Medium for Exploring Mathematics*, MIT Press

Michael Burke, Roland Genise (1987), *Logo and Models of Computation, An Introduction to Computer Science*, Addison Wesley

Eduardo Calabrese, ed (1991), *Proceedings of the 3rd European Logo Conference, Eurologo 91*, A.S.I. Parma

James Clayson (1988), *Visual Modeling with Logo, A Structured Approach to Seeing*, MIT Press

Andrea DiSessa, Celia Hoyles, Uri Leron, Seymour Papert (1985), *Logo 85 Theoretical Papers*, Massachusetts Institute of Technology, Cambridge, July 1985

K. Dunkan, D. Harris, eds (1985), *Computers in Education, Proc. of the IFIP TC 3 4th World Conference*, North-Holland

Paul Goldenberg, Wallace Feurzeig (1987), *Exploring Language*, MIT Press

Dennis Harper (1989), *Logo Theory & Practice*, Brooks/Code

Brian Harvey (1985, 1986, 1987), *Computer Science Logo Style*: Vol 1: Intermediate Programming, Vol 2: Projects, Style and Techniques, Vol 3: Advanced Topics, MIT Press

Celia Hoyles, Richard Noss, eds (1986), *Proceedings of the 2nd International Conference for Logo and Mathematics Education*, University of London Institute of Education, London

Yannis Kotsanis, Constantinos Doukas, eds (1991), *Proceedings of the 1st Conference on Educational Computing*, Athens, Doukas School

Robert Lawler (1985), *Computer Experience and Cognitive Development, A Child's Learning in a Computer Culture*, Ellis Horwood.

Seymour Papert (1980), *Mind-Storms, Children, Computers and Powerful Ideas*, Basic Books, New York, (Greek Edition: *"Νοητικές Θύελλες"*, 1991, εκδ. Οδυσσέας)

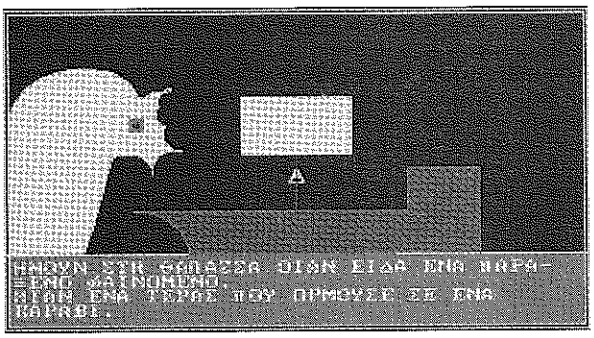
Gilberte Schuyten, Martin Valcke, eds (1989), *Proceedings of the 2nd European Logo Conference, Eurologo 89*, Gent

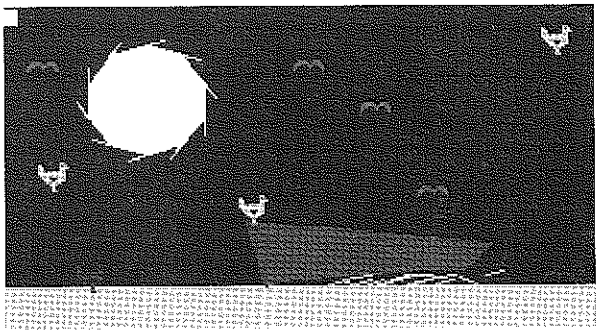
Gilberte Schuyten, Martin Valcke, eds (1990), *Teacher Education in Logo-Based Environments*, Commission of European Communities

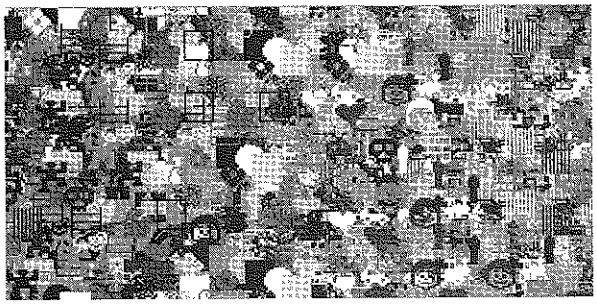
Cynthia Solomon (1986), *Computer Environments for Children: A Reflection on Theories of Learning and Education*, MIT Press

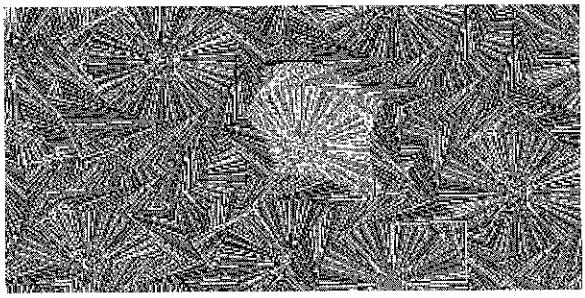
Tipps S., T. P. Riordon, G. Bull (1984), *Nudges IBM Logo Projects*, Holf Rinehart & Winston

Sylvia Weir (1986), Cultivating Minds: *A Logo Casebook*, Haper and Row

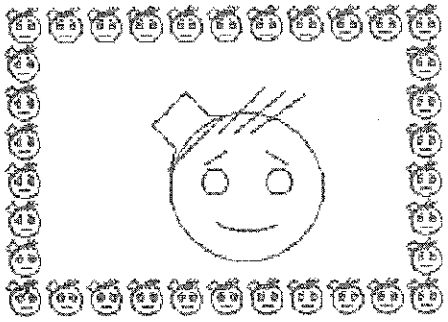


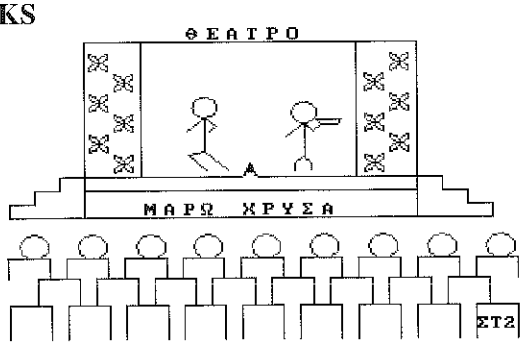


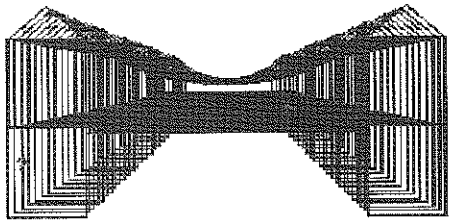


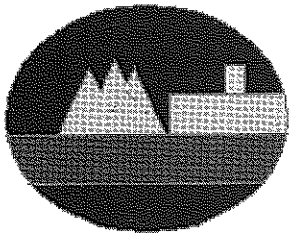


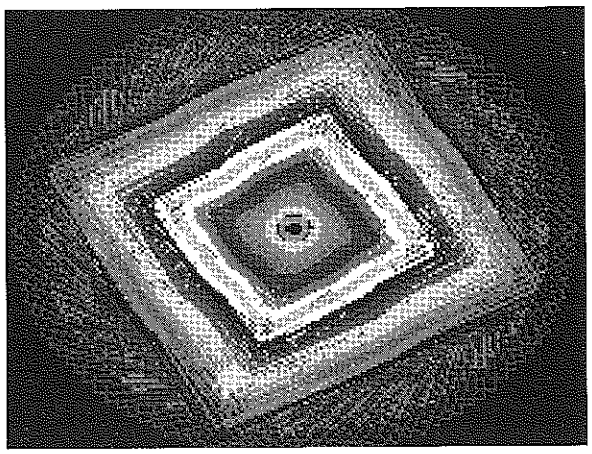
SAMPLE WORKS

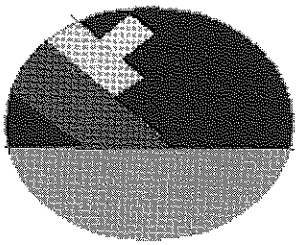












Informatics in Hungarian Public Education Logo Environments in Primary Schools

Martha Kórös-Mikis[1], **Károly Farkas**[2]

[1] National Institute of Public Education
H-1051 Budapest, Dorottya u. 8.
Hungary
tel: (+36-1) 189-1895

[2] Teacher Training College
H-1126 Budapest, Kiss J. altb. u. 40.
Hungary
tel: (+36-1) 178-5820, fax: (+36-1) 156-3482
e-mail: h4604mat@ella.hu

Abstract

In Hungary a new National Core Curriculum is elaborating, in which informatics appears as a new cultural blocks. The greatest novelty is the liberalism and the decentralization, that will end the era of the rigid curricula obligatory for all schools equally. The knowledge of computers and the Logo-environment may be taught not only in a few experimental schools, but in all schools when the conditions are disposal. At present in more than 100 primary schools Logo-microworld is already known and applied in the education for 6-10 year old children. The purpose of the Polya's school of thinking and the heuristic tutoring, formatting of the system theory can be achieved with Logo. The computer in the primary school is first of all the tutoring device in the teaching of reading and writing.

Keywords: curriculum, playful informatics, LEGO-Logo, personification, mother tongue

Martha Kórös-Mikis *graduated at the Eötvös University Budapest from mathematics and physics in 1975 and got her doctor's degree at the same university in 1987. She started her career as a teacher in mathematics in a secondary schools. Three years later she joined the staff of the Research Institute for Electronics where she was involved in the planning of electric circuits by computer. Since 1984 she has been researcher of the National Institute of Public Education. Her main interest is the application of computer science, informatics and technology in education.*

Károly Farkas *is educator with strong backgrounds in engineering, with a doctor's degree of the university. Having gained a profound one-decade-long experience in the industrial engineering, he assumed his activities in college education. Ten years ago — for the first time in Hungary — he started teaching informatics to youngster. Presently more than two hundred primary schools in Hungary are engaged in informatics studies, while the tutors have been integrated into the "Playful Informatics" professional team, which was established and is guided by him. He takes an active part in international research in this field as a member of several research groups.*

Informatics in Hungarian Public Education
Logo Environments in Primary Schools

Informatics in the curricula

Towards the end of the eighties - parallel to the extensive spread of school-computers - more and more demand emerged to teach informatics in the public education as an independent subject. Many alternative pedagogical programs came in being and several schools asked for permission to use their own individual teaching program, so **computer education may be different in its content and profoundness from school to school at present**. There are examples where computer education starts in a playful form already in the first grade (6 years old) for 1-2 hours weekly, while elsewhere the new, independent subject is introduced only at grade 6-8 (12-14 years old) and taught intensively.

In 1989 the National Institute of Education published the **"Technology-Informatics modular curriculum"**. It broke radically with the rigid, central curriculum prescribed for everybody. This curriculum considered the independence of the school, the local conditions (personal and material conditions, the expectation of parents etc.). The real curricula are elaborated by the teachers themselves from the suggested and available modules, further they may create new modules too. In this way individual initiatives may gain ground and thus the technical development can be strictly followed. This modular curriculum also includes, in addition to computer or computer science, also informatics (or information technology) in order to provide the pupils with a wider knowledge. The teaching of all these was suggested already at the end of the eighties, when PC-s and other instruments of informatics (e.g. telefax, parabola aerial, CD-ROM, etc.) practically didn't exist in the public educational institutions, schools!

The change in the Hungarian political system let its effect be also felt in the field of education. **The centralized educational system gradually disintegrates and becomes decentralized.** The situation that the schools submit individual educational programs differing from one an other in their contents and structures to the Ministry and ask for permission resulted in a great confusion. The true independence of schools would be if the "modular-principle" were extended to all the subjects taught. The Ministry of Education realizing the difficulties initiated the elaboration of a **new National Core Curriculum**. (The elaboration of the new Act of Education is going on at the same time.)

The aim of the National Core Curriculum is the diffusion of the universal human values. It is a "frame" curriculum, because it describes what knowledges, capabilities and abilities should be acquired by everybody. It is "national" on the one hand in the sense that is superior to the political interests, on the other hand intends an important role for the national traditions, including the promotion of the culture of the ethnical minorities living in the country. The school-maintainers (local authorities, churches, private persons) and the schools themselves elaborate the real, local curriculum. The National Core Curriculum lists not only the so-called traditional subjects, but cultural blocks. These are as follows:

> 1. mother tongue, 2. foreign languages, 3. mathematics, 4. communication and informatics, 5. natural sciences, 6. man and society, 7. arts, 8. technology and culture of environment, 9. manner of life, family, household, 10. physical education and sport.

It is perceptible that **informatics appears as a new cultural block**. The curriculum recommends to teach the knowledges of informatics primarily together with technology. The subject matter to be taught to everybody isn't distributed among school grades and number of classes, but distinguishes only two levels:

- the primary level for the 6-12 age group,
- the basic level for the 13-16 age group.

The requirements of the "Informatics" blocks are the acquirement of the right attitudes towards the use of information technology and the cognition of the information systems, methods and instruments. At primary level the knowledge of handling different media and software based on the traditional and the new electronic technologies is important. For example, it excepts handling of the catalogue in the school-library, making notes, keeping a reading diary and also handling and operating the school-computer with aid, e.g. learning with didactical software or **working with text, graphics, animation, controlling electrical toys. For that purpose the Logo-environment is especially suitable.** [3]

Fortunately, the basic level breaks totally with the teaching of program languages which became a habit for many years. Now the users' knowledge is dominating: the promotion of problem-solving by using the offers of information technology, such as storing, processing, transferring and retrieving of information; the use of electronic instruments for the purpose of communication (telephone, telefax, video-technics, computer-networks, terminals, etc.). Teaching of program languages and programming is the task of the professional training. As far as the general intelligence is concerned, only recognition and wording of problems and creating of algorithm are expected.

The discussion about the new National Core Curriculum is now under way. The Ministry of Culture and Education by considering the opinion and critical suggestions of teachers, researchers, teachers' union will take steps to introduce of new curriculum. This requires the elaboration of model-curricula and the organization of extension training courses for teachers. The new curriculum will be introduced gradually, the earliest date probably will be the 1994/95 school year. The greatest novelty will be the liberalism and the decentralization, that will end the era of the rigid curricula obligatory for all schools equally. The "first cracks" on the central curricula were brought about just by the appearance of computers and the demands for informatics education. [5]

Informatics in primary schools

In 1985 a program was started in a few primary schools in Budapest in order to find out how Logo can be used in basic education. At the EuroLogo Conferences held in Gent and Parma we have already reported on our first experiences [2, 3]. At present in more than 100 primary school Logo-microworld is known and applied in the education for 6-10 year old children. The number of PC increases substituting the old home-computers.

We would like to show some of our newer results aiming to develop the ability of young children by the aid of informatics and computers:

- Logo-novelties
- Lego-logo connection
- computer based teaching of the mother tongue.

We call the developed method "Playful informatics". **The essence of our method is neither computer nor the informatics proper, but the modern humane pedagogy.** Our aim is to give pleasure while teaching informatics. We want to make children wonder, to develop their talents using their interest. Our method is based on Papert's pedagogy, on George Polya heuristic, on Ludwig von Bertalanffy's system of views and on John von Neumann's theoretical foundations, perceived via computer. Our pedagogical ideas in many aspects coincide with the principles expressed by Gordon, Frenet and Rogers. We in our attempt to establish the child-centered pedagogy have always been very keen on using the newest and the most sophisticated technical equipment. For example, Frenet's school printing house and all the mail exchange among the school are being substituted for by the computer-based word-processing. Another example some lectures of Gordon have been illustrated and supplemented by the videos, prepared by the students themselves. We believe the purposes of the Polya's school of thinking and the heuristic tutoring, formation of the system theory can be achieved with the assistance of Logo environment.

However, we have added some details to the Papert's pedagogy. What are these details? According to Papert, the main hero of Logo, its ultimate mean of motivation, is the turtle. Although this nice fellow indeed knows the way to the children's inner world, we still believe that a lot of children rather stick to the idea of a robot. A real turtle, naturally, won't think of obeying our orders, while a FLOOR TURTLE is always faithfully at our service. Showing how it works is a great deal of help. Here the motivation takes its source. COMPUROBOT, ROAMER turtle or the lego cybernetic turtle can move in the three-dimension space (3D). **These operations with the TURTLE and the whole technical system is the foundation of our conception**, while the expected aim is to learn (playfully) how to use the technical environment constructively. Besides, we feel important to enhance the usage of this technical equipment as a major mean of "detaching" pupils from the computer display, which sometimes seems a rather desirable solution.

According to Papert, very often the screen turtle is introduced by the teacher, who shows how to issue commands to it. We do this step last. Before we play with floor turtle and we play "robot game". Robot game is an acting. Few pupils or the whole group, including the teacher personify the robot. The commands are given with the help of certain set of instructions, depending on the level of the group. At the beginning only the mother tongue is in use, then gradually we begin to apply some formal commands thus approaching the Logo's syntax and semantics. The personification of the turtle can be varied by playing different games, arranging competitions etc. Moreover copying the different generations of robots children will acquire some general technical knowledge, too. In other words, the Papert's ability-developing pedagogy's effect is so completed by some very useful additional technical education. In this sense the games MATCH-Logo, Lego-Logo and "Feed the turtle" are very instructive.

In the first version of "**Feed the turtle**" the turtle moves in the square playfield painted on the floor. The child that is personifying the turtle is directed to the place of "feeding" by other children, who sound different Logo commands. This game proved very useful in training the algorithmic mentality and the sense of position and direction. On the next stage, children move the figures they have made themselves in the TURTLE GARDEN, then on squared paper. At this point a child actually abandons the game proper, making a decisive step from the real to the abstract. Then the child is able to copy the trail of the turtle in the copybook. At last the child is able to direct the turtle in its garden on the display of the computer.

Our game **MATCH Logo** in its first version presumes that the child moves in the 3D space itself. The personified robot is able to proceed any further on the imaginary swampy ground only after it has "paved" its way and only in the direction where the paving elements have been laid. These paving elements, pointing arrow-shaped, have been cut out of paper approximately the size of a foot. On the route built of these elements the algorithm of motion can be transparently seen, since the arrows represent the vectors.

The smaller measurement microworld version of the game includes building the route (of various shapes) from matches. Drawing of the route will demand from the child an even higher degree of independence. The final version of match Logo visualizes the game on the display. This is mainly intended to help the children practise their communication skills with the computer.

Just in the very similar way of playing with the real building blocks, or elements, the **Lego-Logo** game begins. Crawford Craig provided the instructions how to use the Logo language in this activity. (Lego-Logo. Micromath. A Journal the Association of Teachers of Mathematics. 2/1 1986. Oxford.) These manipulations help to develop a good extent of dextrousity, of visual promptitude and a certain algorithmic ability affecting the child simultaneously and in interview mutual-enhancing fashion. We can enjoy the process of building using only simple Logo primitives in the mother tongue. A special virtue of the game is that we can operate in the 3D space. This method comes in very handy if the idea of symmetry is to be explained and it can assist well in illustration of the structuring of creative work. First the constituent parts of the construction - which are prefab single- colored bricks - are put together, then these obtained modules are assembled. More elaborated algorithms are to be found in the lego DACTA teachware. By having developed these didactical means considerably, the lego enterprise have really deserved the worldwide success it enjoys. For the purpose of practicing the human-machine dialogue we have developed the computerized version of our lego-Logo toys. The **program LogoDRIO** is our creation, it is able to show an axonometric image of our construction on the screen.

The manipulation games thus easily separated from the computer usage, can be followed by the appearance of the computerized version. The computer, in our opinion, is first of all the tutoring device in the teaching of reading and writing. According to our hypothesis children can master the art of writing - first typing, and then handwriting! - and reading more easily, early, better and with more tangible success with assistance of a computer. The **computerized tutoring of reading and writing** helps children to prevent and overcome dyslexia, dysgraphia, to attain a good orthographical knowledge, to develop their aesthetical and reading abilities.

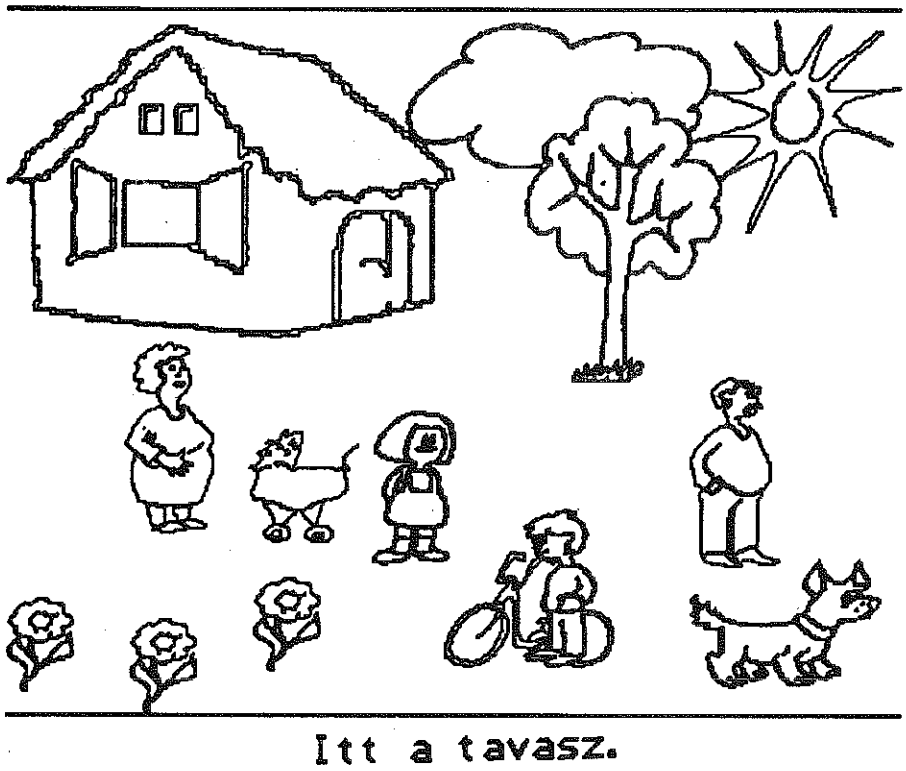


Fig. 1. 6 aged child's work with computer

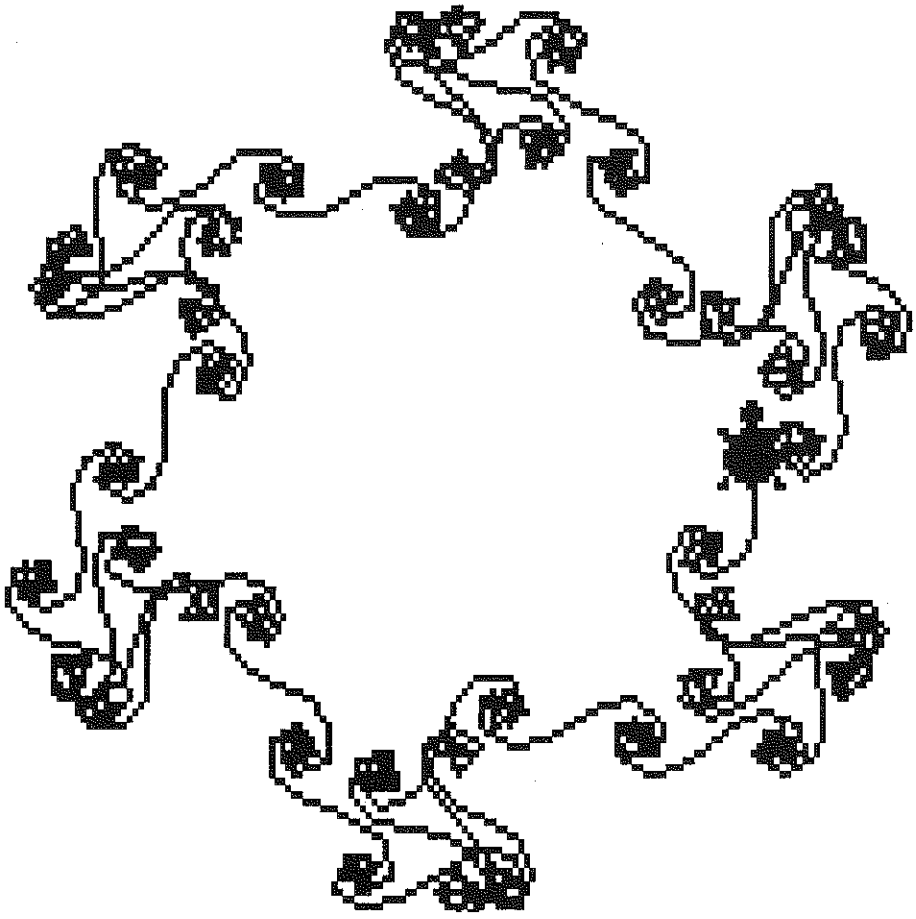
Already in kindergarten a computer and any synthesizer helps to teach **music and singing**. As an example we're going to show the program that will teach children to play a given tune on the keyboard of the computer.

One of the versions of Logo is our so-called **OVI-LOGO** (in English "infant-Logo"). This is a version of the well-known One-key Logo. The Marta Logo of Eduardo Calabrese (Eduardo Calabrese: Marta - the "intelligent" turtle. Universita di Parma / Second European Logo Conference 1989. Gent) have been adapted for the language of LogoWriter. This is a favorite game program in our experimental kindergarten. The process of drawing, controlled from the computer keyboard, is executed by a realistic top view of a helicopter with rotating propeller screw (instead of the turtle) Then, after finishing drawing, the helicopter can automatically follow the lines of the ready picture.

After having played these games and having had some practice in dealing with our toys, after having attained some basic training in communication with the computer, we may start getting acquainted with the Logo language in its "kis-Logo" (in English little-Logo) adaptation.

"**Kis-Logo**" is a Hungarian version of Logo for the 4-7 years old. We consider it an essential addition that we have increased the units of the step and the turn. So little children don't have to work with big numbers,

and they can relate the directions to the clock face. Older children can profit from the Logo language using it mainly as a tool of mathematical discoveries. Our game the "**Turtle roses**" for example serves as a means of artistic interpretation of the numerals, quantity, primary number and relative primaries. [1]



REFERENCES

[1] Farkas, K.: Informatics games for developing children's thinking ability. IFEP OPEN CONFERENCE "Informatics and Changes in Learning" June 7-11, 1993. Gmunden, Austria. Editors: Brin Samways - Dave Johnson.

[2] Farkas, K.: How we teach Logo in a Budapest school. Proceedings Second European Logo Conference 1989. Gent, Belgium, 30 August - 1 September.

[3] Farkas, K. - Kórös-Mikis, M.: Informatics games for developing children's thinking ability. in: Proceedings Third European Logo Conference, Parma, Italy, 27-30 August 1991. pp. 367-376.

[4] Kórös-Mikis, M. - Panuska, R.: Pictures with Symmetry and Recursion, in: Micromath, Journal of the Association of Teachers of Mathematics, Oxford. Vol.3. No.3. Autumn 1987.

[5] Kórös-Mikis, M. - Tompa, K. - Vári, P.: The presence of computers in the Hungarian public education. in: Computer and Education, under publication.

From Programming to Computer Science: Oppurtunities and Pitfalls

Tammy Lapidot, Dalit Levy

Department of Education in Technology and Science
Technion - I.I.T., Haifa 32000, Israel
e-mail: ttr0148@technion.bitnet

Abstract

This paper is trying to challenge a common trap, of focusing on programming aspects rather than on Computer Science ideas. It offers a didactical strategy, called "the rich assignment activity", that is based on the interconnected-learning of concepts. This strategy is demonstrated via the DATE assignment. The paper ends with a short discussion from three perspectives: the subject matter, learning and teaching.

Keywords: Logo, didactical process, programming, data assignment

Tommy Lapidot *and* ***Dalit Levy*** *have M.Sc. on Computer Science Education. They work at the Israel Institute of Technology in Haifa, Israel. They are heading a team which is developing and implementing high school Computer Science courses for the Ministery of Education and Culture in Israel. For the past 5 years, this team has been working on Introduction to Computer Science courses via Pascal and via Logo. During that time, the team have gained a substantial amount of experience with high school students, teachers and curricula.*

From Programming to Computer Science: Opportunities and Pitfalls

Introduction

It is widely accepted that programming is an excellent vehicle for learning Computer Science, because among other things it has "a significant effect on shaping a student's view of computing" [4]. Yet, experience shows that although the major focus should be on Computer Science ideas, it is tempting and easier to remain on the programming level. How then, can we avoid falling into this 'trap', and keep the focus on major concepts and principles?

In this paper, we suggest a "Didactical process" [3] that has been tried out and refined through our experience with students and teachers. We call it "the rich assignment activity". It is organized around carefully selected programming assignments, and is based on the interconnected-learning of concepts. Such assignments invite a rich variety of solutions, where each solution can give rise to Computer Science ideas.

The Didactical process

Working on a solution in small groups

The activity starts when the teacher presents the assignment and asks the students to work on it in small groups. As an example, here is the DATE assignment: Write a procedure that accepts three integers (month, day and year) and checks whether these numbers can represent a legitimate date in the 20th century. (Assume that in each month there are exactly 30 days.)

While the students are working (with or without computers), the teacher walks around in order to collect their solutions. At the same time, the teacher gains an excellent opportunity to talk to individuals, locate misconceptions or powerful ideas and ecourage different solutions.

Analysing different solutions in a whole-class setting

Afterwards, the solutions are presented and discussed with the whole class. But instead of a random order, where each group 'jumps' and offers its solution, we suggest a carefully chosen order of presentation. In other words, we can gain more by organizing the order in which the solutions are to be presented: from simple algorithms to more complex ones; from problematic solutions to "better looking" ones; and from simple concepts and principles to more sophisticated ones.

For example, the DATE assignment can give rise to the following concepts and principles: programming style; procedures, functions and abstraction; imperative vs. functional approaches; "linearity" vs. nesting; efficiency and readability; different branching strategies; meaningful names; the flag metaphor; boolean expressions and boolean functions; tricky solutions.

How can we expose the full potential of each solution? This is the time to suggest one hypothetical scenario.

Solution number 1 - **Deep nesting**

This solution is very common among different groups of learners. Yet, it is hard to read, understand and debug.

Even if students don't suggest this solution, it is worthwhile starting with. It demonstrates many disadvantages, and serves as a good basis for our didactical process.

FIGURE 1 – FOR SOLUTION NUMBER 1

```
To Date :Year :Month :Day
If and :Year>1899 :Year<2000 [ If and :Month>0
:Month<13 [ If :Day>0 :Day<31 [ Print [This is a legal
date]] [ Print [Day is NOT legal]]] [ Print [Month is NOT
legal]] [Else [Print [Year is NOT legal]]]
End
```

The teacher can even put an error (omitting one of the] , for example) in order to demonstrate the problem of unreadability. This is what we call "a didactical trick". The trick lies in the fact that actually students don't care about modularity, readability or other aspects of style, but such a 'didactical bug' forces them to confront these aspects.

If you work with a version that allows indentation, this is a good time to discuss the next variation of solution 1.

FIGURE 1 – FOR SOLUTION NUMBER 1

```
To Date :Year :Month :Day
If and :Year>1899 :Year<2000 [ If and :Month>0
:Month<13 [ If :Day>0 :Day<31 [ Print [This is a legal
date]] [ Print [Day is NOT legal]]] [ Print [Month is NOT
legal]] [Else [Print [Year is NOT legal]]]
End
```

The power of indentation lies in the visualization of the algorithm's structure. It enables you to immediately-recognize modules and to capture the feeling of deep-nesting. But, as Allen et. al [1] mentioned, one problem with conditionals is that

FIGURE 3 – FOR SOLUTION NUMBER 1

```
complex conditions turn into
long nested
conditional
expressions
(one test
inside
another test
inside yet
another test).
Much like
the mouse's tale, they
are hard to
read and hard to
follow.
```

Because of such "anti-social" programming style, we continue with the sequence towards better style conventions.

Solution number 2 - **Tail nesting**

FIGURE 4 – FOR SOLUTION NUMBER 2

```
To Date :Year :Month :Day
  If Or :Year<1900 :Year>1999
        [ Print [Year is NOT legal]]
        [ If Or :Month<1 :Month>12
              [ Print [Month is NOT legal]]
              [ If Or :Day<1 :Day>30
                    [Print [Day is NOT legal]]
                    [Print [This is a legal date]]]]
End
```

Since this solution also uses nesting, it has almost the same disadvantages. Still, using the opposite logical structure makes it a little bit easier to read, follow and understand.

This is the teacher's first opportunity to compare different solutions, and to encourage discussion on this subject. There are (at least) two reasons why it is powerful to consider different solutions. From the individual's point of view, it is fruitfull to 'brainstorm' on different directions, before going into the messy details of the solution that happened to jump first. From the social point of view and due to communicational considerations, it is important to be aware of different thinking styles.

We have chosen to name this solution 'tail nesting', because it reminds us of tail recursion. i.e, tail nesting relates to deep nesting in a similar way that tail recursion relates to deep (real) recursion (Of course, recursion is much more powerful than nesting). Both 'tails' are easier to understand, learn and teach than the 'deeps'. The 'tails' involve lower intellectual complexity because one doesn't have to remember so much or bother with the stack management.

Solution number 3 - **Abstraction**

FIGURE 5 – FOR SOLUTION NUMBER 3

```
To Date :Year :Month :Day
   If And :Year>1899 :Year<2000
      [ Check_month_and_day :month :day ]
      [ Print [Year is NOT legal]]
End

where Check_Month_and_Day is a sub-procedure
that checks the validity of month and day.
```

This example can demonstrate quite well how "we control complexity by building abstractions that hide details when appropriate" [1]. The power of the abstraction methodology, is that "rather than make programmers the slaves of the machine [...], we can solve our programming problems using whatever instructions are natural to our way of thinking" [6].

One way of introducing this solution, is to construct it from the former ones. This can be achieved by identifying modules, boxing and naming them. Obviously, using abstraction in the first place, could spare us from the 'painful path' of solutions 1 and 2. Our recommendation to postpone the abstracted solution, is based on three considerations. First, in most classes it is unlikely to receive abstracted solutions right at the beginning. Secondly, the power of abstraction is better perceived on the basis of the first two. Thirdly, abstraction is a more advanced concept than the formers.

The power of abstraction gains extra strength when recurring it. For example, we can use it for the predicates (conditional expressions) and get something like this:

FIGURE 6 – FOR SOLUTION NUMBER 3

```
To Date :Year :Month :Day
  Test year_is_valid? :Year
  Iftrue [ Check_month_and_day :month :day ]
  Iffalse [ Print [Year is NOT legal]]
End
```

Notice that we used meaningful names for the procedures. This statement hides a variety of interesting issues: What is a meaningful name? To whom is it important? Are there any guidelines for choosing a name? Is it better to name a procedure according to its content, or according to its role? What is the 'optimal' length of a name?

Solution number 4 - **Breaking the hirarchy**

So far, all the solutions had hirarchial inner-relations. A different approach to the problem would be to stay at the same level, like in the solution below.

FIGURE 7 – FOR SOLUTION NUMBER 4

```
To Date :Year :Month :Day
  If Or :Year<1900 :Year>1999 [Print [Year is NOT legal] Stop]
  If Or :Month<1 :Month>12 [Print [Month is NOT legal] Stop]
  If Or :Day<1 :Day>30 [Print [Day is NOT legal] Stop]
  Print [This is a legal date]
End
```

DATE contains three modules, one per each "IF". But, if we try to use sub-procedures for them, we face a new problem. For example, try to replace the second IF with a procedure CHECK_MONTH, and assume that the value of MONTH is 23 (not valid, of course). In this case, although the sub-procedure prints 'Month is NOT legal', the main procedure still prints 'This is a legal date'.

When we try to break the nesting into a linear flow of control (as suggested), and still want to keep abstraction principles, we face the problem of how to connect the modules.

The issue of communication among modules is another major issue. The imperative approach uses variables for this purpose, while the functional approach treats it with functional-programming techniques. Solution

4a demonstrates the first approach. See solution 5 for the second.

Solution number 4a - **Using a flag**

```
To Date :Year :Month :Day
  Check_Year :Year
  Check_Month :Month
  Check_Day :Day
  Test :Wrongdate
  IfTrue [Print [Date is NOT legal]]
  IfFalse [Print [Date is legal]]
End
where for example, check_year is:
To Check_year :Year
  If Or :Year<1900 :Year>1999 [Make "Wrongdate "True
                               Print [Year is NOT legal]]
End
```

Notice that WRONGDATE is more than just a variable, and plays the role of a flag. We can also use three different flags (wrong_year, wrong_month and wrong_day).

The flag metaphor is a powerful idea in Computer Science. Our experience shows that many teachers don't utilize this oppurtunity, and tend to remain at the technical implementation level, i.e they use only the terminology of 0/1, T/F etc. This is a good example for the 'traps' we have been talking about.

Here we have another opportunity to demonstrate meaningful names, this time for variables. For example, why "Wrongdate" and not "flag", "wrong", etc.?

Solution number 5 - **The functional approach**

An interesting challenge for the students could be: "Try to solve the date problem without the use of branching (if or test)". A solution in this spirit could be (we choose to use free parameters, because it is more convenient here):

```
To Date :Year :Month :Day
 ( Print [It is] legal_date [that the date is legal] )
End
To legal_date
 Output (And legal_year legal_month legal_day)
End
To legal_year
 Output And :Year>1899 :Year<2000
End
To legal_month
 Output And :Month>0 :Month<13
End
To legal_day
 Output And :Day>0 :Day<31
End
```

This solution expands the idea of abstraction, with the use of functions. As a "side effect", we improve the modularity and readability of the solution.

If we use the terminology of 'The Little People' metaphor, and think of the people as working towards a shared goal, this solution reinforces the feeling of mutuality. Each 'function-person' not only receives a request (as in the scenario of solution 4a), but also returns an output. This cooperative quality is perhaps another reason for the flourish of the functional approach in the Logo community.

Students find the use of the direct output somewhat strange. Therefore, we can often see solutions like

```
If And :day>0 :day<31 [output "True] [output "False]
```

which can give us yet another opportunity to discuss major issues.

Solution number 6 - **A "tricky" solution**

We can create new opportunities with an additional sub-task: check whether a legitimate date is during the school year (the school year in Israel starts on September 1st and ends on June 20th).

Some students like to "show-off" with tricky solutions. It is important to emphasize that these solutions can make the program much harder to understand, debug or maintain. Here is one of these tricky solution.

```
To School.year :Year :Month :Day
  Check.school.year :Year*10000+:Month*100+:Day
End
To Check.school.year :Date
  If And :Date > 920900 :Date < 930621
    [ Print [The date is during the school year]]
    [ Print [The date is NOT during the school year]]
End
```

Discussion

This paper is trying to challenge a common trap, of focusing on 'hard' technical aspects while dealing with programming assignments. After the introduction of this didactical problem, there came a suggestion for a didactical process. Finally, we offer a short discussion from three perspectives: the subject matter, learning and teaching.

From disciplinary (Computer Science) perspective, this strategy offers a rich variety of opportunities to deal with major ideas. As a result, there is a potential for developing a more advanced disciplinary understanding, by discussing several aspects of each concept, and focusing on inter-connections among these concepts.

From learning perspective, our experience shows that this kind of activity is interesting and exciting for different groups of learners. Students can flourish in different directions, while developing autonomous and creative thinking. At the same time, it encourages social interactions in the flavour of 'samba schools' [5].

From teaching perspective, this paper offers an alternative setting.
But, are there any guidelines for choosing a 'rich assignment'? Like in many teaching decisions, the choice is based on experience and intuition. Sometimes it just happens: an assignment that looks naive or trivial, turns out to be rich. Consider, for example, the following (potentially rich) assignment: Write a function that receives four numbers, and outputs their maximum.

This paper introduced several solutions to the DATE assignment. What guarantee do we have that some or any of them will rise in reality? Our experience shows that at least the first three are usually suggested by students. Still, the teacher cannot rely on the good fortune of receiving these solutions, nor can she or he hope to receive them at the desired order. In order to get ready for such an activity, the teacher can prepare 'home-made' solutions, and use them if necessary. In the setting we offered, it doesn't really matter who offered which solution.

Finally, we call the EuroLogo community to brainstorm on other 'rich assignments', in order to enrich our didactical collection.

Acknowledgements

We wish to thank Uri Leron, Orit Hazzan and Rinat Rachman for their carefull reading and helpful remarks.

REFERENCES

[1] Abelson, H., Sussman, G.J., Sussman, J. (1985). *Structure and Interpretation of Computer Programs*. MIT Press.

[2] Allen, J.R., Davis, R.E., Johnson, J.F. (1984). *Thinking about TLC Logo - A graphic look at computing with ideas*. Holt, Rinehart and Winston.

[3] Balacheff, N. (1990). Towards a Problematique for research on Mathematics teaching. In *Journal for Research in Mathematics Education*, Vol. 21, No. 4, pp. 258-272.

[4] Denning, P.J., Cormer, D.E., Gries, D., Mulder, M.C., Tucker, A.B., Turner, A.J., Young, P.R. (1989). Computing as a discipline. In *The Communications of the ACM*, Vol. 32, No. 1, pp. 9-23.

[5] Papert, S. (1980). *Mindstorms: Children, Computers and Powerful ideas*. Basic Books Inc.

[6] Pattis, R.E. (1981). *Karel the Robot: A gentle introduction to the art of programming*. John Wiley & Sons.

Conversations with a Computer. Third Generation of Computer Assisted Learning

Chris Robinson

Horndean Middle School
Five Heads Road, Horndean, Waterlooville
Hampshire PO8 9NW, United Kingdom
tel: +329 661565 +705 592236
fax: +705 599453

Abstract

The paper follows the concurrent developments of teaching methodology and computer capability and their interaction in the classroom illustrated with observation of actual practice. The observations identify three generations of Computer Assisted Learning being i) "Directed" in which the computer seeks to inform or control, ii) "Random" in which open-ended software is used in various ways, and iii) "Conversational" in which children are actually involved in a two way interaction with the machine. Throughout, reference will be made to the support intended or delivered to the various areas of the school curriculum. Evaluating the successes and failures of the past and present, the leading role of the Logo computer in this genealogy is identified and the ramifications for future.

Keywords: Logo, primary practice, computer assisted learning, active learning, language, mathematics, adventure games, control

Chris Robinson *at 45 years of age, he is presently deputy head teacher in a large middle school (8 - 12 years) in the south of England. During his 23 years of teaching, he has spent 3 years as an advisory teacher and a year's full time study of children's learning with micro-computers. Through LogoS (formerly BLUG), he is frequently involved in a support and advisory role to practising teachers at courses, conferences and in publications.*

Conversations with a Computer: Third Generation Computer Assisted Learning

Introduction of computers into schools

It is worth remembering that computers have only been widely deployed in schools for a decade and, although many of the fundamental precepts inherent in their contribution to good primary practice are older, much change has occured beneficially to the primary education system through their deployment.

What were the perceptions of educators and children of the machines in the early days and in what ways have they been realised, surpassed or abandoned?

The majority of the observations recorded herein are derived from my own experiences during these years.

The early misconceptions of computers were legion. It was frequently erroneously refered to as an "electronic brain" with the implication (particularly in the minds of the children) that it knew all the answers; all that was required was to ask it the questions!

I recall seeing children typing in questions at computer keyboards only to receive obscure replies such as "Syntax error" or, more succinctly, "mistake"! Is the implication that it is a mistake to ask the computer questions? Has the view been re-inforced that the child is stupid?

Teaching methodology

The introduction and development of the new technology closely followed the moves towards a more practical based curriculum centred upon the observations of Piaget.

A fundamental issue in the provision of an environment which promotes active learning, is group work which many teachers found difficult to manage initially. The introduction of the microcomputer in the school, supported and required this style of classroom management as there certainly were not the resources to enable the acquisition of computers for every pupil.

Piagetian influences were really beginning to hit the UK education system in the sixties. The schools' council, disturbed by low levels of numeracy in school leavers produced a report recommending active approaches for the learning of mathematics [1]: "Professor Whitehead said, 'Every child should experience the joy of discovery'. We can say with complete assurance, every teacher who embarks on a programme which will enable children to make their own discoveries in mathematics will share to the utmost their children's enjoyment of mathematics and their increasing powers." It also provided detailed methods to provide a necessary variety of teaching styles which were later echoed in the Cockroft report [2]: "Mathematics teaching at all levels should include opportunities for: exposition by the teacher; discussion between teacher and pupils and between pupils and themselves; appropriate practical work; consolidation and practice of fundamental skills and routines; problem solving, including the application of mathematics to everyday situations; investigational work." The far reaching Plowden Report [3] advocated a much greater emphasis on active learning in primary schools.

One aspect of group work particularly difficult for traditionalists to incorporate into their repertoire was that of conversation. Whereas an ordered classroom might have previously been judged on whether children were working silently or not, children were now being encouraged to discuss their work with each other.

First generation C.A.L.

The first software used in schools, was usually of "drill and practice" type. Not understanding the new technology, it had to be simple enough for teachers to use. Frequently it was "put on and left" for pupils to get on with, though to be fair to hard-pressed teachers, they were not to blame for this state of affairs.

In his reflective look over the previous thirty years and prophetic look at the next, John Vaizey, in the sixties, predicted [4], "(teachers) will also start to use far more of the new techniques - machines, television and radio - to supplement their own work." but he also indicated the considerable expenditure that would be required for effective development. Unfortunately, although considerable finances were poured into promoting the provision of British hardware in our schools, this was not matched with sufficient resources for the effective training of teachers to use it to its best advantage.

If children were occupied at the keyboard, the teachers felt they must be learning. The computer was a great motivator with pupils keen to play with the new technology and enjoy activities that were often considerably different to the standard "chalk and talk" lessons still being delivered by traditional teachers. Thus a myriad of programs testing children's mathematical or spelling abilities etc propagated the myth that computers were cleverer than children (but would still not answer the questions they asked of it)!

The obvious outcomes should have been predicted:

Those interested in the technology itself (pupils and staff) continued to be drawn to it, ("toys for the boys").
Others realised these tasks were better covered in more convential ways.
Others, drawing from both fields of thought, realised a much more important role for the computer in the classroom.
And others discovered Logo.

Second generation C.A.L.

The Schools Council had provided much practical assistance to teachers attempting to promote active learning techniques. One of their major projects, "Science 5-13" described an "ideal" class situation observed by a visitor as the prologue to its introductory guide book [5] in which many groups of children are observed actively engaged in a variety of essentially open ended practical activities: "most of the conversation came from a couple of groups occupied in activities of their own". (Science was still comparatively rarely found in British schools at this time. In 1983 the HMI Science committee were still attempting to reconcile to teachers the methodology for teaching science [6]. The recently introduced National Curriculum places greatest emphasis on the practical nature of science which is now accepted as one of the three core subjects.)

Pursuing this philosophy and attempting to transfer it to the medium of the microcomputer, it is abundantly easy to see the significance of the adoption of word processors and a clamour for "open-ended" software, which began to include databases, spreadsheets etc and adventure games from which much imaginative and creative work was developed by teachers. These latter programs were actually beginning to react to the child, acknowledging his/her existence and beginning to provide a limited amount of conversation.

One problem, however, was the management of these courses of learning. Too frequently teachers left the children to just type on the wordprocessor, "paint" a picture on the screen or play an adventure game without giving much consideration to the learning that was taking place other than attempting to justify it with the comforting thought that if the children were using the machine they must be learning something! In effect, though, this became a very hit and miss affair in which children followed random paths until they bored of the activity.

Alas, many teachers who had discovered Logo as being their saviour from the horrors of first generation

CAL, were to stop at this point. They had found a medium and a means for addressing the shortcomings of the mathematics curriculum highlighted in the earlier reports mentioned in the turtle graphics aspects of the language supported by their interpretation of texts such as those produced by the Chiltern Logo Project [7]. However, some attempted to reduce this to first generation CAL by providing prescriptive lists of exercises which could be adjudged right or wrong whilst others remained at the random level of second generation CAL by permitting unstructured turtle doodling, totally missing the extra achievable level of third generation CAL.

Third generation C.A.L.

The one recurrent aspect in the concurrent developments of teaching methodology and educational software design highlighted above that has been consistently ignored has been "conversation". To be able to ask meaningful questions of the computer and obtain meaningful answers from it is what children wanted in the first place (see paragraph 4)!

Modern, sophisticated, computers found in our schools should be capable of doing this. The Logo computer is the ideal medium for permitting conversations to take place with children dictating the rules under which conversation occurs and its content.

One of the previous restrictions of using adventure games was the vocabulary used which was impossible to redefine. Frustration often occured when asking the machine a question or giving a command that was not understood. Frequently the "error" messages delivered on such occasions were equally unintelligible. These shortcomings were often excused by devotees by remarks such as, "It's all about problem solving!" With real problems to solve, why provide obstacles under this guise and shouldn't the essence of real learning be problem avoidance: recognising the conditions under which problems are likely to occur and anticipating a need? If that is the case, to avoid a problem, the message is not to use the software which may have dubious educational benefit.

When children create their own adventure games in Logo, however, they decide the language and keep control over the sort of conversations that may ensue. The popularity of adventure games may well be the control the player has over the outcome. Writing their own adventure "games", children control the choices that may be made.

Adventure "games" may present real situations. In teacher training sessions, I usually set up scenarios such as, "You are sitting at the computer feeling frustrated. Do you go for a DRINK or PERSEVERE?" and leave the rest up to the participants (with some help, of course: they may not like drinking alone).

At my school, children have made adventure games modelling historical or geographic environments. For example, when studying a local town, they modelled the task of meeting a visitor at the railway station and showing him the important landmarks using a hand held map and answering on-screen questions such as, "Do you walk down the HIGH Street or SWAN Street?" With the acquisition of still video camera and digitiser, we now have the ability to add "rewards" in the form of views of those landmarks as they are reached.

The language of Logo, used conversationally, is a helpful tool in the development of both mother tongue and modern foreign languages since the user defines the words and responses that may be used.

When using Logo as a control language, the ability to define procedures to react to commands such as, "Turn on the light" has led to much fruitful discussion about synonyms for "TURN ON" and what part of speech and function the word "THE" has, etc.

When working with a group of non-English speaking children, I soon realised the importance to them of being able to define words in their own language meaning "Tell the crane to lift the load"!

Children write "electronic magazines" providing pages of information the reader interacts with, possibly

illustrated and with musical accompaniment, to provide news of the school football team, for example.

Databases of information about "ourselves" are typed in. To find out who the computer knows about, it is only necessary to ask it to Print Out TitleS.

But is this really conversation with a computer or is it a bit contrived?

The computer doesn't really know anything about John Smith in 7C, only a list of otherwise meaningless characters someone chose to type into the machine.

When providing an "information board" about the school, a simple map can be displayed on the screen with a helpful prompt such as, "Where do you wish to go?" Typing the destination can lead the turtle to the destination. When setting up such a system, the children wanted to use Cartesian co-ordinates but did not know what they should be for each location so they drove the turtle to each position and asked it to PRINT POS. That was a real conversation with the computer.

The children have also become fascinated by the way computers can converse with each other. Using two machines each equipped with control interfaces, they initially connected a light bulb to an output of one and a light sensor to an input of the other in an attempt to send messages with a flashing light. Finding frustrations occasioned by fluctuating ambient light levels, they later abandoned the idea in favour of direct "hard wiring" from one box to the other, also permitting the transmitter and receiver to be out of line of sight of each other. They have taught them both the morse code so typing the relevant letter sends a series of ons and offs of varying durations corresponding to dots and dashes and recognition procedures timing the duration of incoming signals to type the relevant dots and dashes on the receiver's screen. (They are presently working on a "reading" program so whole passages of text may be entered at a time instead of individual letters and a "translation" program so the receiver can interpret the dots and dashes as letters.)

These children have reached this conclusion: "It is only possible to hold meaningful conversations with computers (or people) if they know what you are talking about!"

Conclusions

With the help of the Logo computer, educational practice has taken great strides. It used to be said it takes 25 years to make fundamental changes in education. The computer has been playing its part for half of that and accomplished much. What will the future hold in store? What will Fourth Generation CAL look like?

"To hold a meaningful conversation with a computer, it must know what you are talking about," or could it make a meaningful guess? Will computers coupled to vast data banks enable more productive learning?

Whatever happens, Logo has so far been able to provide a sound platform to support education and the means to convert theory into practice and there is no reason, given the academic and research support it receives, to suspect it will not continue to do so.

REFERERENCES

[1] Sir John Maud et al, School's Council, (1965) "Mathematics in Primary Schools"
[3] Lady Plowden et al (1967) "Children and their primary schools" Report
[2] W.H.Cockroft et al (1982), "Mathematics Counts" Report
[4] John Vaizey (1962), "Education for Tomorrow"
[5] Schools Council (1972), "With Objectives in Mind"
[6] HMI Science Committee (1983), "Science in Primary Schools" Discussion paper
[7] Richard Noss, Chiltern Logo Project, (1983), "Children Learning Logo Programming"

Pré-lecture par Logo en 3ème Maternelle

Monique Rousseau-Gerard

Institut Supérieur Pédagogique Libre
61, Hors-Château
4000 Liege Belgique

Abstract

This Logowriter experiment in pre-school, with children five years old, began in November 1992 with an Instant Logo devised by the author and regularly completed, even with whole words, instead of just one letter, according to the needs and wishes of the children. The objective of this year's experiment is for the children, after creating a bank of drawings, each with its own name, to choose from that bank to design an elaborate picture with many different elements telling a complete story. The children will locate their turtle at the exact place where the drawing should come, give it the correct heading and adjust the sizes of the drawings. This will be an exercise, first in global reading, by choosing in the list the right drawing at the correct scale, and afterwards in analytical reading when typing the name on the keyboard to make the drawing appear on the screen.

Keywords: instant Logo, preschoolers drawing, pre-reading abilities

Monique Rousseau-Gerard *a une formation de philologue classique complétée par une licence sur l'Informatique appliquée aux Sciences Humaines. Chargée depuis 1987-1988 d'un cours d'informatique destiné aux normaliens de l'enseignement maternel et primaire à l'Institut Supérieur Pédagogique Libre de Liège, elle a élaboré des programmes Logo instantané pour permettre à des enfants de maternelle d'utiliser l'ordinateur avec toutes ses possibilités. Depuis 1988-1989 des expériences sont en cours menées par l'auteur et certains de ses étudiants dans le cadre de leurs travaux de fin d'études. L'orientation privilégiée cette année dans ces expériences est l'apprentissage précoce de la lecture.*

Pré-lecture par Logo en 3ème Maternelle

Historique de l'expérience

L'expérience Logowriter en maternelle a débuté en 90-91 et s'est d'abord axée sur la production de dessins par les enfants, grâce à un programme qui leur permet d'utiliser les principales primitives graphiques de Logo avec une seule touche, validée chaque fois par ENTER (j'ai appelé ce programme Logo Instantané [7]). Petit à petit, à la demande des enfants, ce programme, équipé d'un générateur automatique de procédures, s'est développé et a incorporé quelques commandes plus longues telles que CARRE, TRIANGLE, CERCLE... Les enfants ont donc commencé à entrer dans le monde de l'écrit et de la pré-lecture pour la création de leurs dessins. Mais l'incursion dans ce monde était encore très ponctuelle et réduite, de même chaque fois que les enfants, pour enregistrer leur dessin, reproduisaient un nom, dont le modèle était donné par l'adulte.

Nous voulons maintenant entamer un nouveau chapitre qui va nous permettre d'utiliser au maximum les possibilités de Logowriter dans une méthodologie de la lecture fondée sur les théories récentes présentées notamment par E. CHARMEUX [1]. Les enfants se sont constitué une banque de dessins dont chacun porte un nom significatif. Ces noms ont été au fur et à mesure intégrés à la banque de mots de la classe, en même temps que le dessin qu'ils représentent, et les enfants les manipulent à différentes occasions. Nous allons leur proposer de créer toute une histoire sur une page Logowriter, en utilisant plusieurs de ces mots.

Chaque enfant entamera ses recherches comme il en a envie dans un imagier sur ordinateur, réalisé à partir de ses propres productions. Il choisira pour sa lecture les indices les plus pertinents pour lui, il transcrira le mot au clavier, de façon analytique et obtiendra un feed-back immédiat de l'ordinateur en voyant apparaitre le dessin à l'endroit qu'il a choisi.

Phase exploratoire

Conditions pratiques

L'expérience a commencé au mois de novembre 1992 dans la classe de 3ème maternelle (enfants de 5 ans) de Patricia Vanderhaegen. Elle compte 22 élèves : 11 garçons et 11 filles qui ont tous eu l'occasion, à 4 reprises, au début de la 3ème maternelle de jouer avec la tortue de sol Jeulin. Un certain nombre d'enfants l'avaient déjà utilisée en 2ème maternelle, avec Patricia (qui a changé d'attributions cette année).

Les enfants travaillent par deux à l'ordinateur : 10 à l'ISPL avec Monique Rousseau et jusqu'à 12, sur 5 appareils à St Thomas, avec Patricia. Les séances pratiques ont lieu un lundi sur deux, de 2h30 à 3h30, le plus souvent avec un encadrement de normaliens en formation. Chaque groupe d'enfants dispose d'un livret d'aide où sont illustrées chacune des touches à leur disposition. Ils travaillent uniquement en lettres capitales, de façon à avoir le même «dessin» à l'écran et au clavier.

Remarque importante

Cette expérience se déroule dans une vraie classe et est donc dominée de bout en bout par des contingences matérielles qui empêchent de réaliser une «expérience» véritable : les présences sont plus ou moins régulières selon les enfants; leurs niveaux individuels sont extrêmement différents; d'autres enfants s'intègrent parfois au groupe, selon les absences accidentelles de leur institutrice; il faut compter avec les anniversaires, Saint-Nicolas et autres évènements importants dans la vie de la classe. Mais il nous paraît à toutes les deux très

important de proposer l'activité à tous les enfants; si, en cours d'année, certains enfants préfèrent se tourner vers une autre avtivité, nous aviserons en temps utiles.

Objectifs

Dans un premier temps, les enfants découvriront le Logo Instantané et créeront des dessins.; ils leur donneront un nom significatif et les sortiront à l'imprimante. Ils se constitueront une banque de dessins accompagnés de leur nom, incorporés au fur et à mesure à leurs livrets d'aide et réunis aussi dans leur classe dans la boîte à mots qui se complète tout au long de l'année.

Ce programme Logo Instantané nous a fourni un outil d'observation fine du travail des enfants, grâce au générateur automatique de procédures qui enregistre chacune des commandes utilisées par l'enfant au cours de son dessin; on peut suivre parfaitement et dans le détail ses démarches intellectuelles : l'anticipation, les essais dans le domaine de la latéralisation, les corrections, les hésitations, les vérifications d'hypothèses (il suffit de refaire le dessin de l'enfant par l'appel de la procédure en TORTUELENTE).

Dans un deuxième temps, les enfants inventeront une histoire à illustrer par ces dessins.
Patricia et Monique se chargeront de les reproduire à différentes échelles et dans la page FORMES. Les enfants créeront une histoire en faisant apparaître leurs dessins à l'endroit de leur choix, dans une nouvelle page, en tapant au clavier les noms qu'ils ont eux-mêmes donnés à leurs dessins; ils en choisiront la taille et ils pourront aussi produire un texte par dictée à l'institutrice. En fin de parcours, chaque groupe d'enfants devrait avoir constitué son propre livre et donc savoir identifier une série de mots, à la fois globalement et de manière analytique.

Premières sessions de travail

Lundi 16 novembre

Dans leur première découverte libre du clavier, certains enfants tombent sur la page d'aide (qui apparaît, à l'usage des «grands», quand on enfonce la touche H) et voient du texte à l'écran. Les stagiaires expulsent, à tort, cette page sous prétexte que les enfants ne savent pas lire; mais ils aiment faire surgir un texte à l'écran et ils en tirent des informations utiles, comme par exemple de repérer les lettres «intéressantes». De plus, pour sortir de ces pages d'aide, ils doivent trouver au clavier la lette W qui leur permet de retrouver leur dessin et de ce fait, pratiquent un peu plus le clavier.
Il faut faire verbaliser les enfants, mais pas au seul profit des stagiaires qui les entourent, aussi à l'usage de leurs condisciples des autres groupes : «Quelles lettres avez-vous découvertes ?»... «Comment es-tu arrivé à cette couleur ?»... Que veux-tu dessiner et comment vas-tu faire faire cela par ta tortue?»...

Concertation 23 novembre

A partir du 2ème trimestre, Patricia aura dans sa classe un ordinateur PC et organisera des ateliers informatiques avec Mimi la Fourmi, les programmes du CRAPO, au rythme des saisons.

Dès à présent, il y a en classe une boîte à mots qui complète 3 dictionnaires pour enfants, on y introduira systématiquement les noms des dessins.

Lundi 30 novembre

Les enfants font une mise en commun des lettres découvertes la fois précédente (sur un clavier autonome identique à celui qu'ils utilisent). Ils découvrent à l'écran la page AIDEMAT d'où sont tirées les illustrations de leur livret d'aide. Ils sont invités à utiliser ce livret régulièrement. Il y a trois nouveaux dans le groupe : les enfants sont invités à s'aider et à se débrouiller entre eux (exercice de verbalisation).

Lundi 14 décembre

Les 3èmes maternelles B encadreront le travail des enfants et s'y préparent de 1h à 2h en expérimentant elles-mêmes le programme qui s'est enrichi, à la demande des enfants et comporte : des arcs de cercle plus petits, une procédure PEINS, trois couleurs (bleu, rouge et jaune).

Les stagiaires de l'Ecole Normale recoivent comme consignes d'encadrement de faire découvrir par les enfants, de les renvoyer au livret d'aide, de ne jamais utiliser le clavier à leur place, mais de leur faire trouver les lettres eux-mêmes, de les encourager à enregistrer leurs dessins terminés et à les imprimer, de les encourager à partager leurs connaissances et enfin de leur faire ressentir l'erreur comme positive et constructive. On leur demande d'observer particulièrement les stratégies utilisées, les échanges verbaux, les interactions dans l'équipe et entre les groupes.

Remarque pratique.

Les stagiaires veilleront soigneusement à ce que le clavier des enfants reste bloqué en capitales, de façon à ce que le dessin des lettres au clavier corresponde à celui des touches. D'ici peu, nous rendrons les enfants conscients de la différence entre majuscules et minuscules et nous leur apprendrons à gérer eux-mêmes la touche SHIFT LOCK en vérifiant l'état de la diode.

Lundi 4 janvier

Le jour de la rentrée, nous faisons face à un problème non prévu : les ordinateurs ont été modifiés; il faut de nouveau des petites disquettes et il faut des couleurs différentes (carte VGA). Les enfants se regroupent autrement et ne peuvent plus retrouver leurs programmes de la fois passée...

On commence par vérifier, chacun à son ordinateur, les lettres suggérées par certains enfants pour constater si le mouvement annoncé de la tortue est celui qu'on a prévu. Un enfant propose une lettre et dit ce qu'elle fait faire à la tortue, tout le monde vérifie (on écrit éventuellement la lettre au tableau pour ceux qui ne la connaissent pas).

Lundi 18 janvier

Nous décidons d'essayer d'avoir des équipes stables de deux enfants et de leur faire préparer sur papier un dessin à faire faire à la tortue. Dans la pratique, les groupes ne se maintiennent pas parce que Martin veut être avec Zoé, pas avec Charlotte, et que Ismael ne veut pas venir et est remplacé par une fillette spécialement irrégulière, Marine. Les dessins préparés par les enfants sont (beaucoup) trop difficiles à réaliser. Par exemple, Martin et Zoé veulent faire un building; ils le commencent très bien, mais, arrivés aux fenêtres ils se lassent et effacent tout.

Le problème est souvent d' inciter les enfants à enregistrer un dessin qu'ils considèrent comme non fini... pour qu'ils puissent le reprendre une autre fois. La procédure d'enregistrement avec F est un peu contraignante avec l'obligation de taper au clavier un mot d'au moins trois lettres. L'emploi de V est bien plus facile : la page se nettoie et on peut recommencer tout de suite autre chose...

Mercredi 20 janvier

Je propose à Patricia d'anticiper légèrement sur le programme imagier et la pré-lecture en créant quelques éléments de petites tailles que les enfants pourraient utiliser tout faits pour compléter leurs dessins individuels : des petits carrés (les fenêtres du building de Martin et Charlotte), des triangles (les aspérités du dinosaure de Marine), et des petits cercles. Nous nous arrangerons pour leur fournir les éléments utiles pour eux et trop difficiles à faire tout en limitant les mots à utiliser.

Nous ne voulons surtout pas escamoter une difficulté qu'ils auraient plaisir à surmonter eux-mêmes. Il faut

que l'emploi de l'élément tout fait pose à l'enfant des problèmes d'utilisation plutôt que de lui donner des solutions préfabriquées. Par exemple, quelle succession d'opérations devra-t-il utiliser pour que l'élément s'intègre de la meilleure façon dans son projet? D'autre part, le stade de pré-lecture systématique sera facilité par l'usage de mots entiers qui aura précédé; les enfants disposeront évidemment des modèles dans leur petit manuel tiré de la page AIDEMAT auquel nous les renvoyons systématiquement.

Jeudi 4 février et vendredi 5 février

Les deux groupes de la classe viennent à tour de rôle à l'I.S.P.L. parce que les ordinateurs de l'école d'application ne sont pas opérationnels. Les enfants sont encadrés par des normaliens de 2ème année qui ont eux-mêmes expérimentés le programme. Ils arrivent avec un dessin qu'ils s'efforcent de réaliser à l'écran. Martin a dessiné une fusée, Maïté une maison et un escargot, Claire un château. Ismael a retrouvé le dessin d'une stagiaire et est aux anges... Leur enthousiasme stimule les stagiaires au point que, après le départ des enfants, ils s'efforcent de réaliser leurs propres projets avec le programme des enfants. Ceci met en évidence quelques bugs et anomalies auxquels il sera porté remède sans tarder.

Lundi 8 février

Nous décidons d'insister pour avoir des groupes stables et pour y arriver nous décidons de tirer chaque fois à l'imprimante les dessins des enfants et de les ajouter à leur petit livre. De cette façon, ils s'imprègneront du nom de leur dessin, ce nom sera d'ailleurs ajouté au dictionnaire en classe, si bien que chaque groupe d'enfants aura tous les noms de dessins réalisés par les autres; ils auront plus d'idées pour les autres éléments à introduire dans leur histoire; ils pourront travailler en tenant compte de l'échelle, pour pouvoir regrouper les différents éléments de l'histoire qu'ils créeront.

Pour cette fois-ci, Patricia leur demandera de préparer le dessin d'un personnage ou d'un animal, puisque les éléments déjà envisagés appartiennent tous au paysage.

Martin veut faire un robot, qui est bien ébauché. Maïté dessine (très bien) un cheval. Charlotte veut faire une princesse du Moyen-Age mais abandonne devant les rubans du chapeau pointu... Ismael n'arrive pas à ce concentrer sur son bonhomme.

Mars

Les projets des enfants sont de plus en plus élaborés et les réalisations se font précises et minutieuses. Le moment de passer à la seconde phase du travail approche : ils maitrisent parfaitement le clavier (il ne leur arrive plus jamais de maintenir le doigt trop longtemps sur une touche, ni de frapper une touche par erreur); ils reconnaissent une série de lettres; ils ont envie de résoudre des problèmes plus compliqués.

Pré-lecture

Dès le début de leur travail à l'ordinateur, les enfants ont rencontré la langue écrite, dans sa fonction d'outil de communication qu'ils peuvent maitriser, comme le souhaite R. COHEN [3]; même la production d'une lettre est un geste moteur significatif, dans un espace structuré de gauche à droite de façon automatique par l'ordinateur. L'enfant acquiert la signification des mots par une recherche active qui aboutit à la création d'une oeuvre [2].

Il est évident que ne sont pas absents de nos objectifs les pré-requis traditionnels de la lecture, présentés notamment par M. RASSENFOSSE [6], que sont l'organisation spatiale, l'organisation temporelle, la discrimination visuelle : l'enfant doit positionner sa tortue à l'endroit où il veut voir apparaitre le dessin.; il doit l'orienter dans l'espace, prendre conscience des distances et des échelles. Mais nous voulons observer particulièrement quelles démarches les enfants mettent en oeuvre pour la lecture de leurs mots et la création de leur histoire. Ils doivent générer des images, explorer des relations et gérer différentes tailles. Comment font-ils et comment analysent-ils leur action ?

Cette deuxième phase de notre travail sera présentée lors de la communication au Congrès Eurologo.

BIBLIOGRAPHIE

[1] CHARMEUX E., *Apprendre à lire*, Milan Education, 1987.

[2] COHEN Rachel, *L'apprentissage précoce de la lecture*, PUF, Paris, 1977.

[3] COHEN Rachel, *Les jeunes enfants, la découverte de l'écrit et l'ordinateur*, PUF, Paris, 1987.

[4] COHEN I. et MAUFFRAY A., *Vers une nouvelle pédagogie de la lecture*, Armand Colin Bourrelier, 1983.

[5] JOLIBERT, *Former des enfants lecteurs*, Hachette, Paris, 1984.

[6] RASSENFOSSE M., *Les pré-requis de la lecture.*

[7] ROUSSEAU M. et al, *How chidren from ages 3 to 14 progress in the classroom with Logo*, dans Eurologo 91, pp 122-125.

[8]TOUYAROT Charles, *Lecture et conquête de la langue*, Nathan, 1971.

Where to Place Logo in Teacher Training

Márta Turcsányi-Szabó

Eötvös Loránd University, Faculty of Natural Sciences
Department of General Computer Science
Hungrary
e-mail: Turcsanyine@ludens.elte.hu

Abstract

This paper hopes to illustrate a possible way teachers of informatics could view a programming language like LOGO to be used in the education of curriculum subjects. The topic concentrates on language education.

Keywords: Logo, teacher training informatics education, language education, text modelling, microworlds

Where to Place Logo in Teacher Training

Introduction

The Eötvös Loránd University has been training teachers in Informatics for many years. Personal computers have been introduced since 1981. Students graduating from our university would mainly teach at secondary school level, and due to the reorganisation of many schools, at elementary level too. Until recently schools have had an 8+4 years division for elementary and secondary levels, nowadays you can find 4+8, 6+6, or even a combined 12 setting too. This means that our teachers should be able to place themselves in this overall role of school education.

The courses offered to students are constantly developed and renewed to form an up to date professional image for teachers to fulfil their role in different educational institutions. The program is designed to be completed in 5 years and graduates are given a M.S. degree in Computer Science.

The University also provides a post graduate program, and training for in-service elementary or secondary school teachers of all levels possessing B.S. degrees, to major in informatics. The program completes in two years and graduates are also given M.S. degree in Computer Science.

Both programs provide a large scale of knowledge in informatics with a close relation to education.

- They are trained in several operating systems, different networks and environments, get acquainted with different types of computers and software to be able to design and install a computer network.

- They are introduced to several aspects of computer science, systems of informatics, numerical analysis, artificial intelligence. The idea of constant development is injected into them to be sensitive to changes and needs.

- They are introduced to a variety of application software like word processors, spreadsheet, database management, simulation programs, strategic games, and other educational software. Thus they can compare several similar tools and try to figure out their place in education.

- They are introduced to programming methodology, data structures and algorithms through a smoothed training program [1] [2]. Thus they are encouraged to design educational software.

- They acquire a number of programming languages like ELAN, PASCAL, FORTH, PROLOG, LOGO, and a few others on non-compulsory basis both as a programming tool and an educational tool as well. They learn the main features of each language and taste the possibility of applying them in education.

- Students can choose several other subjects besides informatics as their majors, eg. mathematics, physics,...etc. Thus they are able to look out onto other fields and try to apply what they learned in improving the motivation for learning other curriculum subjects.

- They do a lot of teaching in several schools during their university years whilst they can apply their knowledge, acquire some practice and consult on topics.

The National Basic Curriculum is presently under construction and has been highly debated among the leaders of educational policy, researchers, and educators. It should be concluded and accepted by the parliament this year. The concepts of informatics education is also under way and views are certainly converging. The basic aim of informatics education in schools should

focus on introducing the following aspects to children:

- Getting acquainted with the instruments of informatics, to realise how they work and identify how they could profit from their use.

- Getting acquainted with application software and realise their use in various fields.

- To understand algorithmic thinking and to be able to produce algorithms in solving various problems.

- To be able to build a model on different themes.

- To understand the place of informatics in society.

These aspects should be introduced on different levels of education for various age groups especially aiding children with disabilities.

LOGO is a useful tool in achieving several of the above goals. It is not only a programming language, but a suitable tool in developing educational microworlds that are easy to enhance further. We have developed several educational programs and microworlds on different subjects. Apart from introducing LOGO to teacher training we have conducted several experiments with children of different aged groups (from kindergarten to secondary school) and of different abilities (from children with disabilities to those that could be called genius). Such meetings continuously enriched, guided and gave birth to new ideas. The focus of attention has always been the child, his abilities and needs, and not the programming language itself. Thus the microworlds concentrated on helping to grasp the idea and meaning of something important with the least necessary previous knowledge of computers or programming. This paper discusses an approach to aid language education by modelling the structure and meaning of text.

The model of text

Model 1. A text is a series of sentences.

```
                          n
<text> ::= { <sentence> }
                          1
where <sentence> i ::= { an element of the set of
                          predefined sentences }
```

This definition can be used as the first stage in modelling text. It presumes the existence of a set of sentences of which a random number can be chosen also at random. Depending upon the subject of the text enhancements can be made.

E.g. Postcard: [3]

In this case the set of sentences can be individual thoughts on the subject to be transmitted.

Enhancement:
- `<Sentence> 1` could be something like
  Dear `<name>`.
where `<name>` is the set of persons to whom we should send a postcard.
- `<Sentence> n` could be something like

<ending> *Peter.*
where <ending>::= {Your dear | Bye | See you | Love | ...}
or else <ending> can correspond to the specific <name> used.

Creating this program develops an insight not only on lists and text generation but also on the meaning of random choice.

Model 2. A sentence is a series of grammatically structured words.

```
<sentence> i ::= <time clause> <clause of place>
                 <subject clause> <predicate clause>.
<predicate clause>::=<subjective conjugated clause>
                 | <objective conjugated clause>
<subjective conjugated clause> ::= <subjective
                 conjugated verb>
<objective conjugated clause> ::= <objective
                 conjugated verb> <object clause>
<object clause> ::= <adjective> <object>
<subject clause> ::= <adjective> <subject>
... each of which is an element of a set of
predefined words or expressions
```

E.g. Generating a story: [4]

In this case the sentences should tell unpredictable actions of the actor or actors corresponding to the theme of the story.

Enhancement:

- Like that of model 1
<sentence> 1 could be something like
Once upon a time ...
<sentence> n could be something like
They lived happily ever after.

- Choose the theme of the story then choose the actor or actors of the story and build the rest accordingly.

. In case of one actor, the structure of the sentences might always contain <subjective conjugated clause>. While in case of two actors an <objective conjugated clause> would do better or even a random choice from the two.

. The set of names or synonyms included in <object> and <subject> should be different or correspond to the line of happenings. If one of the actors is chosen as a subject then the object should definitely be chosen so that it corresponds to the other actor.

. Adjectives should be chosen in accordance with the actors.

- Articles should be generated by algorithm.

This program could be further enhanced with deeper knowledge of grammar.

Model 3. Each word has a meaning.

```
a word is the syntactics of a command which bears
     a function as its semantics

There are two types of functions concerning input:

     type1 : takes no input
     type2 : take one or more input

There are two types of functions concerning output:

     1type : does not output
     2type : outputs a value
```

Types can of course be combinations too: `1type2`, or `2type2`, ...etc.

Other internal properties depend upon the actual semantics, whether it has an effect, or whether it modifies environment variables.

E.g. 1. The nationalisation of LOGO

LOGO is a natural environment for English speaking children. It should be a natural environment for children of various language environments. The nationalisation of a command set is not as easy as translation of the words themselves. The commands have to possess communicative power as natural as possible [5].

Constructing the national version of LOGO is very helpful in understanding and interpreting LOGO the proper way. It also gives a deep insight to the properties of a functional programming language. Teachers should encourage the modification of existing LOGO version to a self-suiting variation. [6]

E.g. 2. Kidlogo [7]

Kidlogo is a LOGO-like environment for preschoolers. Algorithmic thinking and implementation is an uneasy task in itself, syntactic restrictions can hinder this approach especially at an early age. Kidlogo was developed for children with disabilities and later completed for preschoolers. The system consists of spirally building LOGO environments like drawing, music, animation, and text processing, the products of which can be combined to form a story. The very first stage is based on the easiest computer controlling device, the joystick. In the second stage the computer can be controlled through the meaningful use of icons that are glued on the side of each key on the keyboard. While the third stage is accomplished through the use of predefined commands. Thus, children can control the computer through stages that are always building on the previous stage and enables the child to control the computer in a more sophisticated manner, if he or she is ready to do so. This spiral grows into the full use of LOGO. From the second stage, the control of computer through icons, the child is able to develop procedures and build new words from basic, preprogrammed, or previously built procedures. Procedures can be built by evoking actions with a single key or typing procedure names, and mistakes can be corrected by deleting the last step (or the whole). The environments are introduced in the following order: joystick drawer, drawer, music, animation, creation of sprites, text processing. The order of introduction is of great importance, since the next environment includes the tools of the previous one and enhances it further.

Drawing in itself is extremely motivating at an early age, but animation has even greater power. Even drawing is created by movement.

In the animation environment one invokes a subject actor and each command specified is acted upon the subject.

E.g. - to change colours
- to change size
- to perform an action ...etc.

The environments are building on top of each other so actions are layered in the following way too:

-LOGO words or corresponding icons: ***forward, back, right, left,...***

-preprogrammed words with the help of the previous commands
go - slowly forward
run - fast forward
jump - up then down
fly - move in a circular motion
.....

-creating words implying movements with the help of the previous functions

Using this environment the children are introduced not only to informatics, but also realize the importance and the meaning of words, and text [8].

E.g. 3. Airport [9,10]

This microworld represents an airport and simulates the basic movements in it. The animation itself is continuous and at any moment an event can be evoked by typing defined commands. The wording is of utter importance, so it is necessary to describe the meaning of the hungarian words in use.

kiszállás - deplane: the passenger gets off the plane, walks to the bus, gets on the bus, the bus takes him to the airport building where he gets off the bus and walks into the building;
beszállás - emplane: the passenger walks to the bus, gets on the bus, the bus takes him to the aeroplane where he gets off the bus and gets on the plane;
leszállás - landing: the aeroplane approaches the runway and lands smoothly;
felszállás - takeoff: the aeroplane takes off at the runway and flies around the airport sky till further instructions;

száll - in hungarian means fly, and ***-ás*** is an adjectival postfix.
ki - out
be - in
le - down
fel - up are prefixes

As you can see from the structure of the words the prefix determines the event itself. Thus the whole meaning lies in the proper use and understanding those few letters.

The actor object of the microworld is the passenger:

He can be activated otherwise to go: into the building, into the bus, into the aeroplane, while exiting from the place he has been inside.

Objects of the microworld:

bus, passenger, aeroplane, sun, ...etc.

The objects could change colours when given a colour name after theirs:

brown, orange, yellow, blue, ...etc..

Other events included: the rise of the sun, or sunset which are by the way expressed in hungarian with the same *száll* adjective. Thus this whole microworld evolves around the same adjective in different forms, giving different meanings to different preposition use.

This microworld is not so much object oriented, but event oriented.

Model 4. Understanding a sentence means understanding the words in their context.

```
Syntactics:
                              n
<sentence> i ::= { <word> }
                              1
where <word> 1 ::= a word starting with a capital
                   letter
      <word> n ::= a dot

Semantics:

word 1 :   Functions with capital letters invoke
           execution of the specified action.
word i :   Each word i is a function which needs an
           input value as its parameter sets the
           environment variable in correspondence
           to its semantics and outputs a value for
           the previous function that invoked it.
word n :   The dot does not have a parameter, it
           starts the flow of information backward.
```

In natural language the start of a sentence is recognised by the capital letter of the first word. Each word has a meaning which is understood in context only i.e. by understanding the word that proceeds it. The dot finalises the sentence which should be interpreted in its context.

E.g. Family movie [11]

Mother - /First word./ /What's with her?/
Mother says /What does she say?/
Mother says doggie /What to do with doggie?/
Mother says doggie come /Where?/
Mother says doggie come here /Ok, and what else?/
Mother says doggie come here . /End of sentence. Inform back./

. reports to ***here*** that the sentence ends
here sets the value of WHERE to the relative value of the talker
and reports to ***come*** + end
come sets the value of DOTHIS to "come"
and reports to ***doggie*** + WHERE + end

doggie sets the value of DOER to "doggie"
and reports to ***says*** + DOTHIS + WHERE + end
says reports to ***mother*** to say + DOER + DOTHIS + WHERE + end
Mother sets the value of TALKER to "mother"
(thus the absolute value of WHERE is set),
and since it start with a capital letter thus it evokes execution.

Since each word /function/ sets the value of the environment, further sentences change only the values that are indicated, thus the rest of the environment variables are presumed to remain.

E.g. ***Doggie go home .***

Presumes that the talker is still mother, ***home*** sets the relative value of WHERE, and ***doggie*** sets the absolute value to the position of his home /which each actor possesses/.

Boy come here .
Changes the DOER to boy and presumes that WHERE is where mother is.

Go home .
Presumes that the DOER is still the boy and WHERE is where mother is.

This "Family movie" microworld is object oriented and produces a context controlled animation environment which is easy to expand.

The simple sentences structure:

```
1.   <do> <where>

where      <do> ::= {come, go,...}
           <where> ::= {here, home,...}

2.   [<who>] <do> <where>

where      <who> ::= {boy, doggie, father, mother,...}

3.   [<who> says] [<who>] <do> <where>

4.   [<who> says] [<who>] <do> <where> [the <what>]

where      <do> ::= {come, go, bring, take,...}
           <do> ::= {ball, bone, cake,...}
```

n. or even further expanded.

Conclusion

The LOGO programming language is not only useful in learning programming, but the knowledge on the subject that is built weighs much heavier in the pan of creating a proper model. Trial and error strengthens the rules that govern a microworld, thus experimenting enlightens even the darkest points. Playing with a well built microworld can be as fascinating as playing with a game. Teachers should be encouraged to develop such microworlds to aid curriculum subjects.

REFERENCES

[1] Péter Szlávi - László Zsakó, Módszeres programozás: rekurzió (Systematic programming: recursion), ELTE University publication, 1988.

[2] László Zsakó, Módszeres programozás: hatékonyság. (Systematic programming: efficiency), ELTE University publication, 1991.

[3] Harold Abelson, A Beginner's guide to Logo BYTE, August, 1982.

[4] Márta Turcsányi-Szabó, "Story", (a LOGO program) 1984.

[5] Márta Turcsányi-Szabó, MLOGO, a LOGO honosítása. (The nationalisation of LOGO) Unpublished dissertation, 1985.

[6] Márta Turcsányi-Szabó, LOGO in a University Education: Experiments and Plans. Proc. International Conference LOGO 85, M.I.T., 1985.

[7] Márta Turcsányi-Szabó, "KISLOGO" (Kidlogo, a Logo-like system) 1985.

[8] Hadnagy - Havas - Körmendi - Turcsányi - Vasánszky, Tanulmány egy óvodai számítógépes kísérletról (Study of an experiment in kindergarten with computers.) Unpublished Study, 1986.

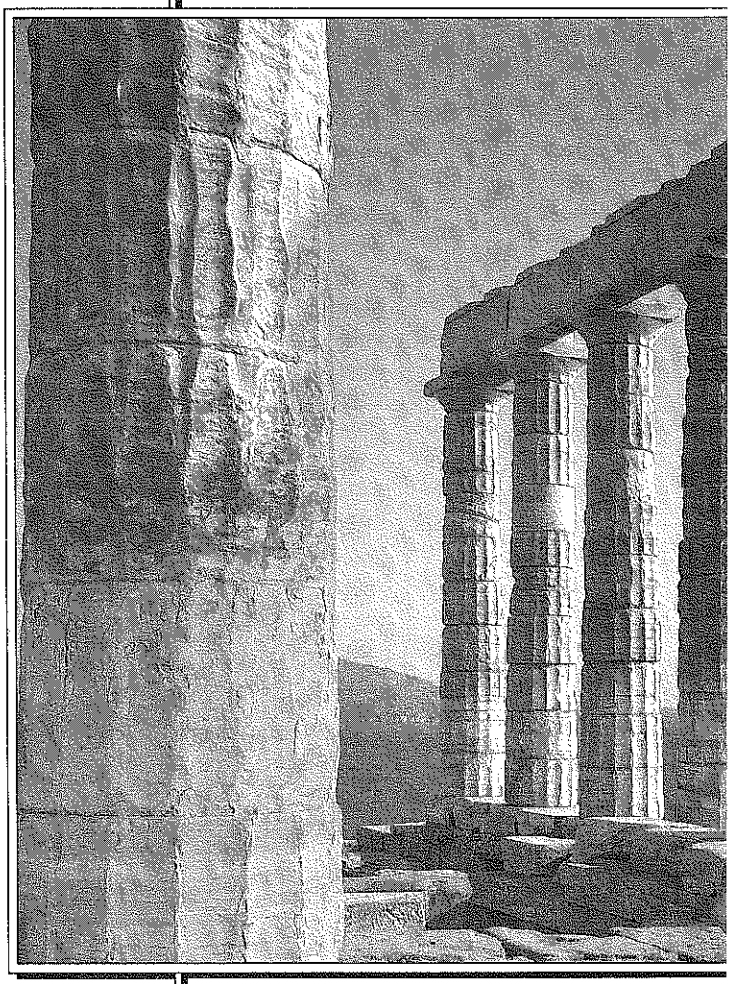
[9] Márta Turcsányi-Szabó, "REPÜLÓTÉR" (Airport, a LOGO program) 1984.

[10] Kóhegyi - Zsakó - Turcsányi - Szlávi, Teaching Teachers the Educational Use of Microcomputers. Proc. International Conference on "Microcomputers in Education", Balaton-Almádi, 1985.

[11] Márta Turcsányi-Szabó, "Family movie" (LOGO program) 1993.

Theme D

Software Design



D. Software Design

213 • *Batagelj V.*
Prfkos Authoring System Based on Logo
223 • *Bellemain F., Dagdilelis V.*
La Manipulation Directe Comme Invariant des Micromondes Educatifs ...
233 • *Grobelnic M.*
Design of Logo Command Names
239 • *Kato H., Ide A.*
AlgoArena: a System for Learning Programming through Social Interactions
247 • *Tergan S., Oestermeier U.*
Cognitive Design Principles of a Generative Logo-like Learning Environment

PRFOKS Authoring System Based on Logo

Vladimir Batagelj

Department of Mathematics, University of Ljubljana
Jadranska 19, 61 111 Ljubljana
Slovenia
tel: (+38) 61 157 100 / 72, fax: (+38) 61 217 281,
e-mail: vladimir.batagelj@uni-lj.si

Abstract

In Slovenia a professor is called *prfoks* by students. In the paper basic ideas considered in the redesign of PRFOKS - an authoring system based on Logo are presented.

The most plausible way to obtain good educational software is to provide teachers with powerful but also user friendly tools for creating educational programs by themselves. A kind of such tools are authoring systems. In the paper we show that in Logo we can implement a very versatile authoring system which combines execution and editing.

Keywords: LogoS, educational authoring system, event driven programming.

***Vladimir Batagelj**, born 1948 in Idrija, is a professor of Discrete Mathematics and Computer Science at the Department of Mathematics at the University of Ljubljana. Since 1985 he is a chairman of Subcommission for educational software at the Board of Education of Republic of Slovenia. He was involved in several activities promoting the computing culture in Slovenia and wrote several papers and some books on the subject. Since 1989 he is developing, together with M. Grobelnik and D. Zupanic, a Logo dialect LogoS. His main research interests are in data analysis, graph theory and theoretical computer science.*

PRFOKS Authoring System Based on Logo

1. Introduction

In Slovenia a professor is called *prfoks* by students in their discussions about school. In the paper we present basic ideas considered in the design of PRFOKS - an educational authoring system based on Logo.
Usually, teachers are bad programmers, and programmers are bad teachers. For these reasons it seems that there are two ways to obtain good educational software:

- to organize the software development teams (like in film production) consisting of teachers, programmers and other specialists. Teachers select the theme in produce a scenario, programmers develop the corresponding program.

- to provide teachers with powerful but also user friendly tools for creating educational programs by themselves. A kind of such tools are authoring systems.

The first approach is very expensive and seems appropriate only for themes of broad interest. For specialized themes the second approach is much more plausible.
There exist several educational authoring systems (Tutor (Plato), Pilot, Gnosis) [4, 3]. We can use also general purpose authoring systems (ToolBook, Guide, Tempra) or presentation capabilities of other software packages (Word, WordPerfect, CorelDraw, Excel, Quattro Pro, Paradox, Visual Basic, ...) [10].

I started to think about an educational authoring system in the fall 1989. The first draft of PRFOKS was finished in january 1990 [1]. Since I was unable to organize it's implementation I put it aside for a while.

2. LogoS

In 1989 we started another project - design and development of our own Logo - LogoS [2, 5].

The programming language Logo was developed to introduce young children to computing and to enable educational use of computers. With LogoS it has been developed to the stage where it can provide a uniform computing culture throughout primary, secondary and university education [6]. Because of its simplicity Logo can become also a language for nonspecialist users.

In further development of LogoS we plan to implement several extensions (special libraries, new devices) and transfer it to Windows environment and to other types of computers. Since in Europe we have many languages we shall pay special attention to methods for writing multilingual (language adaptable) educational software. There already exist Slovene and English version of LogoS. We shall prepare national versions of LogoS for selected European languages (Italian, German, French, Croatian, ...).

We also plan: development of educational programs in Logo for selected subjects: geometry, sound and music, Lego, measurements, analysis of natural languages [12], ...; design and realization of universal educational interface (measurements, Lego, Fischer, sound, ...); research on applications of artificial intelligence in educational software.

To preserve its simplicity there is a need to define a *basic Logo* with all specialized features moved to libraries. Instead of extending Logo with hundreds of new features we can provide commands for cooperation capabilities (EXE, OLE (Object Linking and Embedding), DDE (Dynamic Data Exchange), MCI (Media Control Interface)) which enable Logo user to engage other programs to achieve his task [8, 11, 9].

In last decade a user friendly, menu-driven interface prevailed in programs which in its iconic form also substantially reduced the (native) language dependency. Some programs (dBase) combines menu interface with a programming language and provide on the menu level program generators for specific tasks. These solutions should also be considered when developing educational programs.

3. PRFOKS

In the first version of PRFOKS we planned to develop it as a program written in some standard language (pascal or C) in two steps:

- design and implementation of *language* - the set of commands typical for creation of lectures;

- design and implementation of user friendly *authoring shell* which translates teacher's descriptions of lectures into PRFOKS programs.

This system could be developed stepwise by adding different subsystems

- text,
- graphics,
- data collections,
- sound,
- measurements and control,
- animation,
- other devices, multimedia.

Later we changed our mind. A powerful authoring program should incorporate a kind of interpreter. Therefore it would be interesting to create an authoring system based on Logo extended with a library of commands supporting time critical operations. This can substantially reduce the development effort. A prototype system can be developed entirely in Logo. For these reasons we decided to redesign PRFOKS.

To demonstrate the feasibility of the proposed approach we present in the following section a simplified implementation of kernel commands of such a system.

4. Kernel

The realization of PRFOKS is based on event-driven approach. The current possibilities are determined by the corresponding group of actions. An *action group* is described by a list

[initial-actions final-actions event-1 ... event-n]

and an *event* by a list

[label position code action]

Here are some examples of events:

```
Var :Ev.Esc = [ "Esc [-1 -1] [ 0 27 ] [ Make "Link "Exit Throw :d ] ] End
Var :Ev.Tab = [ "Tab [-1 -1] [ 0  9 ]
    [ Make "ExI Tabular Thing :d :ExList :ExL :ExI ] ] End
```

```
Var :Ev.Ent = [ "Ent [-1 -1] [ 0 13 ]
    [ Make "Q AddLast Item 3 Item (Item :ExI :ExList) Thing :d :Q ] ] End
Var :Ev.F1  = [ "F1  [-1 -1] [ 1 59 ]
    [ SetPos [ 10 1] Pr "'Esc - Exit, F1 - Help' ] ] End
Var :Ev.Idl = [ "Idl [-1 -1] [ 5  0 ] [  ] ] End
Var :Artist  = [
    [ "IdM [-1 -1] [ 5  0 ] [ Make "Marko.i 1 + Rem :Marko.i :Marko.l
    Run Item :Marko.i :Marko ] ]
    [ "Add [-1 -1] [ 1  82 ] [ Make :d AddKey :d Make "ex Expand Thing :d
    Make "ExList Item 1 :ex Make "EvList Item 2 :ex
    Make "AcList Item 3 :ex GrScr ArtStart ] ]
    [ "MUp [-1 -1] [ 1  72 ] [ SetTrDir 0 Forward 20 ]   ]
    [ "MLt [-1 -1] [ 1  75 ] [ SetTrDir 270 Forward 20 ] ]
    [ "MRt [-1 -1] [ 1  77 ] [ SetTrDir 90 Forward 20 ]  ]
    [ "MDn [-1 -1] [ 1  80 ] [ SetTrDir 180 Forward 20 ] ]
    [ "SUD [-1 -1] [ 0  13 ]
       [ If "Up = GetPenPos 1 { PenDown }{ PenUp } ] ]
    [ "cLr [-1 -1] [ 0  99 ] [ ArtStart ]                ]  ]
End
```

The *code* in an event consists of two components:

[type id]

where *type* can be:

0 - ASCII character key,
1 - special key,
2 - expander,
3 - errors,
4 - return to parent level,
5 - idle.

The action group description is decomposed by command `Expand` into three additional lists which enables quicker execution of required actions

- *EventList* - contains transformed (256*type+id) codes of events;

- *ActionList* - contains for each code/event its index in the action group description list;

- *ExpanderList* - contains indices of expanders in the action group description list.

```
To Expand :d
   ( Local "k "t "n "ext "evt "act)
   Make "ext [] Make "evt [] Make "act [] Make "k 3
   Repeat (Length :d) - 2 { Make "t Item 3 Item :k :d
      If 2 = First :t { Make "ext AddLast :k :ext }{ }
      Make "t 256*(First :t) + Last :t  Make "n NMember :t :evt
      If :n = 0 { Make "evt AddLast :t :evt  Make "act AddLast :k :act}
         { Make "act Replace :n :k :act }
      Make "k :k + 1
   }
```

```
   Output ( List :ext :evt :act )
End
```

The execution of the action group is controlled by the 'universal' *event handler* :

```
To Handler :d
   ( Local "ExList "EvList "AcList "c "i "co "ex)
   Make "ex Expand Thing :d Make "ExList Item 1 :ex
   Make "EvList Item 2 :ex  Make "AcList Item 3 :ex
   Run First Thing :d
   Make "ExI 1 DispMenu :ExList :ExI
   Loop :d {
      Make "t GetEvent Make "co 256*(First :t) + Last :t
      Make "c NMember :co :EvList
      If :c > 0 { Make "i Item :c :AcList Make "ft First :t
         Run Item 4 (Item :i Thing :d)
         If (Or (2 = :ft) (4 = :ft))
            {Clr Make "ExI 1 DispMenu :ExList :ExI}{}}{ }
   }
   Run Item 2 Thing :d ; Make :d :des
   Make "Q AddFirst [ 4 0 ] :Q
End
```

Events are registered in the *queue* `Q` controlled by `GetEvent` :

```
To GetEvent
   (Local "evt "ch)
   If KeyPressedP { Make "evt RdEKey Make "ch First :evt
      If :ch = 0 { Make "evt (List 1 Last :evt) }
         { Make "evt (List 0 :ch) }
      Make "Q AddLast :evt :Q }{ }
   If EmptyP :Q { Output [ 5 0 ] }  ; idle
      { Make "evt First :Q Make "Q ButFirst :Q Output :evt }
End
```

If the queue is empty the *idle event* is returned which can perform some 'idle activity' - for example, plays music (see also event `IdM` in `Artist`).

```
To N.do Note 261.63 :N.t End To N.re Note 293.66 :N.t End
To N.mi Note 329.63 :N.t End To N.fa Note 349.23 :N.t End
To N.so Note 392.00 :N.t End To N.la Note 440.00 :N.t End
To N.ti Note 493.88 :N.t End To N.do1 Note 523.25 :N.t End
To DVA Make "N.t 2*:N.t  End To POL Make "N.t :N.t/2  End

Var :Marko = [ [ N.mi N.so N.so N.so ] [ N.mi N.so N.so N.so ]
   [ N.mi N.mi N.re N.re ] [ DVA N.do N.do ] [ N.do N.re ]
   [ POL N.mi N.so DVA N.so ] [ POL N.mi N.mi N.re N.re ]
   [ DVA N.do N.do ] [ Delay 2*:N.t POL ]  ]
End
```

Expanders have visual representation on the screen. We can move from one to another by `Tab` key. The

Enter key requires the activation (expansion) of current expander.

```
To DispExt :xy :lab :m
   If (First :xy) > 0 {
      SetPos :xy If :m { Type "'[' }{ Type "' ' }
      Type :lab  If :m { Type "']' }{ Type "' ' }
   }{}
End
To DispMenu :Ex :ExI
   (Local "tt)
   Make "ExL Length :ex
   If :ExL > 0 {
      Repeat :ExL {  Make "tt Item (Item GetIter :ex) Thing :d
         DispExt Item 2 :tt Item 1 :tt "False }
      Make "tt Item (Item :ExI :ex) Thing :d
      DispExt Item 2 :tt Item 1 :tt "True
   }{}
End
To Tabular :s :x :l :i
   (Local "t "j)
   If :l > 0 {
      Make "t Item (Item :i :x) :s DispExt Item 2 :t Item 1 :t "False
      Make "j 1 + Rem :i :l
      Make "t Item (Item :j :x) :s DispExt Item 2 :t Item 1 :t "True
      Output :j } { Output 0 }
End
```

As an example we present a system which can display 'about information', list a directory and draw simple figures on the screen. The drawing expander contains its own event handler.

```
To Prfoks
  Make "Artist (CompList [
     [ GrScr ArtStart Make "Marko.l Length :Marko ]
     [ TxScr Clr Pr "Exit.Artist ]
     [ "Esc [-1 -1] [ 0 27 ] [ Make "Link "Prfoks Throw :d ] ] ]
      :Artist) ;
  Make "Prfoks (List  [ TxScr Clr Pr "Prfoks ]
     [ Pr "Exit.Prfoks ]
     [ "Link      [ 60 1 ] [ 2 0 ] [ Make "Link "Artist Throw :d ] ]
     [ "Menu      [ -1 1 ] [ 4 0 ] [ Clr   ] ]
     [ "Prfoks    [  1 1 ] [ 2 1 ] [ About ] ]
     [ "Directory [ 15 1 ] [ 2 2 ] [ ListDir ] ]
     [ "Artist    [ 30 1 ] [ 2 3 ] [ Handler "Artist ] ]
     [ "Exit      [ 45 1 ] [ 2 4 ] [ Make "Link "Exit Throw :d ] ]
     Replace 4 [ SetPos [ 1 1 ] Pr
        "'Esc - Exit, F1 - Help, Tab - Move, Enter - Select'] :Ev.F1
     :Ev.Ent :Ev.Idl :Ev.Esc :Ev.Tab )
  Make "Link "Prfoks Make "Q [ ]
  Loop "Screener {
     If :Link = "Exit { Throw "Screener }{}
     Make "CLink :Link Make "Link "Exit
```

```
      Handler :CLink
   }
End
```

For completeness we list also the supporting commands for actions. The about information is displayed by

```
 To About
   Clr SetPos [ 35 12 ] Type "'P R F O K S'
   SetPos [30 14] Type "'version 0.0: April 1993'
   SetPos [29 16] Type "'(c)1993 Vladimir Batagelj'
   SetPos [70 23] Type "Enter Wait
End
```

The drawing actions are mainly controlled in corresponding events. The only longer action is

```
 To ArtStart
   Clr Home Pr "Artist  Make "N.t 300 Make "Marko.i 0
   SetPos [ 1 2 ] (Type "'Commands: '
     CodeToCh 27 CodeToCh 24 CodeToCh 26 CodeToCh 25
     "'-Move,c-Clr,Enter-Pu/Pd,Insert-Add key command,Esc-Exit' )
End
```

The display of directory is supported by commands [7]:

```
 To DrawWindow
   TxScr Clr SetWn [10 5 72 20 ] DrawFrame "' Files ' 1 2 "AroundWn 0 1
   ClrWn SetPos (List (30 - (IDiv Length (GetCurrentDir GetDrive) 2)) 1)
   Type GetCurrentDir GetDrive
End
To GetFiles :Spec
   Make "Directories Sort GetMatchedFiles "'*' ["DirectoryFAttr ]
   Make "Attr [ "ReadonlyFAttr "HiddenFAttr "ArchiveFAttr ]
   Make "Files []  Make "Ext :Spec
   If WordP :Spec {Make "Ext List :Ext} {}
   Repeat (Length :Ext) { Make "Files
         (Complist :Files (Sort GetMatchedFiles (First :Ext) :Attr))
      Make "Ext ButFirst :Ext }
   OutPut (Complist :Directories :Files)
End
To ListDir
   (Local "i "patt "x "y)
   DrawWindow SetPos [ 2 3 ] Type "'Files list = ' Make "patt [ "'*.*' ]
   SetPos [15 3 ] SetRdWnLen 36 Show :patt SetWn [ 24 7 60 8 ]
   Loop "w { If KeyPressedP { Throw "w } {} }
   Make "p Rd If "EndOfFile = :p { Make "p :patt }{}
   Make "fdir GetFiles :p SetWn [ 10 8 68 20 ]
   Repeat Length :fdir {
      Make "i Rem (GetIter - 1) 40 Make "y IDiv :i 4 Make "x Rem :i 4
      If :i = 0 { ClrWn }{}
      SetPos (List 3 + 15*:x 2 + :y) Type Item GetIter :fdir
      If 39 = :i { SetPos [ 48 12 ] Type "Enter Wait }{ } }
```

```
   If 0 = :i {} { SetPos [ 48 12 ] Type "Enter Wait } SetScrWn
End
```

To produce a presentation system we can introduce the notion of *screen*. Screens are connected among them by *links* which form a graph structure. To support linking each screen contains a special *link-expander* which controls the linking to other screens. A simple example of it is

```
    [ "Link       [ 60 1 ] [ 2 0 ] [ Make "Link "Artist Throw :d ] ]
```

from the command `Prfoks` . More complex link-expander could be obtained by replacing `Make "Link "Artist` by

```
    Make "Link SelectFromList :PrfoksLinkList
```

On each screen we can build a nested (tree) structure of expanders corresponding to actions.
A very interesting possibility is the introduction of *editing actions* which offer us a tool for run-time building or modifications of the system. Such an action is the `Add` event in `Artist` supported by a command:

```
To AddKey :d
   (Local "evt "ch "lab "cmd)
   TxScr Clr SetWn [ 2 8 79 13 ] DrawFrame "' Define Key ' 1 2 "AroundWn 0 1
   SetPos [2 2 ] Type "'Key = ' Loop "z { If KeyPressedP {Throw "z}{} }
   Make "evt RdEKey Make "ch First :evt
   If :ch = 0 { Make "evt (List 1 Last :evt) }{ Make "evt (List 0 :ch) }
   Show :evt SetPos [2 3 ] Type "'Label = ' Make "lab RdWord
   SetRdWnLen 75 SetPos [2 4 ] Type "'Action list = ' SetPos [2 5 ]
   Make "cmd Rd SetPos [2 6 ] SetRdWnLen 15 SetScrWn Clr
   Make "cmd (List :d [ -1 -1 ] :evt :cmd) OutPut AddLast :cmd Thing :d
End
```

We activate it by the `Insert` key. Entering `l` , `logo` and `[Logo]` , where

```
To Logo
   SetAutoFill "On SetFillColor Black
   Box -1535 -1000 1535 -920 SetPos [ 2 43 ] SetRdWnLen 80   ; *** Herc
   Box -1335 -1000 1335 -900 SetPos [ 2 59 ] SetRdWnLen 70   ; *** VGA
   Type "'Logo = ' Make "cc Rd
   SetRdWnLen 15 Run :cc
End
```

we can from now on, by pressing key `l` , use LogoS commands in the `Artist`

5. Conclusion

In the paper we demonstrated the possibility of building an (educational) authoring system based on Logo. The PRFOKS authoring system will provide a user with a set of already prepared *forms* which correspond to typical situations and activities in lessons: messengers, displayers, readers, dialogs, editors, computers, communicators, ...; and capabilities to combine these forms in lessons and run them. Since the system is based on Logo the user can easily extend it by his own forms.

To speed up the execution some critical commands (`GetEvent` , `Handler` , `Expand` , `Editor` , `Match` , ...) can be implemented in Pascal as Logo primitives in a supporting library.

REFERENCES

[1] Batagelj V.: *PRFOKS*, manuscript, Ljubljana, February 1990. (in Slovene)

[2] Batagelj V., Grobelnik M., Zupanic: The Design of LogoS. *Proceedings of the Third European Logo Conference, Parma*, Italy, 27-30 August 1991, A.S.I., Parma 1991, p. 287-296.

[3] Bramble W.J., Mason E.J.: *Computers in Schools.* McGraw-Hill, New York 1985. % MaK 3708

[4] Conlon T.: PILOT - *The Language and How to Use It.* Prentice-Hall, Englewood Cliffs 1984.

[5] Grobelnik M., Zupanic D.: LogoS, Version 1.0. Technical manual, Ljubljana, June 1992. (in Slovene)

[6] Harvey B.: *Computer Science Logo Style*, Volume 2: Projects, Styles, and Techniques. The MIT Press, Cambridge, Massachusetts 1985.

[7] Kropf P.: *MusicStudio*, Version 1.00. Program in LogoS. Computer Science Summer School, Ljubljana, July 6-11, 1992.

[8] Mills G.: *MicroSoft Windows Logo*, Version 3.2. Help, March 1993.

[9] Swan T.: *Mastering Turbo Pascal 6.* Hayden Books, Carmel, Indiana 1991.

[10] Vaughan T.: *Multimedia, Making it Work.* Osborne/McGraw-Hill, New York 1993.

[11] Wolfram S.: *Mathematica, A System for Doing Mathematics by Computer.* Addison-Wesley, Redwood City, CA 1991.

[12] Zuravlev A.P.: *Jazykovye igry na komp'jutere.* Prosvescenie, Moskva 1988.

La Manipulation Directe Comme Invariant des Micromondes 'Educatifs: le cas de Cabri-Géomètre

Bellemain Franck, Dagdilelis Vassilios

Laboratoire de Structures Discrètes et Didactique
BP 53X 38041 Grenoble Cedex
CNRS-IMAG-Université Joseph Fourier, France
tel: (33) 76514561
e-mail: belemain@imag.fr

Abstract

Dans le présent travail, nous proposons une étude des micromondes éducatifs du point de vue de leur interface en essayant de situer les éléments qui caractérisent la communication entre l'utilisateur et le système (manipulation directe, langage de commande) par rapport à la problématique didactique. Comme principal appui, pour notre propos, nous utiliserons Cabri-géomètre - modèle d'un micromonde qui fait un large usage du principe de la manipulation directe pour l'enseignement de la géométrie.

Keywords: user interface, geometry, functions, graphical representation

Bellemain Franck, *Ingénieur Informaticien, est né à Valence (France) en 1963.Après avoir fait sa Maîtrise de Mathématiques à l'Université de Grenoble 1 en 1985, il a preparé une Thèse d' Université en Appliquées intitulée:"Conception, réalisation et expérimentation d' un logiciel d' aide à l' enseignement de la géométrie: Cabri-géomètre", 1992, Université de Grenoble 1. Ses intérêts de recherche sont: Didactique des Mathématiques, Informatique Graphique, EIAO.*

Dagdilelis Vassilios, *Enseignant dans le Secondaire, est né à Thessalonique (Grèce) en 1951. Il a obtenu: Diplôme de Mathématiques d' Université de Thessalonique en 1975, Diplôme d' Etudes Approfondies (DEA) Université de Grenoble 1 en 1986, en didactique des disciplines scientifiques. Ses intérêts de recherche sont: Didactique des Mathématiques, Didactique de l' Informatique, EIAO.*

La Manipulation Directe Comme Invariant des Micromondes 'Educatifs: le cas de Cabri-Géomètre

1. Introduction

Pour des raisons déjà largement admises et analysées (l'échec relatif des systèmes éducatifs informatisés du passé), il y a ces dernières années un grand développement de nouveaux systèmes visant à exploiter les possibilités offertes par l'intelligence artificielle. Comme première réalisation de cette nouvelle génération de logiciel, nous voyons apparaître les micromondes. Ces systèmes évolutifs proposent une représentation d'un domaine de connaissances et peuvent ainsi constituer le noyau d'environnements plus complexes intégrant des modules intelligents. Ils représentent l'évolution récente d'une ligne qui débute avec l'Enseignement Assisté par Ordinateur des années '60.

Le nombre important des environnements «Logo-like» - que nous appelerons plutôt *micromondes éducatifs* - témoigne du caractère particulièrement séduisant de ces environnements. Ce développement nous oblige cependant à une étude systématique de leur fonctionnement - théoriquement établi et expérimentalement éprouvé - afin non pas seulement de mieux exploiter leurs possibilités, mais également pour mieux orienter le *design* des micromondes éducatifs futurs.

A l'intérieur même de l'ensemble des micromondes, nous pouvons constater ces dernières années une évolution importante concernant la définition de l'interface et de l'ensemble des moyens qui permettent la communication entre l'utilisateur et le système. Cette évolution trouve ses origines dans l'évolution technique des périphériques de communication. Mais elle trouve aussi ses origines dans la prise de conscience de l'importance de l'interface pour la compréhension du fonctionnement d'un système. Si dans les premiers micromondes, la définition de l'interface apparaît comme ayant été un peu négligée, aujourd'hui un concept se trouve au centre d'intérêt des recherches concernant les micromondes ; il s'agit du principe de la *manipulation directe* ([3], [4], [5], [12], [14], [15], [17]).

Dans le présent travail, nous proposons une étude des micromondes éducatifs du point de vue de leur interface en essayant de situer les éléments qui caractérisent la communication entre l'utilisateur et le système (manipulation directe, langage de commande) par rapport à la problématique didactique. Comme principal appui, pour notre propos, nous utiliserons Cabri-géomètre - modèle d'un micromonde qui fait un large usage du principe de la manipulation directe pour l'enseignement de la géométrie ([2], [3], [5]).

2. Les micromondes

L'origine précise du concept de *micromonde* n'est pas très connue et doit se situer au début même des années '70 ([14]). Le concept de micromonde semble toutefois être véritablement né avec LOGO ([18]). Selon une définition acceptable aujourd'hui, les micromondes sont constitués d'un ensemble :
- d'objets et de relations,
- d'opérateurs opérant sur ces objets, en modifiant les relations et en créant des nouveaux objets ([14]).

Le micromonde se comporte comme un modèle possédant des éléments et des relations de base qui peuvent être combinés à l'aide d'opérateurs pour produire de nouveaux objets [11]. Il propose aussi une *matérialisation* (ibid.) de ces éléments constitutifs pour en permettre l'exploration, le dégagement de conjectures et leur vérification pragmatique. Ainsi, nous compléterons cette caractérisation en précisant que le micromonde propose aussi une matérialisation des objets et relations qu'il permet de manipuler.

Les micromondes ainsi définis constituent un univers assez vaste, comprenant des langages de commande ou de programmation et une multitude de logiciels comme les tableurs, les bases des données, les traitements de texte et divers logiciels plus particulièrement destinés à l'éducation. L'ordinateur n'est même pas nécessaire : le système mécanique utilisé aux chemins de fer pour déterminer la vitesse de trains en fonction d'un lieu de rencontre des deux locomotives, fournit un exemple d'un micromonde élémentaire, qui n'a rien à voir avec l'informatique (fig.1).

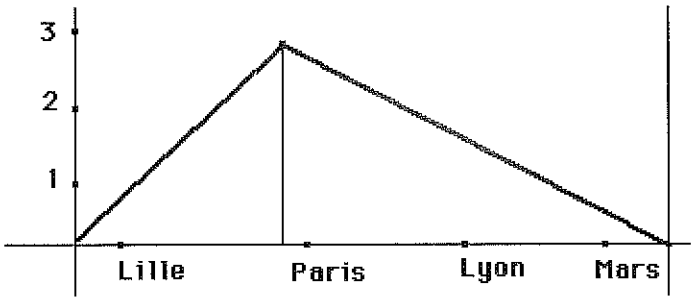


fig. 1 : l'inclinaison des droites détermine la vitesse des trains et la perpendiculaire montre leur lieu de rencontre

Par cette description, le micromonde apparaît comme la simple transplantation dans l'environnement informatique de modèles mathématiques, physiques ou autres. Cependant, cette transplantation n'est pas neutre et l'ordinateur peut apporter des opérateurs, souvent nouveaux et puissants, qui permettent d'envisager des manipulations que l'on ne peut concevoir dans d'autres environnements.

Pour donner un exemple, les traitements de texte offrent des possibilités (copier/couper/coller, de changement de style) bien supérieures à ce qui peut être réalisé dans un environnement papier/crayon.

La transplantation dans l'environnement informatique de modèles et de la matérialisation qui les accompagne, présente d'autres intérêts généraux. L'ordinateur prenant en charge la production, pour chaque objet du modèle élaboré, la matérialisation qui lui correspond, une plus grande possibilité est offerte d'explorer, de multiplier les essais et les tests de conjectures. De la même façon, un micromonde peut ne s'appuyer que sur une partie d'un modèle, évacuant ce qui n'est pas pertinent relativement au domaine d'étude considéré. Une table à coussin d'air (exemple de micromonde proposé par [15]) permet l'étude des seuls mouvements sans frottement.

3. Les micromondes éducatifs

Les *micromondes éducatifs*, par la possibilité qu'ils offrent d'exploration de modèles, permettent d'envisager la mise en place de situations d'enseignement basées sur une démarche originale. Une activité dans un tel environnement est souvent de demander à l'élève de construire de nouveaux objets et de mettre en évidence leurs nouvelles relations par une recherche à caractère expérimental et la mise en évidence de conjectures. Le micromonde permet l'exploration de la matérialisation d'un modèle et, au travers de cette exploration, la mise en évidence du modèle lui-même. L'utilisation d'un micromonde proche d'un modèle mathématique est supposée permettre la construction de connaissances relatives à ce modèle. Ainsi, décrire un carré dans un environnement LOGO, nécessite une perception particulière du carré. Le carré doit être décrit à l'aide des éléments proposés par l'environnement. On peut noter que la grande différence entre l'environnement habituel papier-crayon et un micromonde est que : dans le premier une figure possède les propriétés que l'on a décidé de lui attribuer par une abstraction, tandis que dans le deuxième, ces propriétés deviennent des caractéristiques nécessaires à la description de cette figure.

Tout n'est pas si simple, il est clair que le savoir tel qu'il existe dans la communauté scientifique ne peut pas être enseigné aux élèves sans être transformé, adapté aux contraintes spécifiques du système d'enseignement : contraintes psychologiques, sociales, didactiques, économiques. Dans cette transformation,

lorsqu'elle doit aboutir sur la définition d'un système informatique ou plus particulièrement d'un micromonde, la puissance, mais aussi les contraintes, introduites par l'ordinateur interviennent.

Les situations d'enseignement également doivent faire l'objet d'une attention particulière. On ne peut pas reproduire des situations de découverte scientifique telles quelles, dans la mesure où la problématique du scientifique n'est pas celle de l'élève. On ne peut pas supposer que l'élève tire des connaissances formelles de la seule exploration d'un micromonde, il doit voir l'élaboration de conjectures et la construction de connaissances comme une nécessité liée à la situation dans laquelle il se trouve.

Ainsi, la didactique des mathématiques fait l'hypothèse que l'élève construit ses connaissances en interaction avec un milieu [7], au travers d'une situation, dans laquelle il se trouve généralement confronté à un problème. De ce point de vue, le micromonde intervient comme élément du milieu et les moyens de l'interaction de l'élève avec ce milieu se situent dans la communication entre l'élève et la machine. Deux catégories de moyens permettent à l'élève d'agir sur ce milieu constitué d'un dispositif informatique. La première privilégie la formulation explicite des commandes à appliquer, tandis que la deuxième privilégie la manipulation des objets représentés graphiquement par le dispositif.

Jusqu'à présent, les micromondes éducatifs ont plutôt utilisé le premier moyen de communication. S'il existe une raison technique à ce choix (souris graphique inexistante, ...), il y a aussi une raison didactique : l'adaptation de l'élève aux contraintes de la formulation existantes dans ces environnements, a été considérée comme favorisant la construction de connaissances par celui-ci. Si effectivement l'élaboration de formulations par l'élève présente un intérêt du point de vue de la construction de ses connaissances, nous considérons également que la manipulation directe de représentations d'objets peut, elle aussi, jouer un rôle relativement à cette construction de connaissances.

4. La manipulation directe

La notion de manipulation directe est apparue progressivement avec l'introduction des interfaces graphiques de Rank Xerox. Elle est associée dans le milieu informatique à la notion de *métaphore* (14). Par ces deux idées, les ergonomes ont cherché à représenter des notions informatiques et à permettre la manipulation physique de ces représentations à l'écran de l'ordinateur par l'intermédiaire de la souris graphique. Les premières définitions ([19], [20]) mettent l'accent surtout sur "l'immédiateté" de la manipulation directe, les opérateurs et les objets symbolisés par des icônes plutôt que par des commandes écrites. Des travaux plus récents exigent de la manipulation directe qu'elle comprenne aussi *l'engagement direct* (par exemple [17]). Par cette idée, il ne s'agit plus seulement pour l'utilisateur de choisir des objets et des opérateurs par leur représentation, mais aussi de pouvoir effectuer directement sur ces objets un certain nombre d'actions (déplacement d'un fichier, ...).

Ce principe de la manipulation directe a été jusqu'à présent plutôt mis en oeuvre dans la conception d'interfaces graphiques pour simplifier la mise en Ïuvre de notions informatiques (fichiers, logiciels, ...), que pour permettre la construction de connaissances relatives à ces notions. De notre point de vue, il peut aussi être utilisé dans les micromondes éducatifs pour permettre la manipulation d'objets au travers de leur représentation. La manipulation directe permet à l'élève d'explorer une matérialisation en la manipulant et par l'observation des réactions de l'environnement à ses actions, il peut construire une connaissance des objets sur la représentation desquels il agit directement ou "physiquement".

La géométrie, par l'importance qu'elle accorde aux représentations graphiques, constitue un environnement adéquat pour la mise en Ïuvre de la manipulation directe. En revanche l'utilisation de la manipulation directe est peut être plus obscure dans des environnements où le symbolisme écrit à une part importante, comme l'algèbre. Pourtant, dans une description du cahier de charge d'un futur Cabri-algèbre, Laborde ([14]) cite les exemples des grapheurs à manipulation directe (fig.2).

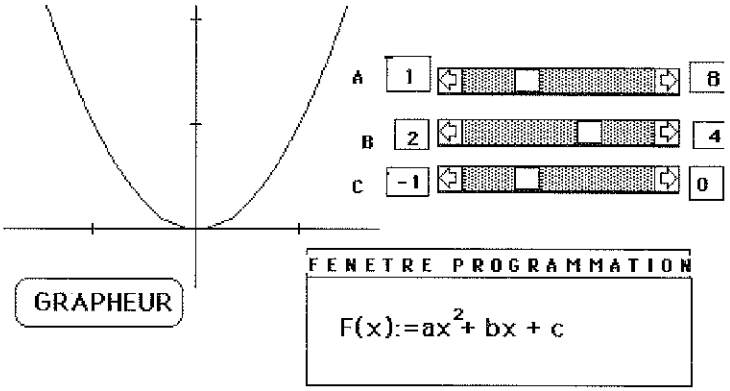


fig. 2 : manipulation directe d'un grapheur où l'action directe sur les bandes de défilement modifie en temps réel la courbe

En fait le concept de la manipulation directe est assez récent et il est probable que son exploitation à fond nécessite, d'une certaine façon, une orientation nouvelle de la construction des didacticiels - micromondes ou non. Ainsi la manipulation directe offre, pour ne donner qu'un seul exemple, des possibilités qui sont encore, à notre connaissance, marginales comme la possibilité d'*inverseurs*. Ces inverseurs constituent un exemple d'applications de la manipulation directe dans des situations qui ne peuvent être envisagées dans les micromondes basés uniquement sur la formulation. Par exemple, dans le cas de Cabri-géomètre, on peut déplacer une droite et observer la modification des coefficients a et b (fig. 3a).

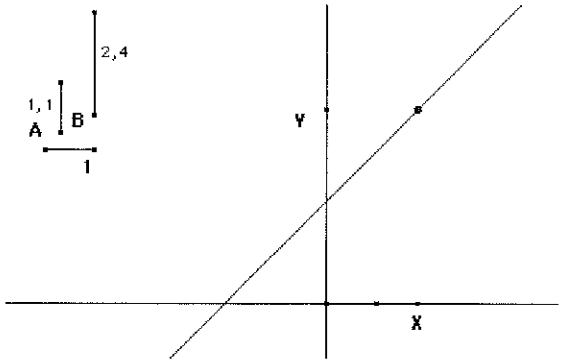


fig. 3a : dans Cabri-géomètre, en déplaçant la droite par les points X et Y, on suit la modification des coefficients A et B de l'équation y=Ax+B.

Dans un grapheur algébrique on peut déplacer la courbe par des "poignées" et suivre la modification des valeurs des a,b,c (fig. 3b).

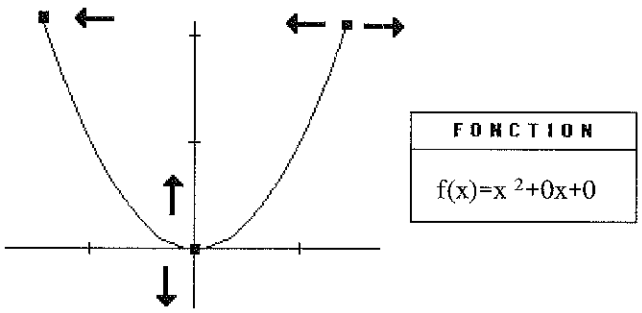


fig. 3b : le grapheur algébrique offre la possibilité de déplacer la parabole et d'observer en temps réel la modification des coefficients a,b,c de f(x)

Même si cette inversion n'est pas toujours possible, l'exploitation de cette voie offre des possibilités nouvelles.

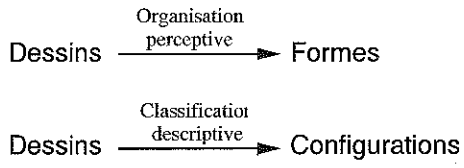
5. Micromondes de géométrie

L'enseignement de la géométrie accorde une place importante à l'utilisation et l'exploration des figures. Il considère en particulier que la géométrie naît d'une mathématisation de l'espace physique et des figures. Dans cette démarche de mathématisation, l'enseignant est amené à gérer une situation contradictoire et se trouve conduit à :

- d'une part, utiliser l'exploration de dessins pour dégager des notions géométriques formelles parce qu'ils leurs donnent du sens en permettant de relier des données graphiques aux données théoriques ;
- d'autre part, rejeter le dessin comme moyen de résolution de problèmes et de validation de propriétés géométriques et s'appuyer pour ces résolutions sur une abstraction de l'espace sensible particulière : celle de la géométrie euclidienne, qui est dégagée de l'exploration de dessins.

Pour gérer la contradiction que ces deux directions font apparaître, l'étude de l'espace et celle de la géométrie euclidienne sont généralement distinguées comme s'il s'agissait de deux objets d'enseignement indépendants. Nous verrons en particulier l'enseignant, lorsqu'il aborde les notions géométriques théoriques s'appuyer sur un contrat implicite auquel l'élève n'a pas nécessairement accès [13]. Il ne valide plus les dessins produit par l'élève, mais la méthode qui a permis de produire ce dessin, alors que l'élève reste souvent au niveau du dessin ([10]).

En fait, il y a deux objets en jeu : l'un étant le dessin concrètement tracé, l'autre étant la figure, abstraction du dessin appartenant au monde mathématique. L'enseignement de la géométrie se trouve ainsi confronté au problème d'amener les élèves à élaborer une abstraction des objets du dessin qui deviennent des objets formels sur lesquels portent notamment les raisonnements déductifs. Pour caractériser cette abstraction et la construction d'une signification du dessin dont elle est issue, nous avons choisi de les considérer sous deux angles différents. Nous avons ainsi décrit cette abstraction comme liée à une classification des dessins et considéré deux classifications : la première consiste à organiser les dessins à partir de caractéristiques graphiques communes. Dans ce cas, l'objet graphique est le point de départ de l'abstraction et de la construction de sa signification. Le deuxième consiste à décrire les dessins à partir des éléments fournis par un modèle (celui de la géométrie euclidienne, en particulier). Dans ce cas, c'est au contraire le formalisme qui permet de caractériser l'objet graphique.



La reconnaissance de formes constitue un moyen, dans l'observation des configurations, de mise en évidence de propriétés géométriques. Dans cette reconnaissance de forme, nous constatons, parce que les formes elle-même ont des caractéristiques implicites, que, tant dans l'élaboration d'un dessin que dans son exploration, des propriétés implicites peuvent intervenir. Ainsi, l'un des rôles de la mathématisation peut consister en l'élargissement des formes aux configurations. Cet élargissement peut notamment s'appuyer sur la généralisation des énoncés concernant les dessins géométriques pour qu'ils concernent des configurations et sur l'explicitation des propriétés implicites pouvant intervenir dans la reconnaissance de forme. L'enseignant doit trouver les moyens qui rendent nécessaire à l'élève cette mathématisation, la généralisation des énoncés et l'explicitation des implicites liés aux formes géométriques.

L'un des apports des micromondes a été ainsi d'introduire, les contraintes de la formulation dans un langage spécifique qui rend nécessaire l'explicitation des notions géométriques et leur formulation dans ce langage (comme Euclide [1], ou LOGO). Si effectivement les contraintes de la formulation peuvent rendre nécessaire l'explicitation et la généralisation de notions géométriques, il y a quelques remarques qui s'imposent quant à l'utilisation de langages spécifiques pour cette formulation.

- parfois, les contraintes de ces langages peuvent ne pas être pertinentes et ne pas avoir de signification du point de vue de la géométrie. Les contraintes syntaxiques de certains langages de commandes en fournissent un exemple ;
- les situations de formulation ne jouent pas le même rôle relativement à l'explicitation des connaissances que d'autres situations qui laissent une plus grande place à l'action. La formulation exige parfois une explicitation telle que certains implicites liés l'action peuvent ne pas être explicités dans les situation de formulation pour réapparaître ensuite dans d'autres situations. L'élève emploie des stratégies de résolution différentes selon qu'il doive formuler ces stratégies ou les mettre en oeuvre ;
- le niveau des connaissances requises pour le travail dans un tel environnement : la description analytique de la figure géométrique nécessite déjà une certaine connaissance des propriétés géométriques des objets en jeu ; par conséquent elle ne peut être utilisée qu'avec des élèves ayant déjà acquis un certain niveau de compétence en géométrie euclidienne.

6. La manipulation directe en géométrie

La manipulation directe permet de poser le problème de la communication avec l'ordinateur à un autre niveau que celui de la formulation. Elle permet aussi de réduire les contraintes liées à la formulation : syntaxe, explicitation de certains implicites parfois difficile, bon niveau de compétence dans la manipulation du langage.

La manipulation directe en géométrie intervient à deux endroits. D'une part, dans la *construction par désignation*, où elle permet par la désignation d'objets d'une figure ou le choix de données arbitraires d'élaborer un dessin. D'autre part, dans la *transformation libre*, où elle permet par simple déplacement d'éléments d'un dessin d'obtenir d'autres dessins possédant les mêmes spécifications géométriques que le dessin initial.

Les objets, au travers de la transformation libre, deviennent des objets dynamiques - au sens qu'ils perdent leur caractère statique pour devenir les représentants d'une classe de figures (équivalentes par rapport à la transformation libre). Leurs propriétés constituent les invariants (visibles) de ces dessins transformables. Cette possibilité de transformation libre offre à l'enseignant l'occasion de modifier les critères de validité d'une construction : la propriété demandée doit rester visiblement invariante pour tous les dessins possibles d'une configuration.

L'enseignant peut utiliser une figure préparée d'avance comme une "boîte noire" en demandant de l'élève la construction d'une figure qui "se comporte" (possède les mêmes propriétés géométriques) que la figure préconstruite - ce processus peut très bien être à son tour informatisé comme c'est le cas du prototype HyperCabri ([8], [9], [21]).

La manipulation directe permet la construction de dessins géométriques avec des exigences de formalisme moindre que celles de la formulation. Elle permet ainsi à l'élève d'élaborer des dessins géométriques sans qu'il ne connaisse complètement les caractéristiques géométriques des objets qu'il construit. Elle lui permet ensuite, par le déplacement, d'explorer ces caractéristiques. La manipulation directe peut ainsi s'adapter à des niveaux de connaissances différents. Nous citons l'exemple des deux figures ci-dessous, dans lesquelles on peut décrire (et même expliquer dans un certain sens) la modification du symétrique du cercle en termes utilisant un *langage familier* avec de notions de "pente", de "proximité", de "grandeur" au lieu de notions mathématiques inaccessibles alors chez les jeunes élèves (fig 4).

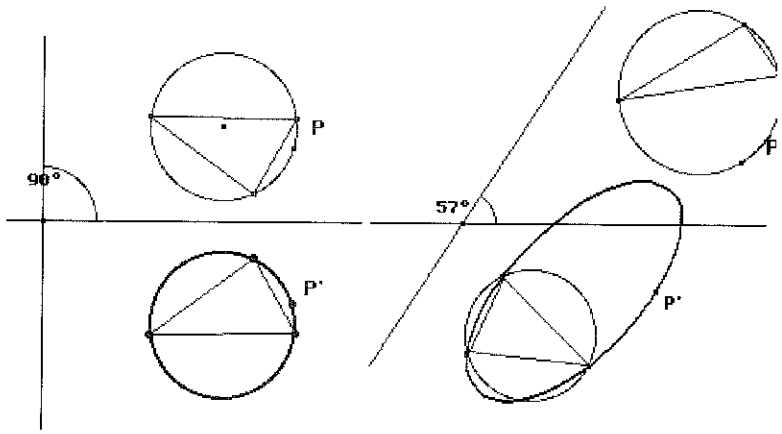


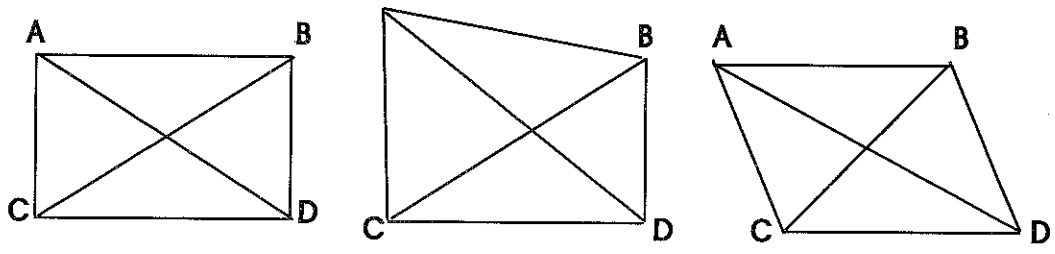
fig. 4 : la manipulation directe dans Cabri-géomètre permet l'exploration des modifications des figures grâce aux mouvements qui peuvent être décrit en termes de "pente", de "proximité", de "grandeur".

Par la manipulation directe, ce sont plutôt les rétroactions de l'environnement, apparaissant dans les différents dessins parcourus au cours de déplacements d'objets, que les contraintes qui doivent favoriser chez les élèves l'adaptation de ses démarches de résolution et l'explicitation de propriétés géométriques des dessins.

Ce qui nous semble plus particulièrement pertinent dans l'utilisation de la manipulation directe pour l'élaboration de l'interface de micromonde est la gestion et l'interprétation des implicites liés à l'action qu'elle permet. Ceci à la différence de la formulation dans un langage de commande qui résoud le problème des implicites en les éliminant de facto : les implicites ne sont pas formulés, l'élimination de ces implicites ne signifiant pas qu'ils n'interviendront pas à nouveau dans d'autres situations.

On peut observer que le problème de la gestion des implicites et des propriétés formelles ne provient pas de leur existence proprement dite, mais plutôt du rôle qu'elles jouent dans la production du discours géométrique. La manipulation directe offre la possibilité de n'expliciter les implicites qu'au moment où ils interviennent effectivement dans le processus de résolution de problème. Pour éclaircir notre propos nous proposons l'exemple suivant : l'élève peut construire par tatônnement un quadrilatère fermé qui visuellement

est un rectangle (fig. 6a). Ainsi, les trois niveaux (la figure du point de vue de l'élève, celle du point de vue de la géométrie et celle du point de vue du logiciel) restent en quelque sorte compatibles. Cependant, par un simple déplacement, le quadrilatère change de forme (fig. 6b). Les propriétés du rectangle ont été implicitement utilisées dans la construction et ne sont pas conservées au cours du déplacement. Cette non-conservation montre que les propriétés doivent être explicitées pour être conservées. Même si dans un stade ultérieur l'élève construit un rectangle qui est en réalité un parallélogramme quelconque, la transformation libre montre effectivement la vraie nature de sa construction (fig. 6c).



(a) rectangle ou quadrilatère?

(b) la transformation montre que c'est un quadrilatère

(c) la transformation montre que c'est un parallélogramme

fig.6

Dans l'environnement papier-crayon, ce qui fait défaut, c'est la figure elle-même, l'enseignant doit s'appuyer sur un contrat implicite pour justifier le rejet des dessins des élèves. Avec la formulation, tout doit être explicités de manière anticipée. De plus "l'immédiateté" des réactions du dispositif accroit le caractère "persuasif" la figure transformée. Même si l'élève prétend que l'ordinateur "ne comprend pas" ce qu'il voulait faire (par une forme d'anthropomorphisme), il nous semble qu'on se trouve devant un problème de nature différente du précédent (le débat entre enseignant et élève sur ce que est un "rectangle quelconque"...).

Dans les micromondes tels que Cabri-géomètre, la gestion des implicites vient "au devant de la scène" au moment opportun.

7. En guise de conclusion

Le développement récent des micromondes ouvre des nouvelles possibilités quant à l'organisation de situations d'apprentissage. L'exploration des objets et de leurs relations, l'élaboration de conjectures et leur validation pragmatique, l'évolution des stratégies des élèves en fonction des rétroactions du milieu, sont des éléments essentiels de ces nouvelles situations d'apprentisage. La manipulation directe joue un rôle important dans la mesure où elle permet à l'élève de mettre en Ïuvre ses propres connaissances, ne pouvant ne pas être complètement formulables avec les objets et les relations du savoir en jeu. L'éventuel conflit entre ces connaissances et les savoirs à enseigner apparaissent au moment où les connaissances de l'élève s'avèrent incompatibles ou insuffisantes pour la résolution du problème.

Si la manipulation directe est restée jusqu'à maintenant relativement peu exploitée pour la réalisation de l'interface des micromondes, elle semble particulièrement adaptée à la création d'environnements de travail permettant à l'élève d'exprimer et mettre à l'épreuve ses propres connaissances - qu'elles soient erronées, insuffisantes ou non pertinentes par rapport au domaine de connaissances concerné.

C'est pour cette raison que nous estimons que la manipulation directe doit constituer un élément constant des micromondes contemporains, qu'elle doit être un invariant de ces micromondes.

BIBLIOGRAPHIE

[1] Allard J.C., Pascal C. [1986] Euclide : un langage pour la géomètrie plane, logiciel et manuel, Cedic-Nathan, 1986, Paris,

[2] Baulac Y., Bellemain F., Laborde J.M. [1988] Cabri-géomètre, un logiciel d'aide à l'apprentissage de la géomètrie, Cedic-Nathan, 1988, Paris.

[3] Baulac Y., Laborde J.M. [1989] Sur l'interface d'un Cahier de brouillon informatique pour la géométrie, Rapport de recherche n) 7401, Grenoble, 1989 LSD2, IMAG.

[4] Bellemain F., Capponi B. [1992] Spécificités de l'organisation d'une séquence d'enseignement lors de l'utilisation de l'ordinateur, Educational Studies in Mathematics, 1992.

[5] Bellemain F. [1992] Conception, réalisation et expérimentation d'un mogiciel d'aide à l'enseignement de la géométrie : Cabri-géomètre, Thèse, Université de Grenoble, 1992.

[6] Bellemain F. [1989] Le logiciel "Cabri-géomètre", un logiciel d'aide à l'enseignement de la géomètrie, Publications de l'Institut de Recherche de Mathématiques de Rennes, (5), 1988-89, Didactique des mathématiques.

[7] Brousseau G. [1986] La relation didactique : le milieu, Actes de la IV Ecole d'été, 1986, Orléans.

[8] Capponi B. [1990] Problèmes des systèmes tutoriels en géomètrie, l'exemple de HyperCabri, Rapport de l'Institut de Didactique des Mathématiques (IDM), 1990, Université de Bielefeld.

[9] Dagdilelis Vassilios [1991] Cabri et Hyper Cabri, un environnement intelligent pour l'enseignement et

l'apprentissage de la géométrie Euclidienne (en grec), 3ème Congrés d'Informatique, Athènes, 1991.

[10] Duval R. [1988] Pour une approche cognitive des problèmes de géomètrie en termes de congruence, Annales de didactique et de sciences cognitives, IREM et Université Louis Pasteur, 1988, Strasbourg, (1), 57-74.

[11] Fey J. T. [1989] Technology and Mathematics Education : a survey of recent development and important problems, Educational Studies in Mathematics, 20(36), pp. 237-271.

[12] Jackiw N. [1988] Geometer's Sketchpad, The Visual Geometry Project, Collège de Swarthmore, USA, 1988.

[13] Keskessa B. [1990] Une étude de résolution de problèmes de lieux géomètriques par des élèves de seconde dans deux environnements, l'un "papier-crayon", l'autre informatique-interactif (Cabri-géomètre), Cahiers de séminaire de didactique des mathématiques et de l'informatique, Grenoble, 1990, LSD2 IMAG.

[14] Laborde C., Laborde J.M. [1991] Cabri-géomètre : Micromondes intelligents et environnement d'apprentisage, Actes des XIII° journées Francophones de l'Informatique, C. Bellissant (Ed.) Grenoble, 1991, IMAG-CNRS.

[15] Laborde J.M. [1989] Designing Intelligent Tutorial Systems : the case of geometry and Cabri-géomètre, IFIP WG 3.1 Working Conference on Educational Software at the Secondary Education Level, 1989, Reykjavik.

[16] Laborde J.M., Strässer R. [1990] Cabri-géomètre a Microworld of Geometry for Guided Discovery Learning, Zentralblatt für Didaktik der Mathematik 5, 1990, 171-177.

[17] Nanard J. [1990] La manipulation directe en interface homme-machine, Thèse de Doctorat d'État, USTL Montpellier, 1990.

[18] Papert S. [1982] Jaillissement de l'esprit, Ordinateurs et apprentissage, Basic Books, N. York, 1980, Flammarion 1982

[19] Schneiderman B. [1982] The future of interactive system and the emergence of direct manipulation, Bahavior and Information Technology, (1), 1982, 237-256.

[20] Smith D.C.,Irby C., Kimball R., Verplanck B., Harslem E. [1982] Designing the Star User Interaface, Byte (7), n° 4, 1982, 242-282.

[21] Strässer R. [1989] Students' Constructions and Proofs in a Computer Environment - Problems and Potentials of a Modeling Experience, Workshop on "Modelling the Student Knowledge - the Case of Geometry", Grenoble, 1989.

Design of Logo Command Names

Marko Grobelnik

Jozef Stefan Institute
Computer Science Department
Artificial Intelligence Laboratory
Jamova 39, 61000 Ljubljana, Slovenia
tel: (+38)(61) 159 199, fax: (+38)(61) 161 029
e-mail: Marko.Grobelnik@ijs.si

Abstract

The paper discusses two main subjects: (1) the problem of using English and/or native language in non-English educational environments, and (2) naming conventions when designing Logo command set. A brief analysis of advantages and disadvantages when using different languages for Logo command set is presented. In the second part, the naming conventions used for the design of the command set in LogoS are defined. Several examples from the LogoS show the effects of the proposed solutions.

Keywords: design of non-English Logo, naming conventions

Marko Grobelnik *was born on July 18, 1966 in Kranj (Slovenia). He started studies in the field of computer science at the Faculty of Electrical Engineering and Computer Science in Ljubljana in 1983. Since the year 1984 he has been incorporated in the work of the Artificial Intelligence Laboratory at the J. Stefan Institute in Ljubljana. He participated in several projects as a researcher and/or implementator. Currently, his main research interest is in the field of logic programming, machine learning and search techniques.*

Design of Logo Command Names

1. Introduction

When talking about command names in Logo, one should consider several issues. There is noticeable kernel of approximately 50 commands forming de-facto Logo standard, used in most Logo implementations. But this command set is obviously not enough. Every typical concrete Logo implementation has a set of its specific commands covering various operating system and other nonstandard features. The last but definitely not the least issue is translation of command names to non English languages.

We made an analysis of needs and possibilities when designing new Logo command set. The results were applied in our Logo implementation called LogoS [1, 2] serving as an example throughout this paper. The next section briefly describes some interesting design issues of LogoS. The third section presents different approaches to teaching of Logo in non English environments. The last section before conclusion analyses naming conventions of Logo commands and proposes some solutions used also in LogoS.

2. Some words about LogoS

LogoS is an integrated environment around Logo interpreter and semicompiler. It defines version of Logo language which is a superset of de-facto standard Logo language. De-facto standard which is widely spread throughout most Logo implementations concerns lexical, syntactical and semantical level of the Logo language and 50-100 standard commands and functions including its English names.

Almost every concrete version of Logo language has its own specific set of commands and functions. LogoS version of Logo includes currently 642 built-in names, where 458 of them are used for executable commands and functions. The rest of names define meta commands and descriptive constants. Executable commands and functions are semantically divided in 16 groups (e.g.: turtles, graphics, sound, ...). Some of them are also further divided into subgroups (e.g. group of commands dealing with turtles has subgroups for moving turtles, turning turtles, defining turtle and pen features, ...).

When designing command set, most attention was payed to the following criterions:

- to make use of possibilities offered by a computer,
- command set must be complete,
- functionality of commands must not overlap,
- whenever possible, commands must agree to the commands from de-facto standard,
- commands must form semantically clear groups and subgroups enabling transparency and easy memorising,
- command naming convention must be simple and powerful,
- command naming must be systematical and consistent.

3. Translation of command names to non-English languages

A lot of words have been said and written to clear out what is the real role of translated Logo language at particular levels of education. There are some important tradeoffs, which must be solved when making decisions about the native language interface to the Logo system. In short, most of the different opinions can be summarised in the following three points:

1. **All commands should be translated to a children's native (possibly non-English) language.** The main advantage of this option lies in the opinion that a child doesn't understand the meaning of the words from non-native (typically English) language. The child doesn't connect the name of the command with its functionality. This concerns mainly child's interest in the Logo and the ease of the learning and teaching. The argument against this option is that a child get used to the native language communication to the computer. This causes more difficult customisation of most computer programs which have its user interface mainly in English.

2. **Only a part of the command set, used for teaching in lower school classes, should be translated. Other commands should be in English.** The positive side of this opinion is that the usual later transition to the usage of English is more contiguous and that unnecessary learning of the English version of command names is spared to the uninterested pupils. Negative side is the possible confusion because of the non-uniform approach to the teaching of the same conceptions. Such an approach with two languages (native at the beginning and English later) also causes differences (or barrier) between average and advanced pupils.

3. **Translation of command names is not necessary.** The main argument for this claim is, that there is no need for a child to connect the name of a command with its functionality - similar effect appears when memorising abbreviations. The obvious argument against is that this is simply not true, that comprehension of the command names strongly influences the ease of the memorising which is crucial when teaching something.

In LogoS we decided for the first option. We made some minor experiments with young (7-11 years) children. At the beginning we showed them only the version of LogoS with only 10 translated commands to the Slovene language (others were in English) which were not enough to make something more serious. Children were not particularly interested in using non-Slovene commands because they didn't understand their meaning and they were not prepared to artificially memorise some *'strange'* words. All this reflected in lower interest in working with Logo. Afterwards we exchanged the version of LogoS with the one with complete Slovene command names. The response of the children was of course much more positive. The same commands which were before only in English became at that moment much more interesting and intuitive for the same children.

4. Command-naming conventions

The importance of the appropriate naming conventions was already mentioned in the second section. It greatly influences the understandability of the complete command set. The naming of the commands must be, first of all, simple and powerful. To achieve this goal in LogoS, we adapted the following guiding rules:

- command names must be as short as possible,

- abbreviations are not appreciated,

- command name must clearly indicate command functionality,

- the same conception must be always described with the same word,

- two different conceptions are never described with the same word,

- if possible, the command name should indicate command's group or subgroup,

- whenever possible, the names from de-facto Logo standard should be used (for the English version only)

- strict usage of this rules.

Next, we'll present the naming conventions for the English version of LogoS.

Typical command (sub)group is formed around short *keyword*, representing characteristic (sub)group feature. As an example, we'll show two subgroups of commands, first dealing with filling regions of screen, and the second dealing with small pictures called *'sprites'*. The characteristic keyword for the first subgroup is **Fill**. Commands, belonging to this subgroup are the following:

SetFillStyle, **GetFillStyle**,
SetFillUserStyle, **GetFillUserStyle**,
SetFillColor, **GetFillColor**,
SetFillMode, **GetFillMode**,
Fill

The second subgroup has the keyword **Sprite** and comprise the following commands:

SetSpritePut, **GetSpritePut**,
PutSprite, **GetSprite**,
GetSprites, **SpriteP**,
DelSprite, **DelAllSprites**

From the both above examples is obvious, that related command names are constructed by adding prefixes and/or suffixes to a base keyword. Most frequently, the type of action performed by a command, is denoted by the one of the prefixes **Set**, **Get**, **Del** or suffix **P**. Subarea command names are constructed by adding a descriptive word to a corresponding keyword (e.g.: the names for the subarea commands denoting the color for filling of a region are constructed by adding the word **Color** to the keyword **Fill**).

The described naming convention is used in majority cases. The main reason for minor exceptions is usage of names from de-facto Logo standard. An example of such command names are the following:

Left, **Right** and **Towards**.

If needed, such a group is completed with some additional commands. Their names are constructed with

the naming convention, as described before. As an example, the command group dealing with the turtle direction includes, besides the above three commands, also the following two commands for direct manipulation with the turtle direction:

SetTrDir and **GetTrDir**

In addition to executable commands there exist also several groups of descriptive constants. For example, the group of constants denoting built-in graphical fonts consists of following 11 words:

NormalFont, **RomanFont**, **BoldFont**,
ItalicFont, **SansSerifFont**, **ScriptFont**,
GothicFont, **SmallSimpleFont**, **MediumSimpleFont**,
BigSimpleFont, **UserFont**

All constants have the same suffix, the keyword **Font**. The same convention is adapted for all other groups of constants, namely each group has its own keyword used as prefix or suffix in constants.

5. Conclusion

In paper we presented a part of the project LogoS concerning the design of command names. The first part of this work was to identify possibilities and to evaluate them when using English and/or native language names in non-English educational environments. Among three options, we decided for the one, with completely translated names to the children's native language. The second part of the work was to specify the set of rules which will result in a simple and powerful naming of the command set. The outcome were nicely shaped and useful command names in English and Slovene language.

REFERENCES

[1] Batagelj, V., Grobelnik, M. and Zupanic, D. (1991) The design of LogoS, *Third European Logo Conference Proceedings*, pp. 287-296, Parma, Italy.

[2] Grobelnik, M. and Zupanic, D. (1991) Inside LogoS implementation, *Third European Logo Conference Proceedings*, pp. 297-311, Parma, Italy.

AlgoArena: a System for Learning Programming through Social Activities

Hiroshi KATO[1], Akiko IDE[2]

C&C Information Technology Research Laboratories,
NEC Corporation, Japan
1-1, Miyazaki 4-chome, Miyamae-ku, Kawasaki,
Kanagawa, 216, Japan
tel: +8-44-856-2317fFax: +8-44-856-2239
e-mail: [1] kato@joke.cl.nec.co.jp
[2] akiko@joke.cl.nec.co.jp

Abstract

A system for introductory programming education - AlgoArena - and principles of its design are described. It is a simulation of 'Sumo' wrestling, in which students program the actions of their Sumo wrestler with a Logo-based programming language so as to play matches against one another. On the basis of situated learning, it is designed to facilitate learning by cognitive apprenticeship [2]. The authors intend for the system to be the core of a student's community of practice [9], where social interactions among students are encouraged. In contrast, conventional microworld systems enclose a student in a socially closed field, where individual students interact only with the microworld system. We focus on designing the social setting for a learning environment through the system, specifically, forming and maintaining a community of practice for software development. The system is currently under development.

Keywords: microworld, situated learning, programming education, simulation, Logo

Hiroshi Kato *was born in Japan in 1959. He received B. E. and M. E. degrees all in Electronics Engineering from Keio University, Japan, in 1981 and 1983, respectively. After graduation, he joined NEC Corporation. He is currently an assistant manager of Applied Information Technology Research Laboratory, C&C Information Technology Laboratories, NEC Corporation. He had been a visiting researcher at CSLI, Stanford University, U.S.A. from 1989 to 1990. His research interests include application of cognitive science to education systems.*

Akiko Ide *was born in Japan in 1967. She received B. E. and M. E. degrees all in Management Engineering from Aoyama-gakuin University, Japan, in 1989 and 1991, respectively. After graduation, she joined NEC Corporation. She is currently a researcher of Applied Information Technology Research Laboratory, C&C Information Technology Laboratories, NEC Corporation. Her research interests include application of cognitive science to education systems.*

AlgoArena: a System for Learning Programming through Social Interactions

1 System overview of AlgoArena

AlgoArena [5, 8] is a learning environment for introductory programming education for novices at school. We focus on problem-solving activities through an algorithmic way of thinking and systematic view of things rather than teaching a specific programming language.

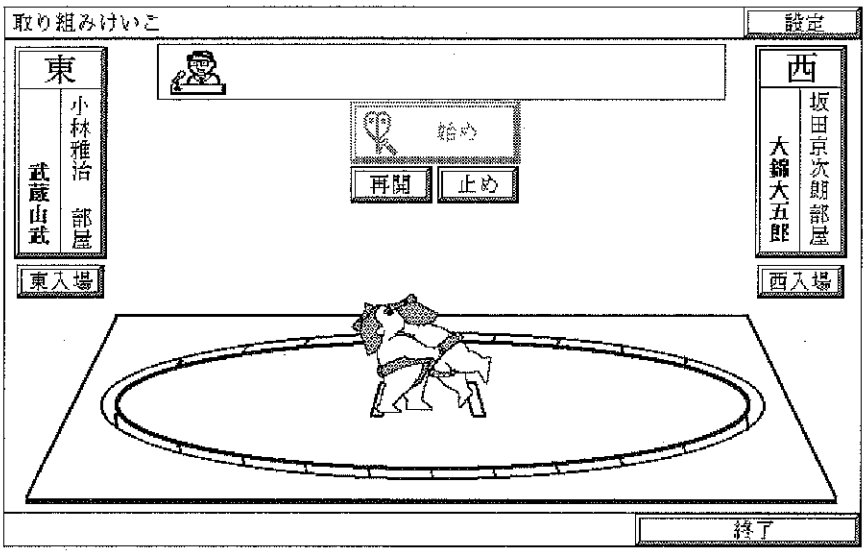


Fig.1 Screen Design of AlgoArena (mockup)

The system works on MS-DOS on a NEC PC-9801 personal computer with 80286 CPU or later. The computer has a graphic screen of 640*400 resolution and 16 colors. A mouse is necessary, and local area network and a sound board are optional.

Shown in Fig.1 is an example of the game of 'Sumo' wrestling — a traditional Japanese national sport — in which players (students) programs strategic actions of their own wrestler with a Logo-based programming language. The wrestler fights with an opponent programmed by another learner or a teacher, and the animation is seen on the graphic screen of the personal computer.
Fig.2 shows a small example of the program specifying strategic actions of a Sumo wrestler. The program takes the following strategy for thirty times: if my wrestler is on the edge of the ring (`my-position=10`), let him go forward. Then, if the opponent is bending deeply (`his-balance=4`), push him down. Otherwise, throw him. (According to Sumo rules, a Sumo wrestler loses when a part of his body other than soles touches the ground or when he steps out of the ring.)

```
repeat 30
[ifelse my-position = 10
     [go-forward]
     [ifelse his-balance = 4
          [push down]
          [throw]
     ]]
```

Fig. 2 Example of a Program

The system is composed of two subsystems: 'training-gym' and 'arena'. The former is for each student, playing the role of a trainer, to program and evaluate his/her own Sumo wrestler(s). The latter is for a teacher or an administrator to gather the wrestlers from the training-gyms to play matches, and for their trainers (students) to enjoy watching the matches together. As to the types of the matches, both team match and individual match are available. The game rules are designed so that, as the student's programming skills improve, his/her wrestler(s) gets stronger.

The game of AlgoArena proceeds as follows. First, a pair of wrestler programs run simultaneously by referring to the current state of the game, such as position in the ring and state of balance. When the next action command for each wrestler is decided by the program, both commands are evaluated at once; consequently, the game state changes according to the pair of commands and the current state of the game. This sequence is called a 'turn' of the game. Turns iterate either until the game state reaches a final one or up to thirty times. If the game state does not reach a final state after thirty turns, the match is drawn.

Compared to similar games such as CoreWars, RoboWar, and Crobots, AlgoArena is designed to pay special attention to its educational purpose, particularly for computer novices. It provides direct manipulation in the human interface, game rules easy to understand by analogy to a popular sport, simple operation, a visual language based on structured flowchart (will be developed in the future version) and so on. This user-friendly interface enables students to devote themselves to the problem-solving itself right away, whereas old-fashioned text-based interfaces force users to learn how to operate the system before getting down to the problem.

According to conventional terminology, AlgoArena is something like a microworld. It differs, however, from a traditional microworld in a major way. The microworld provides a learner with a virtual world, and allows him/her to observe and explore it. What is to be learned there is implicitly represented in interactions between the learner and the microworld. Therefore, interactions in a microworld are closed socially as well as logically. On the other hand, AlgoArena facilitates learning through social interactions in a community of practice consisting of learners and a teacher. What is to be learned is not in interactions with AlgoArena, which is merely a place where students apply what they have learned, but in interactions with other students and a teacher. It plays a central role in a socially open field of learning by attracting learners with the game. In this way, AlgoArena serves as an arena for cooperative learning.

'AlgoArena' is named after 'algorithm' and 'arena': the former stands for the domain that the educational system aims at, and the latter stands for both the literal battleground and the field of cognitive activities.

2 Principles of design

Principles [7] in designing AlgoArena are discussed in this chapter. The principles are based on the standpoint of situated learning, which claims that learning occurs in the process of participation with a community of practice. What we want to achieve through the principles is one thing, i.e., to form a community of practice in a learning environment.

The principles are as follows: cooperative learning, activity with reality, and deculturation from school. In short, the first principle emphasizes the necessity of a community where learning takes place, the second claims that the community must not be for learning itself, but for cultural practices, in other words, authentic activities [2], and the third warns of a trap which conventional educators are apt to fall into.

2.1 Cooperative learning

According to the situated learning theory, learning is viewed as a social activity in an essential sense - a process of enculturation [2] to a community of practice. If so, it is natural to perform education in a group.

Ideally, the group should be an actual community of practice in the real world, but we think, as will be discussed later, that it does not have to be an *'actual'* community. We suppose that even a group of learners, such as a class at school, can be a good field of situated learning, as long as members of the group have a common intention in terms of the practice and that the group includes some authentic practitioners with relatively high expertise.

It should be emphasized that participating in a community includes acquiring informal matters, such as a way of narration among members and an implicit view of things commonly shared in the community, as well as formal matters, such as legitimate procedures to work with. For example, participating in a programmer's community does not mean only to develop programs, but also includes to talk like a programmer with computer jargons. Therefore, it is important for a learner to participate in the community as a legitimate member, and making learners engage only in 'official' works should be avoided.

There are, we think, at least two ways to support cooperative learning. One way is to develop an 'open tool' [4] for learning, which makes participants' interactions with the tool overt so that participants can socially share the works in a natural manner. Previously, we have developed such a system: 'AlgoBlock' [10, 11]. The system makes the programming activity, which is originally a personal activity, open to other participants, so that the process of thought can be socially shared by group members. However, we will not discuss this system any further, as it is not the purpose of this paper. Another way is to design a social setting of a learning environment, which AlgoArena is intended for. The AlgoArena's constraint that a participant needs an opponent to play matches with, can restructure a community of students. In such a sense, a social setting can be intentionally designed through AlgoArena. The following are devices that we have incorporated in AlgoArena for restructuring a community of students.

2.1.1 Forming a community of practice

A common sense of value among group members is necessary for the group to function as a community, where authentic activities take place. Shared intentionality toward practice helps to unite a community.

In the activities using AlgoArena, playing matches with others is inevitably included. Therefore, using AlgoArena requires a group work in some ways. In order to have fun with the game, students have to share the same sense of values that a stronger wrestler is better. The shared sense of value produces a community in which authentic activities for practice take place.

In addition, team game as well as individual game is available. Team game is helpful to facilitate cooperation among members of a small group.

In the process of enjoying the game, social interactions among students, such as competition and cooperation, are naturally promoted.

2.1.2 Maintaining a community of practice

As mentioned above, it becomes a common objective among the members of the AlgoArena community to make his/her own wrestler stronger. On the other hand, our educational objective is to promote students' programming skills in a wide sense, i.e., programming skills including ability of systematic analysis of state of affairs, logical consideration for exhaustive cases, setting a target to solve, inventing specific procedures to follow, and so on. In order to connect these objectives, we have designed the game so that a wrestler gets stronger in proportion to the elaborateness of the program, as is described in the following part of this section. Hence, it spontaneously becomes one of the virtues in the community to devise a better program.

For that purpose, we designed the game rules so that accidentality does not affect the result of the match. First, state transition rules of the game space are not probabilistic, but deterministic. Second, a pair of action commands is evaluated at once, i.e., it is not a game of perfect information, for fear that the first mover should have an advantage. Third, the program language used in AlgoArena has no random function.

In addition, there is virtually no command sequence with which one can always win the game. Because, if one can find such a sequence easily, he/she will not improve the program any more, and the others may lose their motivation. Consequently, profundity of the game is essential for maintaining the practices in the community.

2.2 Activity with reality

Authentic activities by a learner are essential for learning through cognitive apprenticeship, i.e., a learner is supposed to participate in a community of practice, acting as a legitimate member of the community by taking after senior members or experts.

The strictest interpretation of legitimate peripheral participation (LPP) [9] demands 'on the job training' (OJT). In reality, however, it is not always possible for a learner to become a member of an actual community as an apprentice so as just to learn. For example, if some novices, who are temporary apprentices intended just for learning (e.g., school students), are repeatedly put into a team of experienced members, the team may turn unproductive because of much confusion.

So, we would like to raise a question, i.e., whether authentic activities are ever done without joining an actual community in the real world. We suppose that authentic activities can be achieved even in educational situations such as schooling, and AlgoArena is our approach to its solution. We have designed it for classroom use where there are only a few experts, including the teacher and, possibly, some computer buffs who happen to be in the class (as is often the case in Japan). The community of AlgoArena is, therefore, something different from actual community of software development in the real world.

In addition, there are fundamental problems concerning the enculturation: First, denotation of the culture cannot be determined by nature. Second, the teacher, as well as the learner, cannot determine which culture the student should belong to. This is also the case with AlgoArena.

In order to find answers to these questions, we think, first of all, that it is essential to figure out what 'authentic activity' is.

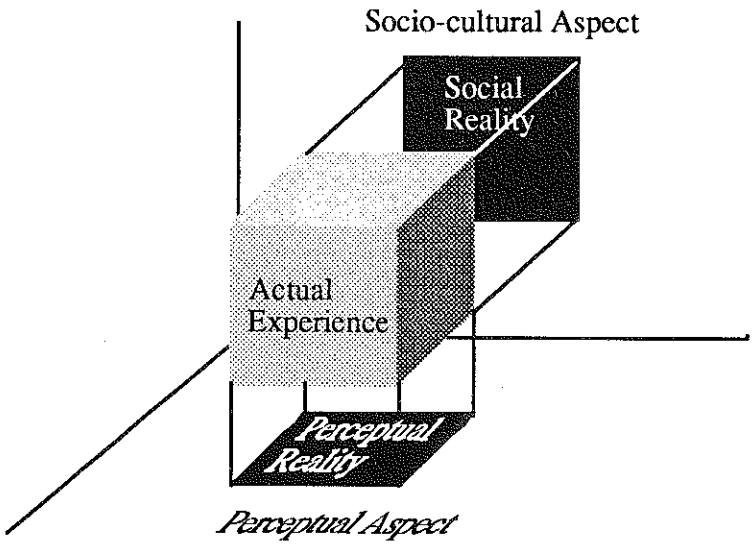


Fig. 3 Relationship between social reality and perceptual reality

We have claimed that real experience in authentic activities includes both perceptual reality and social reality [6]. The former concerns the perception originated from physical stimuli, and the latter concerns the socio-cultural commitment accompanying actions. These kinds of realities, as shown in Fig.3, are independent of each other, though connected by actual experience. Concerning education, perceptual and social realities are related to the learning materials and social setting of the learning environment, respectively.

For example, suppose a method of programming education in which learners in a didactic atmosphere are obliged to use the same tools and methodology as professional programmers employ, and learners make fragments of a practically useless program. Usually, those exercises are problems that have already been solved, and, therefore, have model answers. Solving the problem is not valuable by itself, but valuable only for educational purpose. According to our theory, this is a case in which perceptual reality is thought to outweigh social reality, because learner's activities look like that of programmer's while the social situation is completely different. In contrast, there is another method, like AlgoArena, in which, whatever the tools are, a social setting is designed so that learners spontaneously pursue the same goal by means of programming. This is the case in which social reality is given priority by comparison to perceptual reality, because, in this case, the social situation of learners is similar to that of authentic problem-solvers, such as researchers or programmers.

Although both kinds of reality are necessary for situated learning, we believe that social reality is more important. First, provided that learning is a socially situated activity in essence, the social setting of a learning environment is more influential than the physical one. Second, the subjective reality perceived through an artifact can change according to the familiarity with the artifact; perceived reality on objects can increase as the learner gains practice in the utilization of that artifact, while social reality is not affected by practice. Thus, authors would like to emphasize the importance of social reality, specifically, the social setting of a learning environment, although we appreciate the importance of perceptual reality in learning materials as well. This is why we stand by the latter case of the examples above.

Among the various kinds of authentic activities constituting the culture of software development, we think that the most essential and central activities are to analyze the situation, to set a target, and to devise methods to attain an end. In addition, it is also important that such activities are done in the context of solving a specific problem to which the programmer is committed, i.e., in our words: with social reality. Similarly, in AlgoArena, a learner is encouraged to analyze the weak points of his own wrestler as well as the opponents, make a plan for reinforcement, and design procedures while keeping various constraints in balance; and these activities occur spontaneously under the learner's own initiative. In terms of social reality, activities in AlgoArena can be regarded as an 'epitome' of activities in actual software development.

2.3 Deculturation from school

It is a well-recognized educational problem that knowledge taught in school often fails to be applied to problems in everyday life. This reveals a gulf between school culture and practice culture. We view this problem as a consequence of learner's enculturation to school culture.

Arimoto [1] reported an experiment in which sentence quizzes, of simple multiplication tasks, were given to 10 and 11 years old students. One of the quizzes was irrational in real-life context; e.g., "There are four apples and seven oranges; how many are they if you multiply them?" Seventy out of seventy-seven students answered "28" without any questions, in spite of this being a bizarre and nonsensical problem. When the irrationality of the problem was pointed out by an experimenter, students admitted it. At the same time, however, they insisted that the problem was not bizarre as it was only a math problem. In fact, such an understanding can often be very efficient for doing well at school. As this example shows, students have a peculiar view toward problems in school mathematics.

Thus, a school has its own culture that is rather different from the everyday culture in which knowledge and skills are practically utilized. It might, therefore, be reasonable to assume that students enculturated to school

cannot easily apply the learned knowledge to actual practices.

Consequently, we claim that, in order to acquire knowledge applicable to their actual practices, students should be taken out of the conventional school culture. They should engage themselves in authentic activities and position their activities in actual practices. As a first step, it is important to avoid a 'guessing game' [3] situation in which students, in cooperation with the teacher, try to find out a model answer concealed by the teacher.

Learner's activities in AlgoArena are spontaneous in principle, although some instructions may be needed occasionally. In actuality, the game rule is designed so that there is no absolute way to win — several solutions can coexist. Therefore, the 'guessing game' situation does not happen in learning with AlgoArena. Rather, using the system is more like playing a game than being tutored at school. This stands in good contrast to conventional means of programming education, in which instructions followed by fragmentary exercises are given.

3 Conclusion

The authors have proposed, in this paper, some principles in designing an educational system for situated learning, and introduced a learning environment for program education: 'AlgoArena', which follows the principles. What we want to claim here is the importance of social aspects in system design. Of course, how to use the system is important for social setting in a learning environment, but constraints incorporated in the system can also be significantly influential. The way of use and system constraints are mutually compensable. Therefore, developers of an educational system should be aware that the system design includes the design of social situations as well as functions in a conventional sense. We believe that AlgoArena's system design can be a good example of our claims.

AlgoArena system is currently under development. A first prototype including animated matches will be ready by August 1993. The remaining functions, such as a program editor, team match administrator, etc., will be present in the following versions. Improving rule space of the game is our task at present.

Acknowledgment

The authors would like to express special thanks to Mr. Hideyuki Suzuki, our colleague, for many thought-provoking discussions.

REFERENCES

[1] Arimoto, N.: "Turing test at classroom" revue de la pensée d´aujourd´ hui, Seido-sha, Vol. 19-6 pp. 157-165 (1991) (in Japanese)

[2] Brown, J. S., Collins, A. and Duguid P.: "Situated Cognition and the Culture of Learning", IRL Report No. IRL88-0008 (1988)

[3] Edwards, D.: "Classroom discourse and classroom knowledge" Rogers and Kafocke (ed.) *Social Psychology and passway*. Education (1988)

[4] Hutchins, E.: "The Technology of Team Navigation", Intellectual Teamwork, Galegher, Kraut and Egido (Eds.) pp.191-220, Erlbaum (1990)

[5] IDE, A. and Kato, H.: "AlogoArena: a learning environment for software education (2) — System Design —", Proceedings of the 1993 IEICE spring conference, Part 1, pp.1-477-1-478

[6] Kato, H. and Suzuki, H.: "On the concept of reality in education" Proceedings of the annual meeting of the Japanese Cognitive Science Society PA 2.4 pp.32-33 (1992) (in Japanese)

[7] Kato, H. Suzuki, H. and Ide, A.: "Educational Media for Situated Learning -- Fuidelines for Computer Use in Education--", Technical Report of IEICE ET92-97 (Dec. 1992) (in Japanese)

[8] Kato, H. and Ide, A.: "AlogoArena: a learning environment for software education (1) -- Design concept and rationales --", Proceedings of the 1993 IEICE spring conference, Part 1, pp.1-475-1-476

[9] Lave, J. and Wenger, E.: Situated learning --Legitimate peripheral participation--, Cambridge University Press (1991)

[10[Suzuki, H. and Kato, H.: "AlgoBlock "A Physical Programming Language for Algorithm Education", Proceedings of 8th Symposium on Human Interface, pp.245-248 (1992) (in Japanese)

[11] Suzuki, H. and Kato, H.: "A tangible programming language: AlgoBlock --a tool for thought and collaboration--", Proceedings of 46th annual meetings of IPSJ Vol.1 pp. 9-10, 1Q-5 (1993) (in Japanese)

Cognitive Design Principles of a Generative Logo-like Learning Environment for Promoting Abstract Knowledge Acquisition and Conceptual Change

Sigmar Olaf Tergan[1], Uwe Oestermeier[2]

[1] Deutsches Institut für Fernstudien
an der Universität Tübingen Arbeitsbereich Lernforschung,
de Konrad-Adenauer-Str. 40, W-7400 Tübingen (FRG)
e-mail: tergan@mailserv.zdv.uni-tuebingen.

[2] Wilhelm-Schickard-Institut für Informatik,
Sand 13, W-7400 Tübingen (FRG)

Abstract

As research has revealed existing instructional approaches have not been very successful in developing theories and effective strategies for promoting conceptual change [4]. There is a need for research concerning the cognitive and instructional conditions that may foster processes of abstract knowledge acquisition under conditions of alternative conceptions. LOGO-like computer-based learning systems embedded in powerful learning environments have been suggested to promote effective constructive learning [8]. These systems seem to have the potential to overcome some of the shortcomings of traditional approaches and to facilitate conceptual change. In the following some basic cognitive design principles are outlined the principles concerning the development of LOGO-like learning environments aiming at promoting abstract knowledge acquisition and conceptual change. GELERIAT, a generative computer-based learning environment is presented to demonstrate the implementation of the design principles. GELERIAT has been developed to promote systematic research on the cognitive conditions and instructional implications for promoting conceptual change in the domain of elementary mechanics.

Keywords: conceptual change, software design, cognitive research, exploratory learning environment, interactive learning

Sigmar-Olaf Tergan *was born in 1944. He received his diploma in Psycology. He is researcher at the "Deutsches Institute für Fernstudien an der Universität Tübingen, Arbeitsbereich Lernforschung" (German Institute for Distance Studies at the University of Tübingen, Learning Research Department). His working areas are cognitive research on learning with interactive media; instructional design of computer-based learning environments; learning with interactive systems in further education.*

Uwe Oestermeier *was born in 1963. He received M.A. in Philosophie and Psycology. He is Computer Scientist at the "Wilhelm-Schickard-Institut für Informatik". His working areas are learner-modelling in ITS, computer-based learning environments, psychological theories of reasoning and connectionist reasoning systems (graphical forms of reasoning).*

Cognitive design principles of a generative Logo-like learning environment for promoting abstract knowledge acquisition and conceptual change

1. Introduction

Instruction so far has not been very successful in developing effective approaches for promoting abstract knowledge acquisition and conceptual change [15; 12; 3]. The failure of existing theories and instructional approaches is not at least due to the fact that the design of instruction until recently was oriented toward an information/ verifying/practicing approach following the behavioristic paradigm of knowledge transmission. The design principle underlying traditional instructional approaches is viewed to be inadequate to foster the acquisition of abstract knowledge and higher order learning [15]. In traditional approaches hardly any attention has been paid to cognitive conditions and cognitive processes potentially relevant for conceptual change. There is a need for research concerning the cognitive and instructional conditions that may foster processes of conceptual change. LOGO-like computer-based learning systems embedded in powerful learning environments [7] that promote constructive learning and that provide help and guidance seem to have the potential to overcome some of the shortcomings of traditional approaches and to promote cognitive processes of conceptual change.

In the following some basic cognitive design principles are outlined the principles concerning the development of LOGO-like learning environments aiming at promoting abstract knowledge acquisition and conceptual change. GELERIAT, a generative computer-based learning environment is presented to demonstrate the implementation of the design principles.

2. Cognitive conditions and instructional design principles for promoting knowledge restructuring

According to theory and empirical research there are some cognitive variables that are assumed to be crucial for processes of knowledge restructuring: (1) Abstract knowledge acquisition, (2) constructive cognitive processing, (3) self-regulation and metacognition, (4) qualitative reasoning, (5) structure oriented cognitive processing, (6) inductive analogical reasoning and analogical transfer, (7) "situatedness" of cognitions, (8) resolution of cognitive conflicts, (9) social embedding of cognitions, (10) cognitive sensitivity and emotional willingness.

1. Abstract knowledge acquisition

When students are confronted with the problem of explaining a particular physical phenomenon or to solve problems based on a deep understanding of the subject matter domain the cognitive task is to use abstract knowledge concerning concepts, principles, and structural relations inherent in the domain and to construct an internal model of the state of affairs [14]. It is abstract knowledge as a prerequisite for a reflective application of procedural skills that enable the learner to cope effectively with problems under conditions of different surface features and contexts [15].

Many individual misconceptions refer to inappropriate abstract knowledge about concepts and structural relations and manifest themselves in the construction of mental models and in problem solutions that are not in accordance with physics [16]. In order to promote restructuring of misconceptions instructional events in learning environments, thus, have to focus on fostering abstract knowledge acquisition instead of the acquisition of procedural skills [15].

2. Constructive cognitive processing

The constructivist view which is central for the cognitive approach to comprehension, problem solving and knowledge acquisition has been fruitfully applied to problems of conceptual change [8]. The basic assumption of constructivism concerning abstract knowledge acquisition is that concepts are constructed by active cognitive processing on the basis of an individual's pre-knowledge instead of being imparted by instruction and rather passively internalized by the learner [12]. Cognitive processes must, however, be facilitated and scaffolded by providing adequate instructional and learning events [6; 7].
Constructive cognitive processes that may result in cognitive change are supposed to be promoted in exploratory learning environments that provide personal experiences by means of simulating aspects of real world phenomena [5].

3. Self regulation and metacognition

In self-regulated learning and problem solving, appropriate self-regulatory metacognitive processes such as monitoring understanding and reflections on one's difficulties in learning and problem solving are considered to be crucial determinants for individual learning progress. Processes of self-regulation and metacognition seem also to be of particular importance for promoting knowledge restructuring and transfer of knowledge to new situations [6; 12]. The role of instruction is to provide instructional and learning events that help the learner in monitoring and controlling her or his cognitions and evaluations and thus facilitating processes of knowledge restructuring.

4. Qualitative reasoning

Qualitative reasoning is suggested to be central for processes of mental model construction and model-based reasoning [11]. A characteristic feature of qualitative reasoning is its focus on qualitative aspects of informations and structural relations inherent in a subject matter domain. It is the focussing of her or his cognitive processes on the inherent structural relations that is assumed to be relevant for mental model construction and conceptual change [11; 19].

5. Structure-oriented cognitive processing

Structured-oriented cognitive processing has been accentuated as a central cognitive variable for abstract knowledge acquisition and conceptual change [4; 19]. Problems in abstract knowledge acquisition result when students are not skilled in structure oriented processing or when they hold conceptions based on inappropriate assumptions concerning the relational structure of a subject matter domain. In order to promote conceptual change learners with misconceptions must be induced both a structure-oriented cognitive processing behavior as well as cognitions that focus on the abstraction of concepts and principles inherent in the particular subject matter domain.

6. Inductive analogical reasoning and analogical transfer

Inductive analogical reasoning is suggested to be a crucial process in schema abstraction and mental model development [13]. It is assumed that the processes involved in inductive analogical reasoning and analogical transfer may be of general importance for conceptual restructuring [13; 19]. Computer-based approaches have proved to be quite succcessful in supporting instructional attempts intended to foster inductive analogical reasoning and conceptual change in problem solving environments [10; 21; 22].

7. "Situatedness" of cognitions

As far as misconceptions on observable phenomena in the real world - e.g., the motion of objects - are concerned, these misconception seem to have developed as a result of individual situated experiences and inadequate interpretation of the causal relational structure underlying the observed phenomena [18]. It therefore seems to be psychologically reasonable that instruction, in order to promote cognitive restructuring of the inadequate cognitions should provide learners the opportunity to make alternative experiences on the

same phenomena in a quite similar environment under situated learning conditions. These experiences together with instructional support should be strong enough to induce cognitive processes that result in an invalidation of the learner's naive conceptions and in the abstraction of scientifically appropriate knowledge. Simulations have been proposed to enhance the level of "situatedness" and to induce cognitive task situations similar to those under real world conditions, thus, rendering knowledge transfer to real world problem more likely to occur [1].

8. Social embedding of cognitions

Modern cognitive psychology views human cognitions as being socially embedded in a cultural "milieu" [12]. The validity of the notion of social embeddedness of cognitions has become especially appearent with naive beliefs that are shared by many people even accross cultures and nations [20]. The social embeddedness of beliefs is supposed to be one reason why learners are resistant to adopt new conceptions. On the other hand learners in a shared responsibility social setting may provide motivational and intellectual support for thinking [2; 1]. Cooperative learning thus may promote conceptual change, in particular when learners are helped by the system, advanced peers or a teacher who provides supportive scaffolding.

9. Resolution of cognitive conflicts

The main hypothesis about the origin of principles that has been proposed from within psychology is that "new ideas arise from the resolution of cognitive conflicts" [15, p. 90). The cognitive conflict hypothesis holds that humans when experiencing a cognitive conflict - because of new informations that contradict individual knowledge and expectations - will evaluate the validity of old ideas and may adopt the new ideas when these ideas are rated to be more valuable. The cognitive conflict hypothesis, sometimes called "disequilibration thesis", is also in the centre of instructional approaches that are concerned with conceptual change [9].

10. Cognitive sensitivity and emotional willingness

According to Glaser "conceptual change is self-directed in the sense that humans are intrinsically motivated to understand the world around them and to explain events to extend their knowledge" [12, p. 32]. This view reflects aspects of cognitive sensitivity and emotional willingness that seem to be crucial for knowledge restructuring. Knowledge restructuring does not "happen" to individuals in the course of learning. Learners who are not cognitively sensitive to different views and to the instrumental appropriateness of alternative conceptions are not likely to change their own conceptions. Learners who are not cognitively prepared to recognize cognitive conflicts are not likely to change conceptions either. Cognitive sensitivity must go along with emotional willingness to accept the existance of alternative conceptions, to evaluate their appropriateness and to adopt these conceptions when they prove to be instrumental for the learner for coping more effectively with particular task situations.

In order to succeed in promoting knowledge restructuring instruction has to create a climate for conceptual change by sensitizing the learners to the existance of naive conceptions in a particular domain and by focussing on the instrumental benefits of conceptual change.

3. The structure of GELERIAT

The above mentioned cognitive principles have guided the development of GELERIAT (Generative Experimental Learning Environment for research on knowledge Restructuring, Inductive learning and Analogical Transfer). GELERIAT has the status of a prototype. It has been implemented on a Macintosh using HyperCard and PASCAL. The knowledge domain is the physical domain "superposition of motion", a subdomain of elementary mechanics. The system has been developed to promote systematic research on the relevance of cognitive and instructional conditions for processes of conceptual change.

The environment consists of 9 modules some of them being structured into submodules. So far, the following modules have been implemented:

Content Overview. The module "Content overview" provides informations on the modular structure of the environment. It serves the learner to get free access to each module in an exploratory learning mode. It may also help the learner to construct an appropriate mental representation and to improve cognitive orientation concerning the structure of instructional and learning events.

Introduction. In the module "Introduction" the learner, among others, is reminded of problems many people have in explaining everyday natural phenomena. The module is intended to enhance cognitive sensitivity and emotional willingness for conceptual change by emphasizing that naive conceptions are quite typical for many novices in explaining natural phenomena. It is assumed that the module may also contribute to situate cognitions as a prerequisite for restructuring naive conceptions.

General Problem Situation. The module "General Problem Situation" introduces the particular subject matter domain by describing the general problem situation. The description reads as follows:
Imagine you are in a leisure park. In the park there is a big highlight: a huge conveyor belt. You observe the belt running with a constant velocity to the right side and a person stepping onto the belt at a particular position. The person runs with a constant velocity either
- to the right (i.e. in the direction of the conveyor belt's motion)
- to the left (i.e. in the opposite direction of the conveyor belt's motion) or
- perpendicular to the direction of the belts motion.
The task is to predict the position of the person on the belt after the belt has moved a particular distance.

The module "General Problem Situation" is assumed to situate cognitions and to foster the development of a general situational model as a basis for solving problems on "superposition of motion".

Learning Objectives. The module "Learning objectives" serves as a shell. It may be used to present to the learner different learning objectives. In order to promote conceptual change the learning objectives have to focus on deep level cognitive processing concerning abstract knowledge acquisition, and strucuture-oriented processing. In an open learning mode the learner may choose between the objectives according to the individual learning interests.

Problem Solving. The module "Problem Solving" is a kernal module of the learning environment. It is basically a structured analogical problem solving environment to support mental model progression by inducing structure-oriented cognitive processing, qualitative reasoning as well as inductive analogical reasoning and intra-domain analogical transfer. Two types of simulations are providing different types of dynamic feedback: The simulations either highlight the phenomenon of "superposition of motion" as it may be observed in natural situations or highlight the inherent structural relations of the knowledge domain (see module "Simulations" for more informations).

The module consists of two sub-modules: "Problems" and "Diagnosis". The submodule *"Problems"* provides a sequence of 16 problems on superposition of motion with a focus on composition of motion components. The problems are presented in a pre-structured sequence. They vary with respect to surface features, e.g. direction of the person's movement relative to the movement of the belt, they are, however, identical with respect to structural features in order to facilitate inductive reasoning, schema abstraction and analogical intra-domain transfer.

In order to predict correctly the exact position of the person the learner has to know the schema for problem solving (i.e. how to compose the velocities of the two individual motions). The learner should also know that motions may be described differently with respect to different reference systems (concept of "Relativity"), and that it is important to have regard to the reference system of a particular motion. She or he should further know that motions superpose without influencing each other (concept of "Independence") and that "superposition of motion" is a reference system phenomenon.

In the submodule *"Error Diagnosis"* a simple form of cognitive modelling oriented toward the "enumeration of bugs"-approach is implemented. The module provides cognitive interpretations of some issues of erroneous individual problem solutions and gives adaptive feedback on individual bugs and misconceptions possibly relevant for the incorrect problem solutions. The module is intended to support self-regulation and metacognition, to induce cognitive conflict and to provide conceptual scaffolding for cognitive processes of conceptual change. It is implemented as a shell. It presently accounts for some procedural bugs and the most typical misconceptions in the knowledge domain. It may, however, be extended when more knowledge about possible bugs and misconceptions as well as behavioral indicators is available.

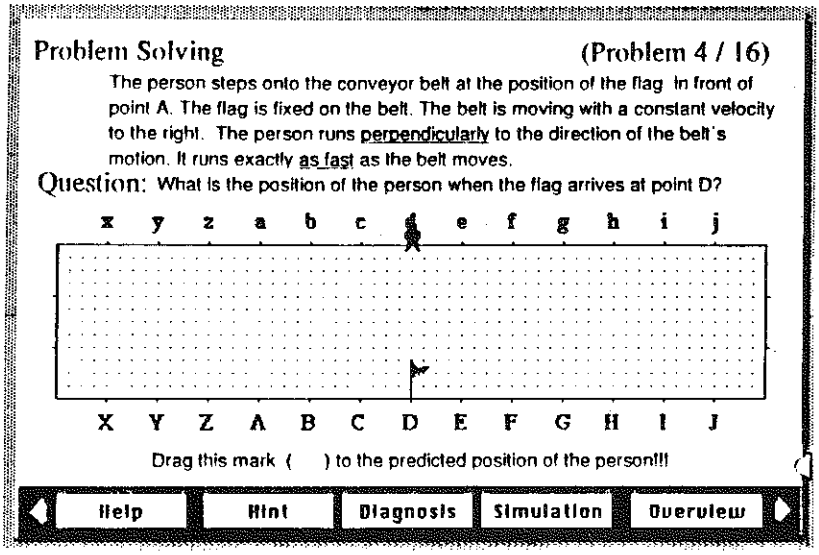
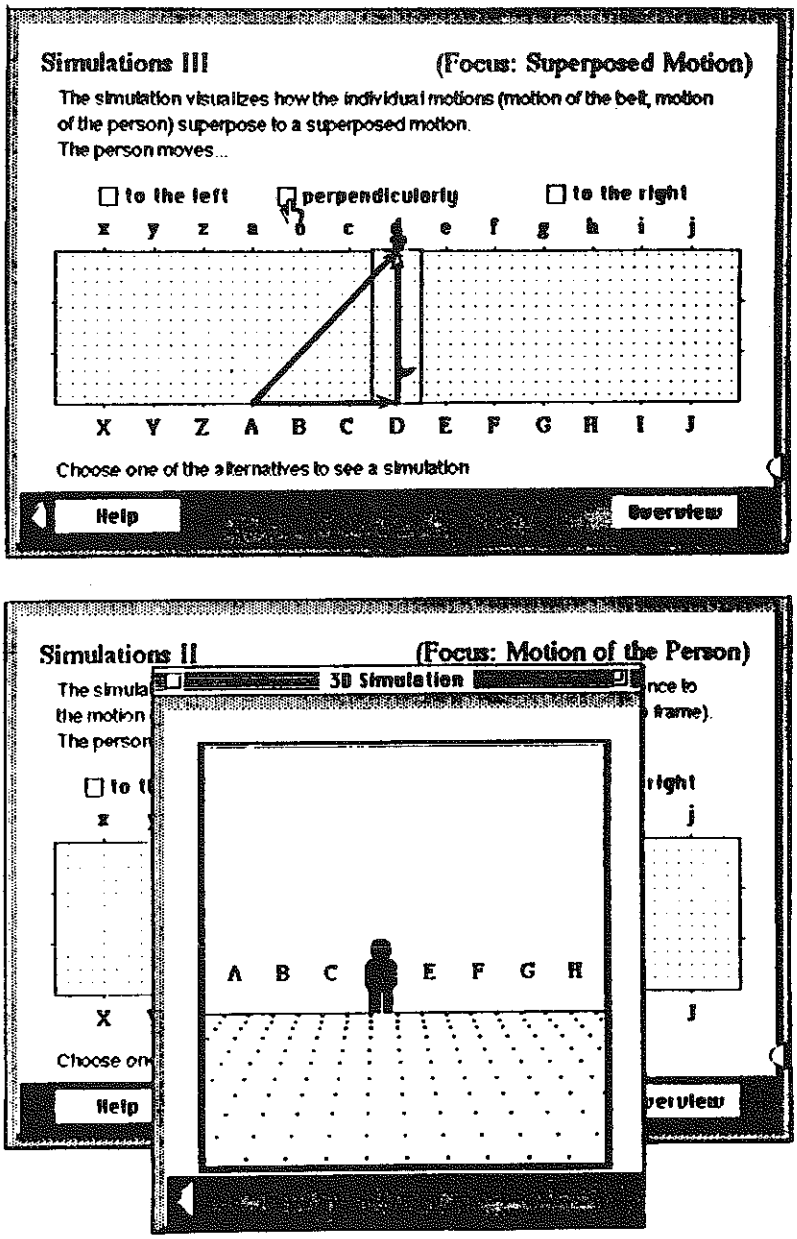


Figure 1: Screen dump showing the problem statement, the conveyor belt (the rectangle dotted area), the response of the learner (marked with "X"), and the positions of the person and the flag at the end of the simulation (original text has been translated into English) [17].

Playground for Experiments. The module "Playground for Experiments" is a simulation-based interactive micro-environment for inquiry learning and problem solving. It is intended to support constructive cognitive processing, inductive analogical reasoning and analogical intra-domain transfer. Up to 81 different experiments may be performed. The module may either be used for self-regulatory exploratory learning or for drill and practice.

Simulations. The module "Simulations" is intended to focus on structure oriented cognitive processing and to provide for some aspects of situated learning [5]. It may also serve to provide for multiple dynamic representations, thus, fostering the construction of physically adequate mental models of the subject matter domain. The module consists of three submodules. Each of the submodules is highlighting different aspects of the structural relations inherent in the domain. The focus of highlighting is on the reference-system dependency of descriptions of individual motions and on how the individual motions of the conveyor belt and the person superpose to a superposed motion.



Figures 2 and 3: Screen dumps showing different highlightings concerning the reference system dependency of descriptions of motions [17].

Comprehension control. The module "Comprehension Control" comprises forced-choice and multiple-choice questions concerning the knowledge domain "superposition of motion". Feedback or no feedback of correctness of response may be provided either automatically or at request of the learner. The questions may be used by the teacher or the experimenter in a pretest or posttest for the assessment of qualitative aspects of individual domain specific knowledge. They may also be used by the students to support learning by reflection and metacognitive control of individual progress in knowledge acquisition.

Tutoring. The "Tutoring Module" is intended to provide additional instruction. The module comprises two submodules. The submodule *"The Physical Principles"* provides verbal informations about the basic physical concepts ("Relativity" and "Independence" of motion). The submodule *"Problem Solving*

Procedures" provides instructions on the problem solving schema. The focus of the instructions is on the conceptual rationale underlying the formal algorithm of vector addition, not on the formal algorithm itself.

4. The generative character of GELERIAT

The above mentioned modules provide a variety of instructional and learning events that may be of relevance for fostering processes of knowledge restructuring. The instructional and learning events are potential candidates for constituating a particular learning environment that has still to be generated. GELERIAT in this sense is a generative learning environment. It may serve multiple functions: It is a didactical design tool in the hands of the instructional designer, a cognitive tool in the hands of the learner and a research tool for cognitive and educational scientists.

The generation of a particular learning environment is facilitated by use of a "Design Control Card". The *Design-Control-Card* provides facilities for automatic and semi-automatic adjustments of the learning environment. It may help the designer in generating a particular learning environment that meets her or his individual pedagogical and didactical intensions. It may help the learner to choose among different learning opportunities, instructional events and learning modes. It may also be used by researchers in designing experimental treatments to study the effects of different cognitive design principles on conceptual change.

REFERENCES

[1] Brown, J.S., Collins, A. & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18, 1, 32-42.

[2] Brown, A.L. & Palincsar, A.S. (1989). Guided, cooperative learning and individual knowledge acquisition. In L.B. Resnick (Ed.), Knowing, learning, and instruction. Essays in honor of Robert Glaser (pp. 393-451). Hillsdale, NJ: Lawrence Erlbaum Associates.

[3] Caravita, S. & HalldÃn, O. (in press). Re-framing the problem of conceptual change. In S. Vosniadou (Ed.), Special issue of the journal Learning and Instruction.

[4] Chi, M.T.H., Glaser, R. & Rees, E. (1982). Expertise in problem solving. In R. Sternberg (Ed), Advances in the psychology of human intelligence, Vol. 1 (pp. 7-70). Hillsdale, NJ: Lawrence Erlbaum Associates.

[5] Collins, A. (1991). Cognitive Apprenticeship and educational technology. In L. Idol & B.F. Jones (Eds.), Educational values and cognitive instruction: Implications for reform. Hillsdale, NJ: Lawrence Erlbaum Associates.

[6] Collins, A., Brown, J.S. & Newman, S.E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L.B. Resnick (Ed.), Knowing, learning, and instruction: Essays in honor of Robert Glaser (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum Associates. [7] De Corte, E. (1992). Embedding new information technologies in powerful learning environments. Invited address presented at the NATO Advanced Study Institute on "Psychological and educational foundations of technology-based learning environments", Crete, Greece, July 26 - August 3.

[8] Duit, R. (in press). The constructivist view. A both fashionable and fruitful paradigm for science education research and practice. In L. Steffe (Ed.), Constructivism in education. Hillsdale: Lawrence Erlbaum Associates.

[9] Dykstra, D. (1992). Studying conceptual change: constructing new understandings. In R. Duit, F. Goldberg, & H. Niedderer (Eds.), Research in physics learning: theoretical issues and empirical studies (pp. 40-58). Proceedings of an international workshop held at the University of Bremen, March 4-8, IPN 131.

[10] Frederiksen J. R. & White B. Y. (1991). Mental models and understanding: a problem for science education. In E. Scanlon & T. O'Shea (Eds.), New directions in educational technology. NATO ASI Series. Series F: Computer and System Sciences, Vol. 96 (pp. 211-226). Berlin: Springer, 1991. [11] Gentner, D. & Stevens, A. (1983). Mental models. Hillsdale, NJ: Lawrence Erlbaum Associates. [12] Glaser, R. (1991). The maturing of the relationship between the science of learning and cognition and educational practice. Learning and Instruction, 1, 129-144.

[13] Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986). Induction. Processes of inference, learning and discovery. Cambridge, MA: The MIT-Press.

[14] Johnson-Laird, P.N. & Byrne, R.M.J. (1991). Deduction. Hillsdale, NJ: Lawrence Erlbaum Associates.

[15] Ohlsson, S. (1991) Towards Intelligent Tutoring Systems that teach knowledge rather than skills: five research questions. In E. Scanlon & T. O'Shea (Eds.), New directions in educational technology (pp. 71 - 96). NATO ASI Series F, Vol. 96. Berlin: Springer.

[16] Pfund, H. & Duit, R. (1991). Bibliography: students' alternative frameworks and science education (3rd edition). Kiel: IPN. [

17] Tergan, S.-O. & Oestermeier (in press). Promoting studies on conceptual change. To appear in S. Vosniadou, E. De Corte, B. Glaser & H. Mandl (Eds.), International perspectives on the psychological foundations of technology-based learning environments. NATO Advanced Study Institute Series.

[18] Tiberghien, A. (1990). Learning and teaching at middle school level of concepts and phenomena in physics: the case of temperature. In E. De Corte, H. Lodewijks, R. Parmentier & P. Span (Eds.), Learning and Instruction: European Research in an International Context, Vol. 2.2 (pp. 631-648). Oxford: Leuven University Press and Pergamon Press.

[19] Vosniadou, S. & Ortony, A. (1989). Similarity and analogical reasoning: a synthesis. In S. Vosniadou & A. Ortony (Eds.), Similarity and analogical reasoning (pp. 1-17). Cambridge, MA: Cambridge University Press.

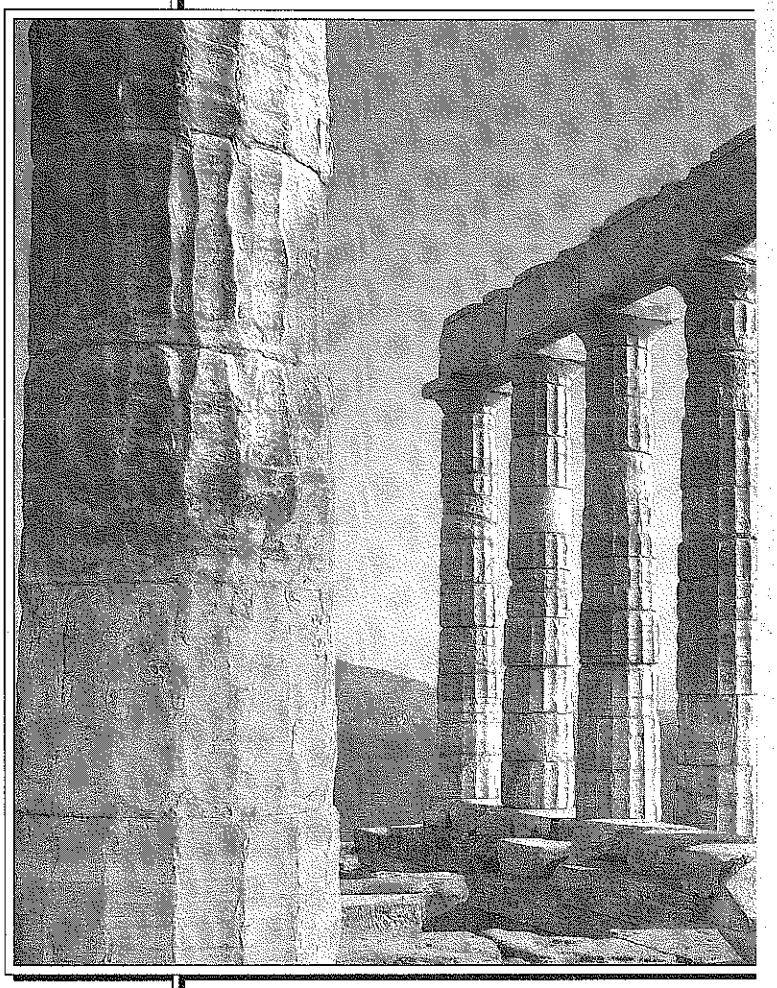
[20] Vosniadou, S. & Brewer, W.F. (1989). A cross cultural investigation of children's conceptions about the earth, the sun, and the moon: Greek and American data. In H. Mandl, E. DeCorte, N. Bennett, & H.F. Friedrich (Eds.), Learning and instruction: European research in an international context, Vol. 2.2 (pp. 605-629). Oxford: Pergamon.

[21] White, B.Y. (1991). A microworld-based approach to science education. In E. Scanlon & T. O'Shea (Eds.), New directions in educational technology. NATO ASI Series. Series F: Computer and System Sciences, Vol. 96 (pp. 227-242). Berlin: Springer, 1991.

[22] White, B.Y. & Frederiksen, J.R. (1990). Causal model progression as a foundation for intelligent learning environments. In W. Clancey & E. Soloway (Eds.), Artificial Intelligence and learning environments (pp. 99-158). Cambridge, MA: MIT Press.

Theme E

Programming Environments



E. Programming Environments

259 • *Calani Baranauskas M. C.*
A Constructionist Approach to the Design of a Logic Programming Environment ...

269 • *Gow J.*
Novice Knowledge Engineering with a Logo-Based Expert System

275 • *Krivy I., Weinberger J.*
Project Management Analyser in Logos

281 • *Sendova E., Sendov B.*
Learning to Speak Mathematically and Speaking Mathematically to Learn ...

289 • *Sklenar J., Sklenarova K.*
Network Paradigm Modelling in the Logos Language

297 • *Suzuki H., Kato H.*
AlgoBlock: a Tangible Programming Language - a Tool for Collaborative Learning

305 • *Triantafillou E.*
Preschoolers Logo World

311 • *Zupanic D.*
Parallel Programming in Logo

A Constructionist Approach to the Design of a Logic Programming Environment for Novices

Maria Cecilia Calani Baranauskas

Computer Science Department DCC
& Nucleus of Informatics Applied to Education NIED
State University of Campinas Unicamp

Caixa Postal 6065
13081 970 Campinas S.P. Brazil
Fax: 55 192 397470
e-mail: cecilia@dcc.unicamp.ansp.br

Abstract

The constructionist approach to the design of computer based learning environments has been applied to research based on Logo, within the "Logo culture". The work reported here represents an effort in extending some important ideas from the constructionist theory to the domain of other programming languages. Based on these ideas we are proposing some tools to constitute a Prolog programming environment for novices. The design of the tools presupposes a learning environment in which the knowledge is built by the subject, itself, by means of her/his interaction with objects of the computational media. The purpose of the tool package is to make explicit the underlying programming paradigm, enriching the type of feedback provided by the Prolog environment. A graphical formalism is used to represent the knowledge involved in Prolog programming in its operational and declarative semantic, as well.

Keywords: Constructivism, semantics, logic programming, Logo, Prolog

A Constructionist Approach to the Design of a Logic Programming Environment for Novices

1. Introduction - The motivators to the work

The great contribution of Logo as a programming language for novices is its subset named Geometric-Logo or Turtle Graphics. The manipulation of the graphical object named "turtle", capable of walking on the screen tracing its path, is the main characteristic for which the language power is well known.

Teaching the turtle to do something (a square or a house picture, for example) is a metaphor to the activity of programming in turtle's context. The program the child develops to "teach" the turtle is based on a "procedural" model of programming. Representing the solution to a problem, in this context, involves knowing what the turtle should do to produce a picture on the screen and "how" to instruct it to achieve the desired result (using a step by step approach, a repetitive command, etc.). The turtle (and indirectly the computer) is considered an "object" that obeys orders. Therefore the model of "how" the machine (computer) "works" is represented by the turtle's role in the environment.

The results (turtle traces on the screen) produced in this model of programming give the user a feedback which can take the novice to reformulate the procedure "taught". The beauty of this idea lies in the fact that during the process of teaching the turtle, the child can think about his/her own learning process and being aware of that, becomes an "epistemologist" (Papert, 1980). In my view, apart from the theoretical reasons which made Logo a successful programming language for unsophisticated users, the power of turtle's programming in novices' context arises from the graphical feedback provided, which feeds the cycle in which the novice is engaged during the programming activity.

There is also another programming paradigm Logo arises, enabling users to work with "lists". List processing is an inheritance from the Lisp language. Actually, the first version of Logo was named "Baby Lisp" and it did not have the turtle and its graphical primitives.

Thus, despite being considered "procedural" (Johanson, 1988; Mendelsohn, 1990), by working with Logo, the user can also flavor the functional paradigm, by defining functions for list processing. This multiplicity of programming paradigms makes Logo a very flexible and friendly tool to the user who wants to flavor the different paradigms in a similar programming environment. On the other hand, this variety of models for problem solution representation causes difficulties when the novice wants to go beyond the frontiers of the procedural style to work with list processing (Baranauskas, 1991a). According to my view, the difficulties exist because the novice constructs a conceptual model of the language based on the turtle's actions, which reflect the procedural paradigm. When the model of programming changes (to list processing, for example,) the user hardly becomes aware of this change. Furthermore, (and more important) there is not a feedback reflecting the underlying entities of the new paradigm (the mechanics of composing and evaluating a function, for example,) as there is for the procedural paradigm, by means of turtle graphics.

Prolog is another programming language which has attracted attention within the educational context. It is the programming language which best exemplifies the logic programming paradigm. According to the model of programming proposed by Prolog, the meaning of a "program" is not given anymore by a succession of elementary operations which the computer is supposed to perform, but by a knowledge base in a certain domain and questions asked to this knowledge base. There is an "inference machine" (invisible to the programmer), responsible for finding answers to the questions. The beauty of this idea lies in the fact that programmers state only the relevant facts and rules about the problem domain, and they do not have to prescribe the actions the machine should make in order to achieve the solution.

Nevertheless, Prolog programming has its drawbacks. Several empirical analyses of the difficulties novices

have with Prolog have been carried on (Sherz, 1989; Fung, 1990; Hook, 1990; Taylor, 1990). Studies in literature argue that novice Prolog programmers need a complete and detailed execution model of Prolog (Rajan, 1990; Van Someren, 1990). Tools that give the student access to the details about the notional machine behavior are considered essential and even more important for learning Prolog than for learning other languages (Van Someren, 1980).

My proposal to cope with novices' difficulties in the Prolog environment is based on the idea of empowering the type of feedback provided by the programming environment, when a question is answered by the machine. This represents the "visibility" metaphor we have in Turtle Graphics transposed to Prolog's context. Thus, instead of having only a "yes" / "no" or a value associated to a variable as a response to a question, the novice can investigate, by means of a graphical formalism, "why" and "how" the inference machine reached the response.

2.The constructionist environment

In the proposed scenario the novice interacts within an environment composed by the Prolog interpreter added by the designed tools. There is the presence (not necessarily during all the time) of an "experienced" person who knows about this environment and may facilitate the novices' interaction.

The theoretical basis to this learning environment has roots in Piaget's constructivism (Piaget, 1977), Papert's constructionism (Papert, 1986) and Vygostky's theory about the influence of the language in the learning process (Vygotsky, 1962). According to those theories, the knowledge, rather than transmitted from a more informed person or a computer system to the learner, is constructed by the subject (the learner) by means of her/his own actions in the learning environment. As in Papert's constructionism, this work proposes objects from the computational media (the tools) to enrich the environment where the subject acts (the Prolog environment). Vygotsky's influence appears in my concern to the programming language paradigm mediating and influencing the process of novice-environment communication, and in the presence of the experienced person in the scenario.

Programming as a cyclic activity: the importance of the feedback

My view of the programming activity, which underlies the proposed tools and its use by novices is a cyclic process. During this process, the subject builds a database (program) from previous hypotheses (about the problem domain, about the environment, about the language) and questions it. As soon as the database has been executed by the machine (which tries to answer the questions) it is possible to have a feedback which leads the novice to a reflection and reformulation of her/his set of hypotheses, continuing the programming cycle. The importance of the feedback in the general process of acquiring knowledge is explained by Ackermann (1987):

> "... Learning occurs, moreover, through probing and confronting one's current theories in one way or another. One could say that growth requires feedback, and feedback requires engagement and action. Thus both action and evaluation of the effects-of-an-action are essential to learning." (Ackermann, 1987, p.1)

Adopting this approach, debugging a program is not treated as an activity of search and elimination of errors, in the sense of being a "burden" that we want to get rid of; it becomes a process during which knowledge is built. Such knowledge refers not only to the concerned domain (the problem being solved), but also to the interlocutor (programming environment) and, in a meta cognitive level, it refers to the knowledge about the process of "knowing". Therefore, the novice's programming activity is an incremental process which involves the cycle fed by responses provided by the environment. According to this approach, the programming environment must provide the user with feedback which enables her/him to revise her/his set of hypotheses and to continue the programming cycle. The main goal of the proposed tools is to enrich the environment responses, based on Prolog's operational and declarative semantics, considering the nature of the feedback.

3. The designed tools for the Prolog environment

According to the aspects discussed earlier, the programming environment must favour the engagement of the novice with the elements of the paradigm raised by the language. Considering Prolog's context, this means making accessible to the novice both the logical and the operational semantics of the language.

The basic aspects which oriented the design of the tools were the visibility metaphor, the multiplicity of representation and the use of a graphical formalism to represent both: the program and the inference machine in action. Visibility does not mean only showing the inference machine in action, but also showing the knowledge expressed by the program regardless of its use by the machine. The multiplicity of representations aims to provide the novice with alternative ways to look at the problem. The graphical formalism in problem solving has already shown its usefulness in several domains and, particularly in the computational context, as discussed before, Logo is a good example of the effectiveness of using visual formalisms in the programming activity.

The set of tools constitutes a Declarative Module and an Operational Module (Baranauskas, 1991b). The Declarative Module deals with a graphical representation of the knowledge expressed in the database, in a way that is not dependent on how this database is understood by the inference machine. The approach used to cope with the operational aspects involved in Prolog programming is based on showing a graphical representation of the inference machine in action during the process of looking for the answer to a question. The model of representation used by the Declarative Module is based on semantic network diagrams. The model of representation used by the Operational Module is based on the construction of a proof tree for a goal.

Creating a Database

In order to create a database the subject uses a simplified editor named "Semantic Editor". By means of this tool, s/he has access to the semantic diagram representation of her/his database at clausal level. Through it the novice may have a feedback on the "meaning" of her/his sentence, based on the concepts of "objects" and "relations". This tool is supposed to help the novice when s/he is creating her/his database, by making explicit the interrelationships between the objects of a proposition. Figure 1 shows an example of the visual representation for the sentence *"john likes X if X likes wine and X is a woman"*, which is represented in the clausal form by *"likes(john,X):-likes(X,wine), woman(X)."*

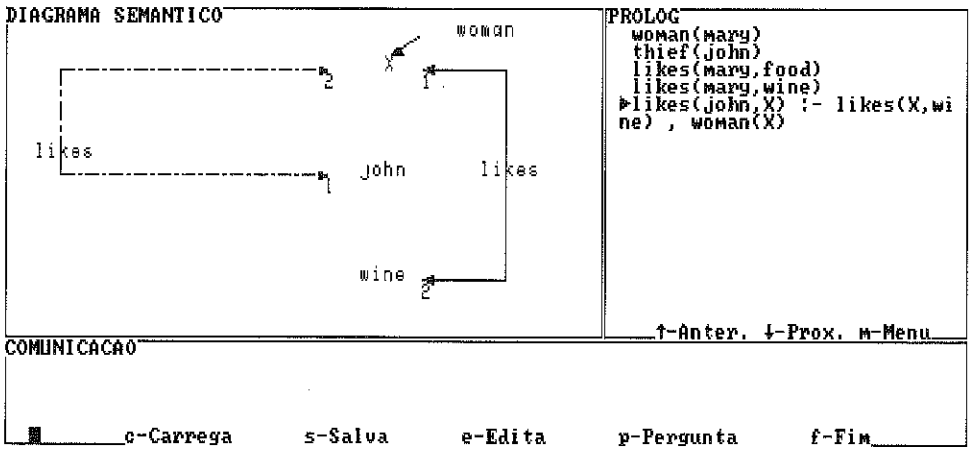


Figure 1. A Semantic Editor screen, with 3 regions: one for the database written in Prolog' syntax (top, right), one for the visual representation of the selected clause (top, left) and one for i/o communication (botton).

Investigating a Database

At the database level, the system provides a semantic diagram representation of the clauses which constitute the user's database. At this level, the model of representation makes explicit the inter-relations between clauses which refer to the same objects. With this tool, the user can locate objects or relations s/he wants to investigate. Without depending on the execution process, associations which are implicit in the database are made explicit by means of the diagram representation. As illustrated in figure 2a, s/he can observe that there is a "path" connecting the objects *john* and *mary*, for example.

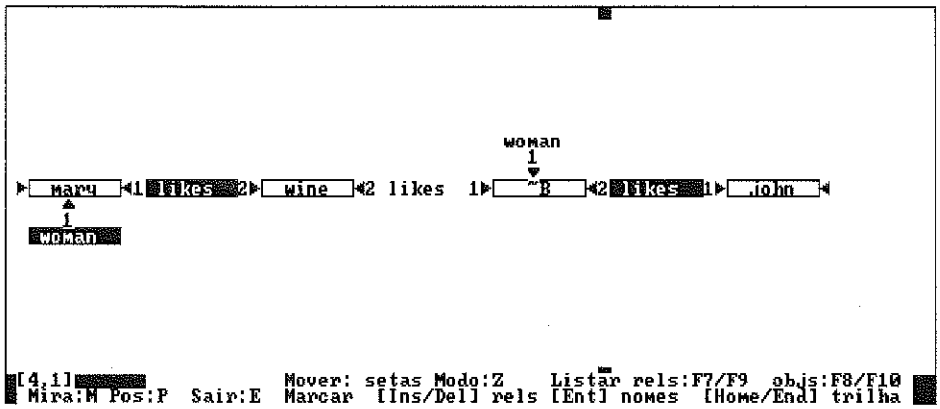


Figure 2a. Semantic Diagram representation for part of the database

The relations are represented in reverse video (black in this picture), unless they represent the antecedent part of a rule. The numbers indicate the order of the objects in the relation. Another window can be used to list relations or objects present in the database, as illustrated in figure 2b.

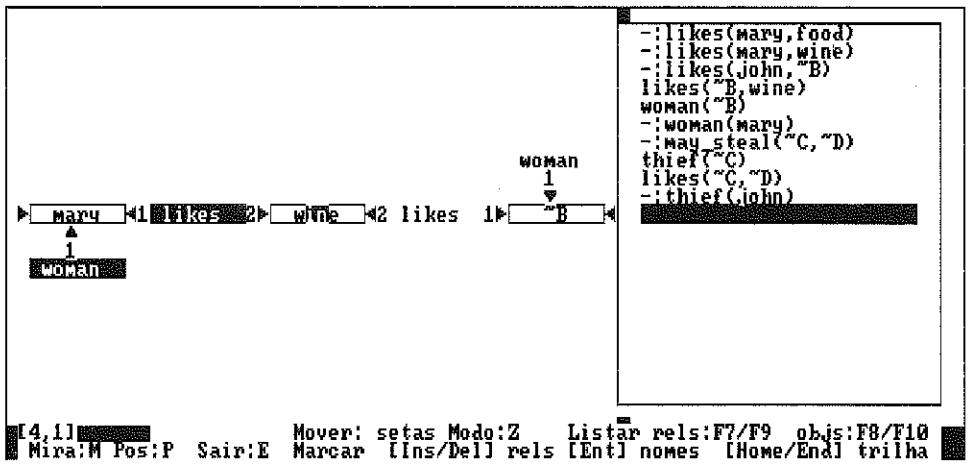


Figure 2b. Semantic Diagram representation with the listing of the database relations

Through the window which lists relations and/or objects, the user can locate, mark and select parts of the diagram to be shown. The diagram can be reduced, zoomed, moved for visualization of the whole or of parts in the representation.

Investigating Inferences in a Database

In order to understand "why" the response to a question is a specific one, the subject can proceed with a systematic examination of the execution flow or s/he can make a meta-analysis of the program (Hook, 1990).

Meta-analysis

The semantic diagram representation enables the subject to make a meta-analysis of the database by showing her/him the relations between the elements of the database (or relations that can be inferred from the database), without refering to the knowledge of the execution mechanisms of Prolog.

At the inference level, the tool shows the semantic diagram representation of the sentences inferred during Prolog's response to a user's question. Thus, the user can visualize facts implied by her/his database, as for example, *"john likes mary"*, which combined to other facts in the database lead to the response. Figure 3 shows the diagram representing the solution to *"may-steal(X,Y)."*

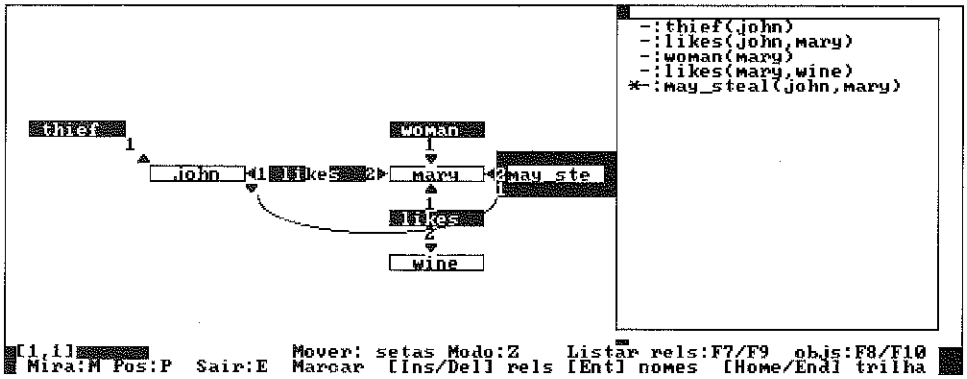


Figure 3 Semantic Diagram representation for the database at inference level.

Control flow analysis

There are some situations in which the meta-analysis is not sufficient for the novice's understanding of why the machine answered the way it did (situations in which the order of the sub-goals in a rule changes a result, are an example). To cope with these situations, the tools in the Operational Module enable the subject to make a systematic analysis of her/his program execution flow. The aim of the tools in this module is to help the user to construct the conceptual model of Prolog's inference machine. Our approach involves showing the execution process by means of a step by step construction of the search tree for a goal. Figures 4a, 4b, 4c and 4d show a sequence of the search tree construction for the goal *"may-steal(A,B)"*, considering the same database of the previous examples:

```
thief(john).
likes(mary,food).
likes(mary,wine).
likes(john,B):-likes(B,wine),woman(B).
woman(mary).
may_steal(A,B):-thief(A), likes(A,B).
```

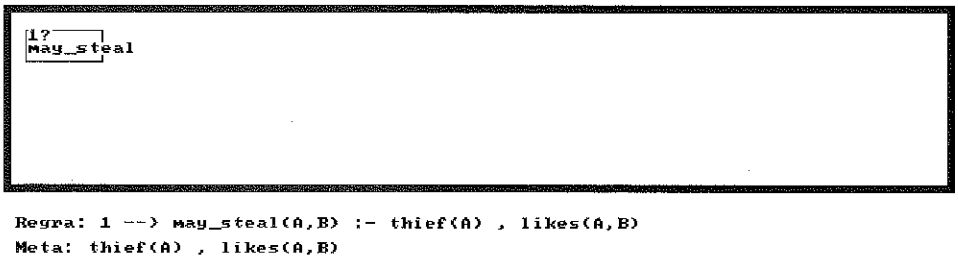
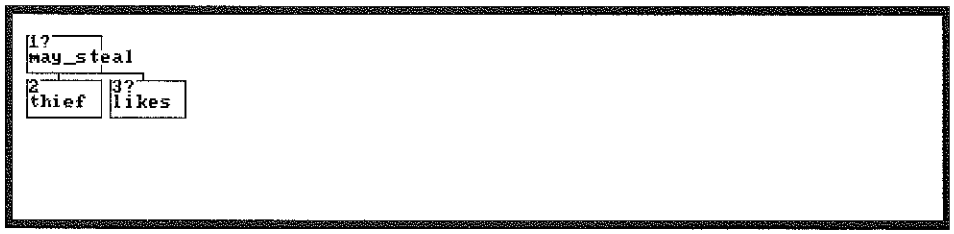


Figure 4a. Rule 1 is being applied to the first moment in the process of solving the goal




```
Regra: 1 --> may_steal(john,B) :- thief(john) , likes(john,B)
Meta: likes(john,B)
```

Figure 4b. Moment during the process when *likes(john,B)* is the new goal (node 3)




```
Regra: 3 --> likes(john,B) :- likes(B,wine) , woman(B)
Meta: likes(B,wine)
```

Figure 4c. Moment during the process when rule 3 is being applied to the goal *"likes(john,B)"*




```
Regra: 1 --> woman(mary) :- true
Meta: true
```

Figure 4d. Final moment in the process when goal and sub-goals are solved, and there is a response (*may_steal(john,mary)*).

Working with this tool, the user has access to elements necessary to the understanding of "how" Prolog's machine is using her/his database to find answers to her/his questions. S/he can observe the structure of the execution process, the matching of goals in the database, the instantiation of variables, goals which failed, goals which succeeded and the proof tree.

4. Discussion and conclusion

The creation of computational tools to constitute a programming environment for novices is centered in two basic aspects: first, the subject for whom the tool is designed (the novice) suggests the programming

environment should be a "learning environment". In this sense, programming means more than the representation of a well known solution to a problem, in a language. Knowledge of the problem's domain and knowledge of the language used to represent the solution to the problem are both being constructed during the interaction of the subject within the environment. Second, a programming environment should reflect to the user the underlying programming language paradigm. In Prolog's case this implies dealing not only with the operational aspects of the language, but also with its declarative aspects.

Some significant tools were developed to deal with the operational aspects of Prolog, proposing models of how the interpreter obtains a response to a certain goal (Rajan, 1986; Dewar, 1986; Plummer, 1988; Eisenstadt, 1988). Despite being considered essential, these tools deal with only the operational aspects of the language and are used to assist the student in conventional Prolog instruction settings.

In my proposal of a constructionist learning environment, the novices interact strictly within the designed environment. As a result, the programming environment plays an important role since the very beginning of the process. Such proposal encompasses the following aspects: it presupposes that the knowledge acquisition takes place during the subject's action in the environment and that this novice-environment dialog is mediated by the paradigm implicit in the language. The underlying idea is that the learner is the agent in his process of learning, and the concepts assimilation takes place according to her/his mental schemes, instead of instructions or "intelligent" interventions which are elaborated *a priori*.

An observational study of different types of novices interacting with the proposed tools in a constructionist scenario was undertaken (Baranauskas, 1993). Some issues on their process of language acquisition will be reported in a forthcoming paper.

REFERENCES

Ackermann, E. K.(1987). Pathways into a child's mind: helping children become epistemologists. In P. Helthe & L. Marquardt, (Eds.) Symposium Proceedings Science Learning in the Informal Setting, Chicago Ac. of Science, Chicago, 1977 p. 7-19.

Baranauskas, M. C. C. (1991a). Procedure, Function, Object or Logic? Proceedings of the Eight International Conference on Technology and Education, Toronto, Ontario, pp. 730-731.

Baranauskas, M. C. C. (1991b). Creating a computer-based learning environment to help novices to develop Prolog programs. Proceedings of the Sixth International PEG Conference. Rapallo, Italy, pp. 92-98.

Baranauskas, M. C. C. (1993). Criacao de ferramentas para o ambiente Prolog e o acesso de novatos ao paradigma da programacao em logica. PHD Thesis, Electrical Engineering Faculty, State University of Campinas.

Dewar, A. D. e Cleary, J. G. (1986). Graphical display of complex information within a Prolog debugger. International Journal of Man Machine Studies, 25, 503-511.

Eisenstadt, M. and Brayshaw, M. (1988). The Transparent Prolog Machine (TPM): an execution model and graphical debugger for logic programming. *The Journal of Logic Programming*, 5, 277-342.

Fung, P., Brayshaw, M., Du Boulay, B. e Elsom-Cook, M. (1990). Towards a taxonomy of novices' misconceptions of the Prolog interpreter. *Instructional Science*, 19:311-336.

Hook, K., Taylor, J. and Du Boulay, B. (1990). Redo "try once and pass": the influence of complexity and graphical notation on novices' understanding of Prolog. Instructional Science, 19:337-360.

Johanson, R. P.(1988). Computers, Cognition and Curriculum: Retrospect and Prospect. *Journal of Educational Computing Research*, 4(1), 1-30.

Mendelsohn, P., Green, T.R.G., Brna, P. (1990). Programming Languages in Education. Document TECFA 90-8, Universite de Geneve.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York: Basic Books.

Papert, S. (1986). *Constructionism: A new opportunity for elementary science education.* (A proposal to

the National Science Foundation). Cambridge, MA: MIT Technology Laboratory.

Piaget, J. (1977). A teoria de Piaget. In P. Mussen and S. Pfromm Netto (Eds.). *Manual de Psicologia da Crianca*, (translated to portuguese from Carmichael's Manual of Child Psychology), S.P.:EDUSP, pp. 72-117.

Plummer, D. (1988). Coda: an extended debugger for Prolog. In D. Plummer (Ed.) Logic Programming: Proceeding of the Fifth International Conference and Symposium, pp 496-511, Cambridge, MA: MIT Press.

Rajan,T. (1986). APT: A principled design for an animated view of program execution for novice programmers. Technical report 19, Human Cognition Research Laboratory, The Open University.

Scherz, Z.,Goldberg, D. and Fund, Z. (1989). Cognitive Implications of Prolog Learning. Mistakes and Misconceptions, *Journal of Educational Computing Research.*

Taylor, J. (1990). Analysing novices analysing Prolog: what stories do novices tell themselves about Prolog? *Instructional Science*, 19:283-309.

Van Someren, M. W. (1990). What's wrong? Understanding beginners' problems with Prolog. Instructional Science, 19:257-282.

Vygotsky, L. S.(1962). *Thought and language*. Cambridge, MA: MIT Press.

Novice Knowledge Engineering with a Logo Based Expert System

James Gow

Nova University
P.O. Box 700585
Miami, Florida 33170-585
e-mail: gowj@novavax.nova.edu

Abstract

Students were taught fractions in the conventional manner and for a class project designed knowledge bases with an expert system written in Logo. Students with stanines one through three are selected from a public middle school for participation in a knowledge base design project in fractions. Knowledge bases were based on Perkins, design criteria for knowledge and Harels, software design for learning project. An attempt to apply statistical evaluation to artificial intelligence tools in the public middle school classroom was mitigated with interpretive and observational methodology to resolve conflicts in results. Case study methods were utilized to interpret behavior of students participating in the project the depth and time logs are reviewed to help identify changes in student behavior.

Keywords: expert systems, novice knowledge engineering, knowledge bases

James Gow *is currently an eighth grade basic math teacher win Dade County Public Schools, Miami, Florida. He has been teaching math and computer education for eighteen years in private and public schools. James is a doctoral candidate at Nova University in Fort Lauderdale in the center for computer information sciences (CCIS), computer education department. James coordinated the recent establishment of comp.lang.logo on Usenet and did a poster presentation at the Emerging Technologies in Education Conference in Moscow sponsored by American Association for the Advancement of Computers in Education.*

Novice Knowledge Engineering with a Logo Based Expert System

Purpose

In depth studies are 'limited' Lippert [10]; Murphy [11], and Bigum [1] and little data has been gathered regarding the cognitive benefits of constructing KB's. Due to expense and hardware requirements the transition has been slow for classroom use. Up to this point expert systems have been used with pre-existing KB's for decision making purposes and the idea of constructing KB's as classroom activities has been seldom tried. More data needs to be gathered to document the usefulness of constructing knowledge bases in the classroom with public middle school students.

Because of some design changes to an existing backward chaining expert system shell developed by Murphy [11], it is possible to execute Logo programs from the knowledge base. This shell, because it is written in Logo, has made expert system shells more attractive to children bringing the focus of Logo research into the artificial intelligence and novice knowledge engineering (NKE) arena.

Data Source

One basic math teacher's classes, seventh (one class), eighth (one class), ninth (three classes) were targeted for implementing the knowledge base design (KBD) project.

Subject selection was based on enrollment in the basic math classes in a local public middle school class, Miami, Florida. Acceptance into these classes was based on stanines one through four and the average of three components: applications, concepts and computation. Students were assigned at random to the classes based on available seats after subject selections were given consideration. The majority of stanines were between one and three. The students were selected after leveling for one semester. Students of above average ability for this level were moved to higher classes. No significant differences were found between classes based on analysis of SAT test scores before the implementation.

Methodology

In this study, the teaching model used for constructing knowledge bases is based on Idit Harel's [7] work at the Branagin school in Boston where she did her dissertation implementation on software design for learning. Harel's work is based on Perkins, [13] illumination of four questions that guide design; 1) What is the purpose of the design? 2) What is the design's structure? 3) What are model cases of the design? 4) What are arguments that explain and evaluate the design? In the construction of knowledge bases the four design questions were the guiding rule in KB design in this application of artificial intelligence in the classroom.

Students were told they were going to design a knowledge base that would teach fractions to classmates in seventh grade. Students were given a sample knowledge base to implement before they started designing their own. Students learned Logo as they worked on the example KB's including Logo programs in the **if** or **and** part of the rule. Students participated in discussions of different programming examples on the chalk board as KB's required Logo programs to depict knowledge.

SAT and post-test scores based on MacMillian constructed chapter tests on fractions were subjected to t-tests and one-way ANOVA's with SPSSX online at NOVA operating under UNIX. A ten question

questionnaire was answered by all students in the control and experimental classes at the end of the semester. The results were reviewed for attitudinal changes and preferences.

A time log was kept of the number of times students went to the computer and how much time they spent there. The original design of the time log was to provide equal time for all students. In the end it became a way to measure how much time each student took when he was not interested in participating to insure his/her involvement in the project. This became important near the end of the semester when exams were coming up and students had less time for the KBD project. Only those students who wanted to go to the computer went. The activity was used as a reward for finishing review work since that had to be finished first.

Limitations

In order to deal with hardware limitations, students took turns rotating through the computer station. It was found that the one-computer classroom could achieve some of the same results as the multi-computer classroom employed in Harel's work (Gow, [3]). According to Greeno [5] and the general consensus of learning, all learn the same regardless of whomever or wherever. The assumption here is that assigning students by stanine creates a homogeneous group. Time is a limitation since students were not able as in Harel to finish their design projects. Students were instructed in metacognition earlier in the year; this exposure may have had an effect on learning. Video tapes were shown and steps for managing their own learning processes were studied. Because of Guernon [6] and Eyler's [2] discovery that students who made more metacognitive references were better problem solvers, metacognition training may have affected fraction post-test scores.

Results

After analysis of the t-tests and one-way ANOVA it was discovered that despite the non significant differences before the project between any of the groups the eighth grade classes had significant differences whereas the seventh grade classes didn't. Because of the many variables that effect public school classrooms, isolating specific causes was difficult. It became apparent that the conclusions would have to be more interpretive and observational. This is consistent with Papert's [12] claim that in constructive environments attempts to use statistical rigor promotes scientism and overlooks the important actions of students seen with observational methodology. No comparisons were made for the ninth-grade classes since there was no control class possible.

On the surface contradictory data has been collected. Upon further examination of the classes, teachers, and environment, it was discovered that the eighth-grade teacher was not certified to teach math. He was a band teacher, who because of budget cuts and overpopulation, was required to teach eighth-grade basic math. The seventh-grade control teacher had been teaching in the school for 25 years and had very good control of his students. The seventh-grade experimental teacher was new to the school and was considered to have poor control over his students. The experimental teacher introduced metacognition training earlier in the year to help students improve problem solving abilities (Eyler, [2]). The experimental teacher created a constructionist environment by teaching the students how to build balsa wood bridges to demonstrate principals in geometry the area of study for the semester. No other differences were observable that might account for similarities or differences in classes .

The analysis of the questionnaires indicated students felt they knew more about fractions than they did when they started. Students also felt they learned more about computers in general due to their learning how to build the KB's and build Logo programs. Most students enjoyed their experiences and indicated they felt the experience was worthwhile.

Analysis of the time logs showed students in the first part of the semester and until the last two weeks of

the year evenly shared the time on the computer. Students who eventually lost interest were still persuaded to take their turns at the computer and work on their KB designs.

Implications

Due to the difficulty of eliminating variables to make the groups equal in the public school environment interpretations are limited but observations are salient. Since the differences between the seventh graders existed only in the perceptions of the teacher's control of student behavior, and since the results were similar, it is possible to say the KBD classes did no worse when prepared with metacognition training and constructionist environments than the EM (expository methods) classes. This suggests that any teacher who wants to implement a KBD project may do so with the assurance that, if the methodology is followed, students will do as well as other students who do not do novice knowledge engineering (NKE) combined with Logo. Looking a little further into the situation, it might be argued that since the seventh-grade experimental teacher students scored as well as the control teacher students, perceptions of a teachers' ability to control class behavior may not be a good measure of learning. The differences found in the eighth-grade classes may be due to superior methodology of KBD and not whether or not a teacher is certified in a subject. In fact, there may be other reasons the researcher is not aware of that the band teacher was selected to teach the math class. In order to substantiate this claim a closer look at student behavior must be examined.

In the Harel study a close look at student behavior was made through the case study method on a student named Debbie. Debbie was a quiet, heavy set, withdrawn student who was not doing well at school and would not participate in class activities. In observations of Debbie on video tape (supplied by MIT media center), a similar scenario developed in the seventh, eighth and ninth-grade experimental classes. In the seventh-grade class a large, heavy set girl named Corinne was at first introverted and having a great difficulty with math. She was a behavioral problem because she was always arguing with the other students and seemed to have no interest in maths. After working on the KBD project she began to make more attempts to solve problems. When it was her time to work on her KBD project, she was anxious to go to the computer. Corinne worked on her bridge project but seemed more comfortable with the computer than the bridge. She was always asking when she could go to the computer and trying to use other people's turns so she could spend more time designing her KB. From there she was the first to volunteer to go to the board even though many of the problems she worked were wrong. Her transformation was reminiscent of the Debbie of Harel's experiments. In each of the classes there were similar students who made changes after getting involved in the KBD project. This continued even in the ninth-grade class despite the lack of a control comparison. One student named Grace was a perfect example. She had a hard, combative attitude and was difficult to deal with. After getting involved in the KBD project she was the first student to finish the example KB and design her own KB on fractions. Although she never finished her project, similar to Debbie in Harel's work, she got more computer time than most of the students because she was using other students' time whenever they were not present. Her attitude softened slightly as she took pride in her accomplishments on the computer. In a time log of students at the computer time Grace had the most time.

Conclusion

If a teacher chooses to implement a KBD project they can be assured that student attitudes towards mathematics will change. This study was done in fractions but could as easily done in any other area of mathematics or any other domain (Gow, [4]). Similar to the reports Lippert [10] and Starfield [14] received, students will feel they are doing something worthwhile. Combining novice knowledge engineering (NKE) with Logo is possible and functions similar to software design. Hayes-Roth, Waterman, Lenate (1983) indicate many of the rules software designers use are similar to the ones knowledge engineers use in designing their knowledge bases.

REFERENCES

[1] Bigum C. J., Burton, D. and Gilding, T. (1988). Learners as novice knowledge engineers. Educational Research and Perspectives. 14 (1) pp. 58-61.

[2] Eyler, Charles Edward. (1989). The effects of metacognition on mathematical problem-solving. (Doctoral dissertation, Pennsylvania State University). Dissertation Abstracts International. Order #AAC8922049.

[3] Gow, J. M. (1992a). ISDP in the one computer classroom. Unpublished paper. Nova University, Ft. Lauderdale, Florida.

[4] Gow, J.M. (1992b). Constructing cognitive models in Logo: A feasibility study with seventh graders. Book of Abstracts. Conference on Emerging Technologies in Education. ICSTI Moscow, USSRx.

[5] Greeno, J. G. (1989). A perspective on thinking. American Psychologist. 44, 134-141.

[6] Guernon, V. E. (1989). The effects of teaching heuristics within the context of a prescriptive metacognitive control system on problem solving. (Doctoral dissertation, University of Connecticut). Dissertation Abstracts International. Order #9006699.

[7] Harel, I. (1988). Software design for learning. Doctoral Dissertation Massachusetts Institute of Technology. Media-Laboratory. 20 Ames Street, Cambridge, Massachusetts.

[8] Hayes-Roth, F., Waterman, D. A. & Lenat, D. B. (1983) Building expert systems. Addison-Wesley Publishing Company, Inc., Reading Massachusetts.

[9] Lai, K. W. (1989). Acquiring expertise and cognitive skills in the process of constructing an expert system: A preliminary study. American Educational Research Association. San Fransisco, Ca.

[10] Lippert, R. C. (1988). Refinement of students' knowledge while developing expert systems. (Doctoral dissertation, University of Minnesota, 1988). Dissertation Abstracts International. Order #8900551.

[11] Murphy, D. D. (1985). The little inference engine that could. Proceedings Florida International Computing Conference (FICC). Orlando, Florida.

[12] Papert, S. (1990). A critique of technocentrism in thinking about the school of the future. E & L Memo No. 2. Cambridge MA: MIT. Media Laboratory.

[13] Perkins, D. H. (1986). Knowledge as design. Lawrence Erlbaum Associates. Publishers, Hillsdale, New Jersey.

[14] Starfield, A.M., Butala, K.L., England, M.M., Smith, K.A. (1983). Mastering engineering concepts by building an expert system. Engineering Education. 10, (1 & 2), 50-105.

Project Management Analyser in Logos

I. Krivy[1], J. Weinberger[2]

[1]Department fo Mathematics
University of Ostrava
701 03 OSTRAVA, Czech Republic

[2] TIMING, Hermanova 30
170 00 PRAHA, Czech Republic

Abstract

A program tool is presented for the simulation analysis of arising or existing projects regarding their time consumption, economical indices and risk factor. The Project Management Analyser (PMA) is based on the paradigm of neural network, whose neurons simulate the individual activities within the analysed project and axons represent connections among these activities (inclusive of time delays if any). PMA is programmed in Logos, which is a specialized language for discrete simulation with its procedures written in Logo. It can be used in learning process at both secondary schools and universities as a good starting point for understanding the principles of object oriented programming and mastering more sophisticated project analysers.

Keywords: Logo, Logos, project analysis, simulation, neural network, teacher training

Ivan Krivy *is a physicist (PH. D. in solid state physics) who switched to applied mathematics. He is now teaching courses in probability theory and statistics for both secondary school teachers and specialists in applied mathematics at the University of Ostrava. His major research activities are oriented to dynamical systems theory and computer simulation. Since 1985 he is working on the application of Logo and Logo-based program tools in teacher training. He is a committee member of the CSSS (Czexh & Slovak Simulation Society).*

Jiri Weinberger *is a mathematician with great experience in object-oriented programming and computer simulation. He has developed Logos - a program tool, written in Logo, for discrete simulation. Since 1992 he is working as a private dealer in making simulation and optimization software. He is a committee member of the CSSS (Czech & Slovak Simulation Society).*

Project Management Analyser in Logos

1. Introduction

In this paper, a program tool is presented for the simulation analysis of arising or existing projects from the standpoint of their time consumption, economical indices and risk factor. The main categories under consideration are time, money, resources and capacities, but the user is allowed to introduce any other categories necessary for solving his particular problems. Design of the simulation analyser (as well as of the commercial Project Management Forecast - PMF [1]) is based on the paradigm of neural network, whose neurons simulate the individual activities within the framework of the analyzed project and axons represent connections among these activities (inclusive of time delays if any).

This Project Management Analyser (PMA) is programmed in Logos, which is essentially a specialized language for discrete simulation with its procedures written in Logo (see [2, 3]). However, no special knowledge of Logos commands is required from the users of PMA. To start working with PMA it is necessary only to describe activities of the individual neurons in Logo. The execution of simulation experiments or risk analysis is controlled by the data that the users introduce using editing mode of Logo.

2. Description of PMA

Complete PMA version of the time consists of about 140 procedures and functions (inclusive of those defined in Logos), but usual users take advantage of no more than 30 of them. The "structure" of PMA is essentially similar to that of PMF. Two special classes of objects-processes are declared within PMA:
(i) class N (neuron),
(ii) class A (axon).

Any instance of the class N has 19 non-standard attributes, the most important of which are:

COST	charges caused by a given neuron,
FROM	time of starting a given iteration of neuron activity,
ITER	sequential number of iteration,
HOLD	duration of one iteration,
LOCK	indicator of permanent blocking of neuron,
RATE	charges referred to unit time period,
TOTT	total time period consumed by neuron,
PCON	pre-condition for starting neuron activity,
BUSY	indicator of neuron state (busy or idle),
WUMI	indicator of wait-until monitoring (value gives number of frozen activation signals),
AXON	list of system axons going out from neuron,
WAXN	list of wild axons going out from neuron.

There are also two attributes-procedures BEFH and AFTH, which describe the neuron activity before and after executing Logos command !HOLD, resp.

Body of the neuron class is composed of 12 segments. Each object of the class (after its activation) takes successively the following steps during its life:

(i) testing the entry conditions for starting its operation (tests of own PCON and LOCK attributes),
(ii) executing the commands of its BEFH procedure;
(iii) passivation for the time period given by the value of its HOLD attribute;
(iv) executing the commands of its AFTH procedure;

(v) activation of both the system and wild axons going out from it;
(vi) passivation until a successive activation signal comes to its input.

In case, an attempt at activating a given neuron is registered while the neuron is busy, this activation signal is frozen and will be of use as soon as the neuron completes its activity.

Declaration of the N class body is still general that the users need not step in it at all. Values of the individual neuron attributes are accessible by using so-called "dot notation".

Three attributes are defined in the A class; its meaning is as follows:

DEST reference to a target neuron,
MINI minimum transmission time for activation signal,
MAXI maximum transmission time for activation signal.

If the values of both MINI and MAXI attributes are the same, the transmission time is considered fixed and its value is equal to common value of these attributes. In case, the value of MAXI attribute exceeds that of MINI attribute, the value of transmission time is generated as a random number from uniform distribution in the time interval [MINI, MAXI].

From the standpoint of origin, two types of axons are distinguished:

(i) system axons, introduced into the simulation model automatically by analyzing PCON attributes of all neurons;
(ii) wild axons, defined by the user.

Data structures corresponding to axons of both types are identical; no differences being in handling with them. Each object of the A class (after its generation) lines up into a queue of free axons, which are at disposal for the transmission of activation signals, and then it is passivated. After having been activated by some neuron, it leaves the queue, takes the values of its three attributes from this neuron, waits for the corresponding transmission time, activates target neuron and, finally, lines up again into the queue of free axons. The users need not take care of axons, their attributes and "living rules".

The course of simulation experiment (as well as of risk analysis) is coded in the body of the class !MAIN. This class has no attributes. Only instance of the class ensures the following principal actions: the declaration of both the N and A classes, the initialization of all the global variables, the generation of requisite neurons and axons, the initiation of their interactions and, finally, the output of the results.

All the variables introduced by the user (when programming BEFH and AFTH procedures of the individual neurons) are of global type. Among these variables, a special position is taken up by the output variables, whose values are displayed or printed at the end of any simulation experiment. Some of the output variables constitute components of vector-type criterion for evaluating the simulation results. Besides the user's global variables, any simulation model within PMA involves also the system global variables, namely

START	initial value of simulation time,
COSTS	total charges of a project,
D	number of working hours per day,
RISK	admissible risk of project failure,
STOP	final value of simulation time for sim. experiment,
SHUT	final value of simulation time for risk analysis,
GOAL	indicator of attaining goal,
RAC	criterion for the acceptance of simulation results,
NTAX	initial number of free axons in model,
STN	list of the neurons initiating sim. experiment.

Identifiers of all the system procedures, the system global variables and the object attributes start with a special character (!, | and ^, resp.) in order to avoid the collisions with the identifiers defined by the user. In this paper, the special characters are omitted for brevity.

3. Simulation analysis of projects

After the simulation model of a given project having been established, the user can provide its snapshot referred to any simulation time. From the formal point of view, there are two principal ways of using PMA:

(i) individual simulation experiments,
(ii) risk analysis.

In the first case the user proceeds usually as follows. He loads the snapshot made at zero time, then edits the values of the system global variables (e.g. STOP, NTAX) and, finally, he starts with the execution. The user is also allowed to modify the pre-conditions and activities of neurons. If the modifications result either in changing at least one pre-condition or in defining at least one new variable, it is necessary to update the structure of neural network (by using GENAXON procedure to generate the corresponding system axons). At the end of each simulation experiment, all the output variables are displayed on screen or saved in a text file if needed.

The PMA takes in full advantage of the tracing tools implemented in Logos. As regards the tracing mode, the user has possibility of activating it at any simulation time.

The risk analysis represents in fact a simulation study (a sequence of simulation experiments with the same values of all the system global variables excepting the seed for random number generator) followed by the statistical evaluation of risk level. The analysis is controlled by the values of the system global variables RISK, RAC and SHUT. For each simulation experiment, the risk analysis protocol includes the following information items:

(i) sequential number of simulation experiment,
(ii) values of all the output variables,
(iii) indication of fulfilling the RAC criterion,
(iv) actual value of risk level (relative frequency of the unacceptable results).

PMA offers also some additional tools to the user, namely the statistical procedures for ran-dom drawing from a given distribution (e.g. normal, exponential, Erlang, Poisson) and the procedures for graphic description of the neural network structure.

When using the Logo interpreter developed by IBM in 1983, there are two principal limitations to PMA applications.

1. Calculations are very time-consuming. For example, one simulation experiment for a model consisting of six neurons and about 350 axons takes approximately six minutes of CPU time, when performing execution on IBM PC-AT486.

2. Size of the analyzed projects is also limited by small working area capacity of the interpreter (32 KB).

The current version of PMA sets certain bounds to programming the simulation model. In particular, the users are not allowed to place more Logos command !HOLD within the neuron class body, to introduce simulation time into the neuron pre-conditions and to use the neuron names in the model.

A simple example illustrating how to apply PMA to solving a real task will be given at the paper presentation.

4. Conclusions

In this paper, the description of the program package PMA (written in Logos) is presented for the simulation analysis of projects by using personal computers compatible with IBM PC-XT/AT. Its possibilities are evidently limited as compared with the commercial product PMF [1], supplied by the firm TIMING for solving complex problems in practice. Nevertheless, we believe that the proposed PMA package will be useful for evaluating relatively less complicated projects. It is simple (easy to learn), modern (based on the principles of object-oriented programming) and, when activating its tracing mode, capable of giving very objective image of the processes taking place in the simulated system. It can be of use first in the learning process at both secondary schools and universities, as a good starting point to understanding the principles of object-oriented programming and mastering more sophisticated program tools for the project analysis.

REFERENCES

[1] Weinberger, J.: *Project Management Forecast.* User Manual. Prague, Timing 1993.

[2] Krivy, I. - Weinberger, J.: *Logos - a Program Package for Discrete Event Simulation.* Acta Fac. Paed. Ostrav., A 25, 1990, pp. 39-54 (in Czech).

[3] Weinberger, J.: Logo, Simulation and Object-Oriented Programming. In: *Proceedings 3rd European Logo Conference* (August 1991). Parma, Italy, pp. 219-226.

Learning to Speak Mathematically and Speaking Mathematically to Learn in the Logo-based Environment "Geomland"

Evgenia Sendova[1], Bojidar Sendov[2]

[1]Bulgarian Academy of Sciences
Institute of Mathematics
"Acad. G. Bonchev" str. bl.8
Sofia 1113, Bulgaria
fax: +359 2 752078
e-mail: edit@bgearn.bitnet

[2]Sofia University
Faculty of Mathematics and Informatics
"James Baurchier" str., 5
Sofia 1126, Bulgaria

Abstract

The importance of having language based computer environments in search of a better achievement of the educational goals in mathematics is considered. The understanding, that in order to be educationally powerful, the environment should provide the best possible tools for defining new notions and expressing ideas that could be communicated and developed is suggested. Some supportive examples are demonstrated in the context of GEOMLAND - a mathematical laboratory in Logo style, being developed and experimented by Bulgarian researchers.

Keywords: Logo, mathematics, geomland, mathematical laboratory

Evgenia Sendova *graduated from Sofia University (the Faculty of Mathematics) in 1969. Since then she has been working at the Institute of Mathematics (Bulgarian Academy of Sciences). Her first steps as a researcher were in mathematics modelling. Since 1984 she has been working in the field of eduaction. She is author of a number of textbooks in informatics and mathematics for primary and secondary schools. Evgenia Sendova is a member of the editorial staff of the journals "mathematics and Informatics", "Education in mathematics anf Informatics" and "Mathematics Plus". Since 1985 she has been involved in the activity of the IFIP WG 3.5 dealing with the use of computers in primary schools.*

Bojidar Sendov *graduated from Sofia University (the Faculty of Mathematics) in 1968. He got his Ph.D from the Warsaw University in the field of informatics in 1979. Since then he has been working at the Faculty of Mathematics and Informatics (Sofia University). At present he is the head of the Educational Computer System Laboratory and is involved in the development and experimenting of language-based environments of a laboratoty type.*

Learning to Speak Mathematically and Speaking Mathematically to Learn in the Logo-based Environment "Geomland"

Logo's level of formalization is intermediate between mathematics and language - between Goedel and Goethe.
M.P. Doyle

1. Introduction

To enable students to see the beauty and the great ideas in mathematics, to guide their explorations and help them experience the joy of the discovery becomes an important goal for many educationalists in mathematics. When persuing this goal researchers involved with Logo-based educational environments generally implement the constructivism as a philosophy of learning through discovery. We share their view that a relevant (programming) language can serve as a vivid example for the potential of a computer-based constructivistic environment. Furthermore, we believe that such an environment can implement Seymour Papert's ideas of *thinking mathematically and mathematics as a language.*

Still a very arguable question for many designers of educational software is the following one:

2. Do we need a language when doing mathematics in a computer environment?

2.1 Speaking "naturally" v/s speaking "mathematically"

In an attempt to answer the above question let us first consider the definition of the notion of language as proposed in [10]: *A language is a system that relates some physical signal (a sound, a gesture, a mark on a paper) to meaning... We can think of a language as rich and complex cognitive systems that represent information in symbolic form.*

When learning mathematics students need to experience a broad range of language to develop the fullest understanding possible. Furthermore, they are expected to learn how to express themselves clearly and precisely. This might be a significant cognitive process of passing through different levels of formalization. The language provides a rich system of tools for students to represent those levels so as to make sense of the notions involved and (possibly) communicate them.

Virtually, every mathematical definition is a mixture of a natural language and special mathematical symbols. But even in the case of a natural language there is a distinction between having an idea and putting it into words. The more so with the mathematical language. Generally pupils are not aware why mathematical definitions should be so precise. When memorizing these definitions they often replace one word with another. Or they would make a mathematical construction in their notebook just mechanically, without taking into account the deep relationships among its components.

Although the very step from using a mathematical term to giving its precise definition requires considerable mental effort, the next step - to construct a certain representation of a notion based on its definition - appears to be even more difficult. A good reason for this is the fact that the mathematical definitions are most often of declarative (propositional) rather than procedural type. The distinction between *propositional knowledge and procedural knowledge is often colloquially expressed as knowing-that versus knowing-how* [7]. When doing mathematics students should learn how to shift from *propositional to procedural* mode of activities, i.e. they should not only *know that* but also *know how*...

2.2 "Talking" or "gesturing" to the computer?

Our belief is that computer microworlds based on Logo-like languages can help students establish links between the properties of mathematical objects and the actions needed to construct these objects. The use of the language will enable them to think about the relationship of the mathematical objects rather than constructing them automatically, without making sense of it. And not less important, the language-based environments stimulate student-to-student and student-to-teacher verbal communication about experiments thus helping to match one's mental representation to somebody else's.

A very important common characteristic of the mathematical and the natural languages is the fact that they are based on a system of primitive notions which can be extended and enriched by defining new notions on the basis of the existing ones.

This characteristic is typical also for the Logo-like languages enabling users to define new notions (in this case procedures) and thus - to extend the initial vocabulary of the language according to their needs.

Of course *language* could be a series of options made available from menus but why impose a *sign language* on people who already speak (or are expected to speak) the language of mathematics? We would not like though to contrast the menu approach with the language approach. Our concern is that students should not experience any constraints on their working style, but rather should be able to choose between menu-driven, language-based and possibly some other activities and to combine them. Besides, the very process of creating definitions of procedural type (i.e. whose validity might be easily checked) is of a great educational power since the learner can experience what it is to be *a builder of one's own knowledge.*

To summarize, we consider a computer microworld to be a good learning environment if it provides the best possible tools for defining new notions and (similarly to the mathematics and natural languages) for expressing in an unambiguous way ideas that could be communicated and developed.

3. In which part of mathland to settle a microworld?

Geometry seems to be a very reasonable answer:

- it is a rich ground where the students can do mathematics and thus learn mathematics in an active and constructive, as opposed to a passive, way [6];
- it is an ideal area in which to help students develop mathematical habits of mind, such as seeking patterns, representing the essential aspects of situations, describing things precisely, and using visualizations [2].

Although researchers in mathematics education see the potential of geometry as a field in which children can practise inductive inferences from personal experience while simultaneously being engaged in deductive thinking, *there is a little evidence of children working in geometrical environment which might fulfil the above potential for geometry in education* [5].

Significant hopes to change this situation were set on the *turtle geometry* - the first famous Logo microworld. Most of the teachers however have been traditioned to think of geometry in a particular way and when they were suddenly faced with a relative geometry, the effect of the *shackles of Euclid* was evident. The designers of Logo-like microworlds faced a new problem - how to harness the novelty with the tradition? In other words - how to preserve the best features of the Logo philosophy in a more traditional geometric context?

A possible answer is suggested by a group of Bulgarian researchers who are developing a computer microworld called GEOMLAND. This microworld was designed in 1985 (originally under the name of PLANE GEOMETRY SYSTEM) and is under development at the Educational Computer System

Laboratory at Sofia University. GEOMLAND represents a mathematical laboratory in a Logo style [8] where various experiments with geometric objects can be performed, their properties can be investigated and conjectures may be formulated and verified.

3.1 Do you speak the Logo-dialect of GEOMLAND?

If you know Logo and geometry the answer is YES. Because the language of GEOMLAND is a Logo extension designed for working with geometric objects [3]. A new data type, representing the basic objects of plane geometry (point, segment, line, ray, circle, vector and set of such objects) has been added in the frames of the traditional Logo syntax and semantics. The operations that can be performed upon these objects are: constructing a value of a certain type, extracting components of a value, intersection, checking for incidence with certain precision, etc. The primitives for constructing geometric values (the so-called constructors) and for extracting components of geometric values (the so-called selectors) correspond to the traditional names: POINT, SEGMENT, LINE, RAY, CIRCLE, VECTOR, SET, ABSC, ORD, LENGTH, HEADING, RADIUS, CENTER, COUNT, etc. There is a primitive POINTON - for constructing a point on a line, on a segment, on a ray or on a circle; there is also a predicate POINTON? - for checking the incidence of a point with a geometric object of the above types.

The main primitive used for constructing and modifying geometric objects is OBJECT. It takes a name and a value as inputs. As a result the value is assigned to the name and the object is constructed and visualized on the graphics screen. The OBJECT primitive is equivalent to MAKE extended with visualization of the geometric objects and an automatic maintenance of the relationships among the newly constructed objects and those, used in their construction.

One of the extra features of the lines and segments in GEOMLAND (as it will be seen in the examples below) is that they have direction with the first input being the initial point (the origin).

3.2 Reflecting on the reflection

Defining new geometric notions and then generalizing them is a process which can be carried out in GEOMLAND in a very convenient way - the language is close to the traditional mathematical language used in plane geometry and the definitions are executable, i.e. their validity can be easily checked by the learner.

Let us demonstrate the process of generalizing a well known mathematical notion - that of reflection (symmetry). We start with the definition of the reflection of a point (P) with respect to a line (L). We could follow the process of construction - first constructing a line L1 passing through the given point P and being perpendicular to the given line L; next finding the intersection O of the two lines; then constructing ray R starting at P and passing through O; and finally finding the reflection point of P with respect to L as a point on the ray R at a distance from its origin (point P in this case) twice as long as the distance between P and O:

```
TO REFLECT_PL :P :L
        LOCAL "L1 "O "R
        MAKE "L1 LINE :P 90 + HEADING :L
        MAKE "O ISEC :L :L1
        MAKE "R RAY :P :O
        OUTPUT POINTON :R 2*DISTANCE :P :O
END
```

We could also follow the mathematical definition more closely [1] according to which: *A reflection in a line L is a transformation among points in a plane that pairs each point P of the plane with a point P' in such a way that L is the perpendicular bisector of the segment PP' as long as P does not belong to L, otherwise P = P'.*

The corresponding procedure REFLECT_PL will look as follows:

```
TO REFLECT_PL :P :L
      OUTPUT POINTON SEGMENT :P (ISEC :L LINE :P 90 + HEADING :L) 2
END
```

Here is the place to reflect on the two versions of REFLECT_PL which represent somewhat different approaches manifested by the students to the notion of the *reflection image of a point with respect to a line.* The first one deals with small steps corresponding to some intermediate elements making the procedure more readable; the second one makes use of the functionality of the language and reflects the mathematics definition better.

A lot of other versions of the REFLECT_PL procedure are possible. What appears very often to be a challenge for teachers is to react adequately when certain versions *work* only in special cases. Here is a typical example:

```
TO REFLECT_PL :P :L
      LOCAL "L1 "O
      MAKE "L1 LINE :P 90 + HEADING :L
      MAKE "O ISEC :L :L1
      OUTPUT POINTON :L1 2*DISTANCE :P :O
END
```

Not only does this procedure seem synonymous to the first one but even more economical. The problem though is that it outputs the reflection image of point P only when it is from the half-plane to the right of line L (remember that the lines in GEOMLAND are in fact axes). It is only after suggesting to students to construct the reflection point P" of the reflection point P' of P with respect to L that they realize that P and P" do not coincide (differently from what should have happened). Alternatively they might ask if P and P" are incident:

```
POINTON? :P (REFLECT_PL (REFLECT_PL :P :L) :L)
```

Thus students become convinced that their definition does not work in the general case.

Further on, to construct the reflection image of a line L1 under reflection in line L we take two points on L1 and find their reflection points with respect to L. The desired line will be determined by those two points (the best solution is to take the first point to be the line's origin and the second one - a point on its positive direction at a distance, say 100 from the origin):

```
TO REFLECT_LL :L1 :L
      LOCAL "A "B
      MAKE "A INITIAL :L1 0
      MAKE "B POINTON :L1 100
      OUTPUT LINE REFLECT_PL :A :L REFLECT_PL :B :L
END
```

Similarly we can describe the reflection in a line L of a segment S (REFLECT_SL), of a circle C (REFLECT_CL) and of a ray R (REFLECT_RL). Finally we can generalize all the procedures into one outputting an object symmetric to a given one with respect to a line:

```
TO REFLECT_L :O :L
   (OUTPUT  CASE TYPE :O  [[ "POINT  REFLECT_PL :O :L]
            [ "SEGMENT REFLECT_SL :O :L]
            [ "LINE  REFLECT_LL :O :L]
            [ "CIRCLE  REFLECT_CL :O :L]
            [ "RAY  REFLECT_RL :O :L] ])
END
```

Another more Logo-like version of the above procedure is the following one:

```
TO REFLECT_L :O :L
      OUTPUT RUN (LIST (WORD "REFLECT_    (FIRST TYPE :O) "L) :O :L)
END
```

Similarly the notion of reflection of an object about a point (point symmetry) could be developed starting with separate procedures for the cases of the first input being a point, a line, a ray, a circle and a segment, respectively, and then generalizing them into a single procedure REFLECT_P. Combining the two general procedures REFLECT_L and REFLECT_P we can obtain a single procedure REFLECT which outputs the reflection of a given object O1 (point, line, ray, segment, circle) with respect to object O2 (point or line):

```
TO REFLECT :O1 :O2
      OUTPUT (CASE TYPE :O2
              [[ "POINT    REFLECT_P :O1 :O2 ]
               [ "LINE REFLECT_L :O1 :O2 ]
              ])
END
```

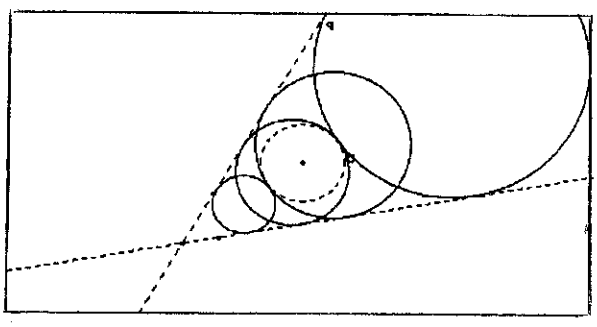
A still higher order of generalization takes into account the case of the reflection image of a set of objects. Again different approaches are possible but our favourite example (developed by a university student who is a great Logo-admirer) makes use of the recursion, of course:

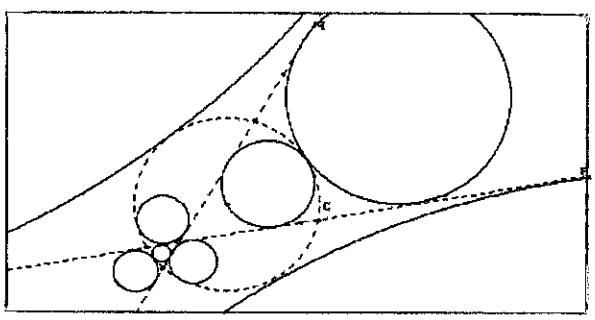
```
TO REFLECT :O1 :O2
      (IF (TYPE :O1) = "SET
        [OUTPUT (IF 0 = COUNT :O1
                 [(SET)]
                 [ SET  REFLECT (BF :O1) :O2
                        REFLECT (FIRST :O1) :O2])
        ]
        [OUTPUT (CASE TYPE :O2
                 [["POINT REFLECT_P :O1 :O2]
                 ["LINE    REFLECT_L :O1 :O2]
                ])
        ])
END
```

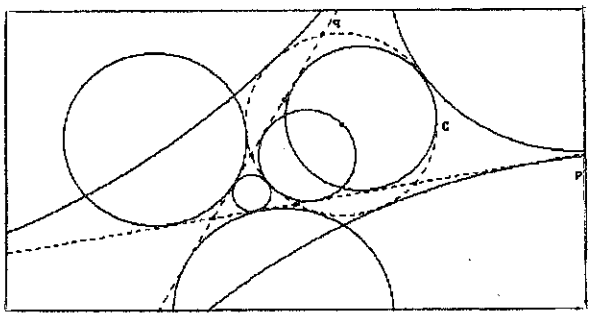
There are plenty of other interesting mathematical notions whose dissection might have a great educational effect.

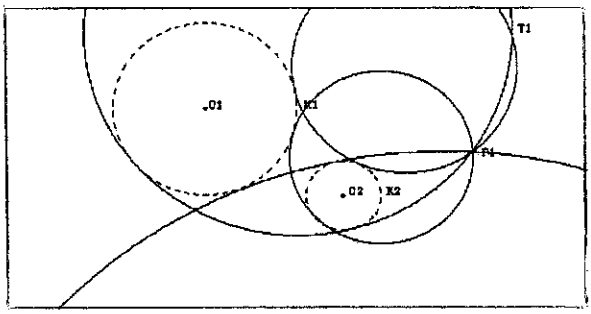
Let us mention here as an example the incidence of a circle with three objects each one of which could be a point, a line or a circle. The simplest case is a circle passing through three points. The generalization in the above sense (i.e. when each one of the inputs can be of any type) is a real educational eppopee embracing such a geometrical transformation as the inversion.

Some examples of constructing circles incident with three fixed objects (represented by dotted lines) are given below:

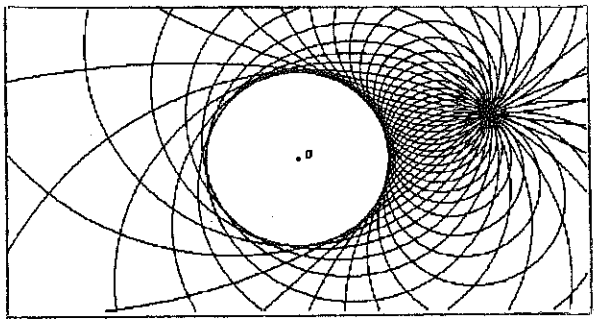


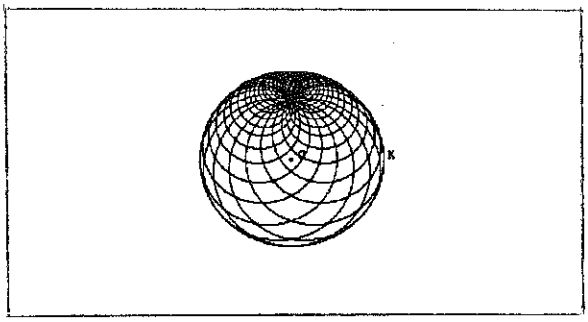






This constructive problem is full of special cases and its precise consideration demands a very subtle language. Here are two such cases (with the three fixed objects being a circle and two coinciding points) when the result contains infinitely many solutions:





Our belief is that when represented in a Logo-like environment the inversion (which has generally been avoided at school) might become a fabulous educational topic for the secondary school [12].

4. Conclusions

Many researchers find that the naming and defining of objects through language is of a significant educational effect since *it leads to the abstraction of the object's qualities so that they can be manipulated as objects of thought* [11].

Our experience with 7th graders [4] and with future teachers [9] shows that the process of generalizing a concept, of reconstructing it in new situations broadens and deepens the concept.

Having been involved in the definition and the generalization of some geometric concepts in the context of GEOMLAND, students would use later on the procedures defined by them on a higher level - as primitives, in-built in the language. The important message here is to cultivate in the learners a style of creating themselves the tools needed for a specific activity. (As was the case with the best masters who created themselves the tools they needed).

Recently, the Faculty of Mathematics and Informatics at Sofia University provides the students graduating as specialists in informatics or mathematics with the additional qualification of teachers. It was when working with those students that we felt their satisfaction of the richness and the flexibility of a language-based environment - enabling the mathematicians to confirm their understanding that the language is a powerful tool, and the informaticians - to see attractive applications of their programming skills in a mathematical context. In such a way the links between mathematics and informatics are reinforced for their mutual benefit.

If our ultimate goal is to create such an atmosphere in the mathematical classroom that each student should be able to fully express his/her individuality we have to think of environments where *the easy things are done*

easily but at the same time where *there are no thresholds of one's imagination.* Thus we would put off neither pupils who feel certain fear from mathematics nor those who realize that mathematics is more than a craft, that it is art!

Acknowledgments

We would like to express our deep gratitude to prof. Edward Friedman from the Stevens Institute of Technology (NJ) who appriciated our work with GEOMLAND and during his stay in Bulgaria encouraged us with plenty of interesting ideas. He and John Muirhead (a specialist in natural languages from Scotland) contributed a lot to the English version of GEOMLAND's documentation and being involved in the naming of the system produced as an output the name of GEOMLAND.

REFERENCES

[1] Billstein et al, Problem Solving Approach to Mathematics for Elementary School Teachers, The Benjamin/Cumming Publishing Company, 1987, p.521

[2] Cuoco A. and E. Goldenberg, Reconnecting Geometry: A Role for Technology, Education Development Center, Newton, MA, (1992) p.2

[3] Filimonov R. and B. Sendov, Drawing Logo Closer to the Curriculum, in G. Schuyten and M. Valcke, eds., Proceedings of the Second European Logo Conference, Gent, Belgium, 30 August - 1 September, 1989, pp. 363-373

[4] Kolcheva M. and E. Sendova, Re-inventing the "Elements" in a Logo-based Environment, in EUROLOGOS (Incorporating LOGO Almanack), M.Doyle (Editor), Vol.1, 1992, England, EC, BD23 1QQ, pp. 3-6.

[5] Kynigos C., Can Children Use the Turtle to Develop Understandings of Euclidean Ideas in an Inductive Way?, in U. Leron and N. Krumholtz, eds., Proceedings of the Fourth International Conference for Logo and Mathematics Education, Kibbutz Ramat Rachel, Israel, 2-5 July, 1989, p.115

[6] NCTM, Curriculum and Evaluation Standards for School Mathematics,(1987)

[7] Papert S., Mindstorms, New York: Basic Books (1980), p.135

[8] Sendov B. and D. Dicheva, Mathematics Laboratory in Logo Style, in F. Lovis and D.E. Tagg, eds., Computers in Education, Proceedings of the IFIP TC3 First European Conference on Computers in Education (ECCE'88), Lausanne, Switzerland, 24-29 July, 1988, pp. 213-217.

[9] Sendova E. and B. Sendov, Columbus, Da Vinci or Prometheus: A new Role for the Mathematics Teacher in Computer Environment (to appear in the Journal of Technology and Teacher Education)

[10] Stillings N. et al, Cognitive Science, A Bratford Book, (1987) P.215

[11] Swan K., "Knowing What": Declarative Knowledge and the Cross-Contextual Transfer of Problem Solving Skills, in E.Calabrese ed., Proceedings of the Third European Logo Conference, Parma, Italy, 27-30 August, 1991, p.573.

[12] Yalamova S., Appolonius Problems in Geomland (in press)

Network Paradigm Modelling in the Logos Language

Jaroslav Sklenar[1], Katerina Sklenarova[2]

[1] University of Malta
Department of Computer Studies
Msida, Malta

[2] Verdala International School
Pembroke, Malta

Abstract

The paper deals with Petri net based modelling of discrete systems. The modelling environment has been written in the Logos language which is a Logo extension. The Logos language supports Object Oriented Simulation using ideas of the Simula's system class Simulation. The paper introduces the environment and describes a typical simulation study of a simple communication system for message transmission. The model is at first expressed as a Petri net, then as the so called Time net. This enables construction of a simulation model which may be used for experimentations providing statistical parameters of the original system. Environment for both abstractions (Petri and Time nets) has been designed using the inheritance principle of Object Oriented Programming. The modelling is based on individualisation of standard procedures generated during network definition. The purpose of the software is a support of education of the topics mentioned above.

Keywords: Logos, petri nets, time nets, object oriented simulation

***Jaroslav Sklenar**, Ph. D. graduated at the Electrotechnical Faculty of the Technical University of Brno, Czechoslovakia in 1973. Then he became a postgraduate student at the Department of Telecommunications of the same university. He got a Ph. D. degree in 1979. His doctoral thesis dealt with the use of System Theory in Simulation of discrete systems. From 1982 he was as Senior lecturer at the Department of Telecommunications. Now he teaches at the University of Malta, Department of Computer Studies. His main interest is Computer Simulation and all related topics.*

***Katerina Sklenarova**, M.S. graduated at the Electrotechnical Faculty of the Technical University of Brno, Czechoslovakia in 1973. After graduation she became a member of the Department of Mathematics of the same University, where she worked as a lecturer. Now she teaches Computing Studies and Mathematics at the Verdala American International School in Malta. Her main interest is Education of Computing and related topics.*

Network Paradigm Modelling in the Logos Language

1. Introduction

Paradigm oriented modelling is based on some concept (abstraction, paradigm) which is used as a tool for the description of the model. The tool used depends on the nature of the system that is modelled (simulated). A commonly accepted tool for a description of discrete systems are Petri networks and their modifications. Petri networks (P-nets) enable a very natural and lucid way how to express cooperation and communication of parallel processes. This is very useful for example in modelling computer and communication systems but there are also many other application areas. Description of P-nets and their modifications is out of the scope of this paper. In [2] and [5] there are all necessary definitions and results for P-nets, [1] deals especially with their modifications. A very introductory explanation of basic ideas can also be found in [4]. It is necessary to point out that P-nets have also some powerful analytical capabilities. The idea is as follows: describe the analysed system as a P-net and analyse it to find out its properties. From the known properties of the P-net it is possible to deduce some properties of the original system. Unfortunately there is a contradiction between the analytical and the modelling power. The original P-nets are very analytically powerful but their modelling capabilities are relatively limited. All modifications of P-nets increase the modelling power but the analysis becomes difficult or impossible - see [1]. This paper deals with modelling applications of network models to show the modelling capabilities of the Logos language.

2. The Logos Language

In [6] there is an introduction to the software package Logos for *Object Oriented Programming* (OOP) and *Object Oriented Simulation* (OOS) written in Logo (referred to as the Logos language from now). From Logo point of view the new constructs are procedure calls (all starting by "!") with appropriate parameters. In [4] and [6] there are examples of Logos use. The simulation in Logos consists in definition of classes (prototypes of object instances) and their subclasses possibly in several steps. This is the first main idea of OOP: specialization of general classes by a definition of additional attributes using all attributes of the parent class (*inheritance*) in so many steps until the classes with desired features are obtained. Another principle of OOP is the fact that objects may have both value and procedure attributes-*methods* (Object = Code + Data). Logo untyped variables are a very efficient support for this. After definition of attributes (Logo variables with special names) it is possible to treat the attribute as a variable one (both simple variable or a list) or as a procedure. Third main principle of OOP is the *Polymorphism*: different behaviour of objects generated using the same pattern in various situations. Here the Logos goes even further than most OOP oriented languages. Methods of object instances are generated for all of them separately. The consequence of this is a possibility to change them according to the modelling needs (*individualisation* of methods). Most OOP languages use the same code for all object instances and individualisation can be achieved by using value attributes. To support OOS the Logos language offers procedures for work with (quasi)parallel processes (the most advanced way of time control in discrete event simulation). The procedures enable generation, activation, deactivation and synchronisation of processes. The ideas used are taken from the system class Simulation of the Simula language - see [3]. This paper does not assume a detailed awareness of them.

3. The PETSIM Modelling Environment

In [4] there is an introduction to the modelling environment PETSIM written in Logos. PETSIM enables creation, modification and simulation of any number of P-net based models at a time. The appendix A of

[4] is a simple user guide describing all PETSIM commands. Modelling in PETSIM can be split up to these steps: 1) off-line creation of the P-net. 2) Inputting the P-net by entering the numbers of places and transitions, the markings of places and for all transitions the input and the output arcs. An arc is defined by its input/output place and by its weight (multiplicity). 3) Simulation of the P-net. For simulation (firing of transitions) there are 4 different modes: 0 - stop before every transition firing, 1 - stop if there are at least two enabled transitions (the user selects the transition to be fired), 2 - random selection of the transition to be fired with displaying messages about enabled and fired transitions, and 3 - fast run with random selection without any displaying. In the modes 0 and 1 the user has a complete control of the behaviour of the net. The mode 2 enables an observation of the behaviour. The mode 3 is used for fast searching for *deadlocks* (states with no enabled transitions). The simulation experiment may be interrupted, markings of places may be changed and the simulation may be resumed. The simulation experiment finishes by reaching a deadlock or it may be stopped by the user. After deadlock, the markings of places can be changed and the simulation may be resumed. The simulation can be interrupted and the whole system (all procedures and variables) can be saved to disk (breakpoints of long simulation experiments). The PETSIM commands are analysed and processed using standard Logo and Logos commands. All objects involved in PETSIM supported simulation (places, transitions and P-net as such) are declared as Logos objects, it means for all of them there is a prototype (class declaration) used as a pattern when generating instances of objects (individual nets, places and transitions). Because all object instances are generated dynamically, there can be any number of P-nets of any complexity at a time - the only limitation is memory capacity. PETSIM has been used in educatin of the subject "Modelling and Simulation" at the university level. Because of simplicity of the commands students were able to master PETSIM and to enter and simulate P-nets during one laboratory session (assuming the P-net has been prepared).

3.1 Simulation using P-net modifications

The classes used for P-net based modelling which are very general can be used as parent classes in declaration of more specialized subclasses. This approach has been used for implementation of more specialized modelling environments which use modifications of P-nets as modelling paradigms. In [4] there is an outline of two such modelling environments supporting modelling of both Petri and Coloured nets and both Petri and Time nets. Always it is possible to work with any numbers of nets of both types at a time.

4. Building a Simulation Model

This chapter describes all steps involved in building a simulation model. Some steps are general, used in all simulation studies, some are specific for network based modelling. The example model is a relatively simple one which enables its complete and fast mastering by students. More complicated models don't use more ideas, only the memory requirements and especially the computing time are bigger. There are these main steps: Informal description, Creating a P-net model, Experimentation with the P-net model, Creating a Time net model and Experimentation with the Time net model. Experimentation involves also planning and evaluation of experiments.

4.1 Informal Description of the System Simulated

The system simulated is a hypothetical communication system which transfers messages from the SENDER to the RECEIVER stations in one way only. The user of the station SENDER generates randomly messages. The message can be sent to the RECEIVER station if the SENDER is ready. The SENDER is ready at the beginning and then after every successful transmission of a message. The fact, that a message has been successfuly transmitted and accepted by the RECEIVER station is announced by a special short Acknowledgement message sent from the RECEIVER to the SENDER. To accept a message the RECEIVER must be ready. The RECEIVER is ready at the beginning and then after every completed

processing of a message by the user of the RECEIVER station. The duration of the transmission, the intervals between generating messages and the duration of the processing in the RECEIVER's user are known. Also the probability of error free transmission is known. If there is an error (an ideal error detection is assumed) the message must be retransmitted as many times until there is no error. The question to be answered by the simulation study is like that: is the message transmission protocol correct? If yes, what is the throughput of the system if the transmission speed and the message size are also known ?

Informal description of this type is very often converted straight to the simulation model (program). There are many simulation languages supporting this process. But writing the model straight from an informal description can be dangerous. Informal descriptions like the one above may contain ambiguities, omissions or even contradictions. If the model is a large-scale one, it may be difficult or practically impossible to find errors in informal descriptions. That's why there is a permanent effort to use formal expressing tools to make the analysis of the model either simple or even to automate it. The least we can get from a formal description is the fact that its author is forced to express the whole system in a unified and consistent way which itself can help a lot.

4.2 Creating the Network Model

The Petri network which models the transmission system described above is in the figure 1. The P-net is a formal description of the system simulated. Markings of its places represent the following facts about the transmission system:

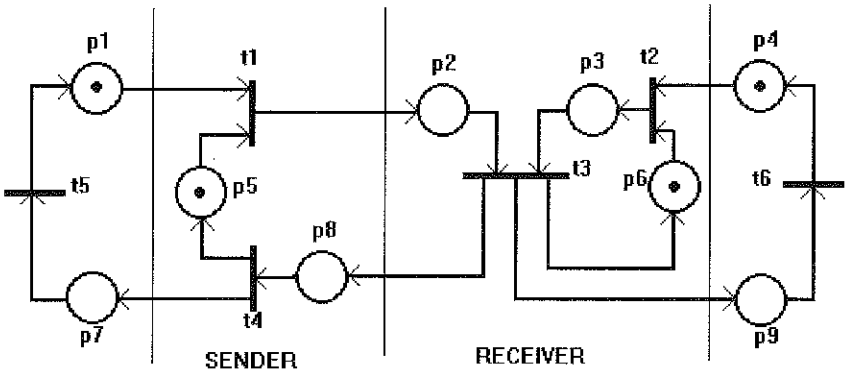


Fig. 1

p1 - the message has been generated by the user of SENDER,
p2 - the message has been transmitted to the RECEIVER,
p3 - the RECEIVER is waiting for a message (result of a request from its user),
p4 - the user of the RECEIVER asks for a message,
p5 - the SENDER is ready for transmission,
p6 - the RECEIVER is ready to accept a message,
p7 - Acknowledgement passed from the SENDER to its user,
p8 - Acknowledgement received by the SENDER,
p9 - the message passed to the user of the RECEIVER.

The transitions represent the following activities in the transmission system:

t1 - transmitting the message from the SENDER to the RECEIVER. For this transition to be fired there must be at least one token in the places p1 and p5 because there is one arc from p1 to t1 and one arc from p5 to t1. All places of the net in figure 1 have either one or no token (multiplicity of all arcs is 1). Using the transmission terminology for a message to be transmitted the message must be generated (p1) and the

SENDER must be ready (p5). The firing of the transmission t1 removes tokens from the places p1 and p5 and adds a token to the place p2 because there is one arc from t1 to p2. This means that the message has been transmitted.
t2- the RECEIVER accepts a request for a message from its user. For this transition to be fired the request must be generated (p4) and the RECEIVER must be ready (p6). The result of the firing is the fact that the RECEIVER waits for a message (p3).
t3 - the RECEIVER accepts a message, passes it to its user (p9), sends the Acknowledgement (p8) and becomes ready for the next message (p6). For this transition to be fired the message must be present (p2) and the RECEIVER must wait for it (p3).
t4 - the Acknowledgement (p8) is accepted by the SENDER, it is passed to its user (p7) and the SENDER becomes ready (p5).
t5 - activity of the SENDER's user between accepting an Acknowledgement (p7) and generating the next message (p1).
t6 - activity of the RECEIVER's user between accepting a message (p9) and asking for the next one (p4).

The correspondence between the P-net and the transmission system is evident. The P-net in the figure 1 is in the initial state (both stations ready, a message prepared to be sent at SENDER's end, a message awaited at RECEIVER's end).

4.3 Petri Net Simulation

The P-net in figure 1 has been inputted to PETSIM by entering its parameters (see chapter 3). The creation is simple, the user is always prompted what to enter. After creation the P-net and all variables and procedures were saved to disk. The following is the beginning of the simulation (comments in { } have been added, the bold text is entered by the user):

```
?LOAD "EXPNET                          { loading the net and the whole environment}
?!GO                                   { Logos restart, P-net is in the initial state - see fig. 1}
* * * ENTER COMMAND ( C | F | F N | S | S N | Q )          { PETSIM prompt }
{ C    = create a new P-net ,          F  = fix (modify) the last created P-net, }
{ F N  = fix the P-net number N,       S  = simulate the last created P-net, }
{ S N  = simulate the P-net number N,  Q  = quit to Logo prompt }
S                                                  { command "Simulate the current net" }
SET MODE | DISPLAY | QUIT ( 0 | 1 | 2 | 3 | D | Q )   { Simulate command's prompt }
0                                      { starting simulation in the mode 0: always waiting }
TRANSITION 1 ENABLED          { list of enabled transitions, prompt to enter a number }
TRANSITION 2 ENABLED
ENTER 'S' TO SET MODE OR NUMBER OF TRANSITION TO BE FIRED
2                  { firing the transition number 2: accepting a request by the RECEIVER }
TRANSITION 2 FIRED
TRANSITION 1 ENABLED
PRESS 'S' TO SET MODE OR ANY KEY TO CONTINUE
                                       { only one transition enabled - pressing eg. spacebar }
TRANSITION 1 FIRED    { message taken by the SENDER from its user and transmitted }
TRANSITION 3 ENABLED
PRESS 'S' TO SET MODE OR ANY KEY TO CONTINUE          { spacebar pressed }
TRANSITION 3 FIRED                     { message accepted by the RECEIVER }
TRANSITION 4 ENABLED
TRANSITION 6 ENABLED
ENTER 'S' TO SET MODE OR NUMBER OF TRANSITION TO BE FIRED
4                                      { Acknowledgement accepted by the SENDER }
TRANSITION 4 FIRED                                                    { etc. }
```

Simulation in the mode 0 enables a detailed analysis of P-net's behaviour, which in this particular model means analysis of the transmission protocol used. For large networks it is very difficult or impossible to evoke all possible network states. The partial help is simulation in mode 2 or 3 (fast run without displays). After a long run it may be assumed, that deadlocks are at least not very probable.

4.4 Time Net Simulation

P-net simulation like the one mentioned above enables analysis and proof of correctness of synchronisation and communication protocols. But it is evident, that P-net as a modelling tool is relatively very primitive especially if some quantitative parameters are to be obtained. From simulation point of view there is no notion of time at all. That's why another network model has been created using the so called Time net. The environment which supports work with both Petri and Time nets is called TIMSIM. Its implementation has been outlined in [4]. There are several approaches how to incorporate time to P-nets. We use a very general Time nets based on the following ideas which are mostly generalisations of the original P-net mechanisms:

1) A transition is enabled if its the so called *predicate* is true. The predicate is a function whose parameters are markings of input places. For every transition the TIMSIM generates a procedure which tests whether the predicate is true or not. Markings can be any data structures, not only tokens as in P-nets. Because Logos is used as a host language, marking can be "anything" what can be stored in a Logo variable. For all places the TIMSIM generates three procedures. The 1st one returns the marking, the 2nd one modifies the marking if the place is the input one and the 3rd one modifies the marking if the place is the output one. These procedures are generated in the standard Petri form. It means returning the number of tokens, subtracting token(s) for input place and adding token(s) for output place. Logos generates all these procedures separately. That's why the user may (and is supposed to) modify them according to the modelling needs. For example when updating markings it is possible to collect statistical data, when reaching some values it is possible to display messages, etc. The standard generated form of the procedure which tests the predicate is the Petri test whether there is a sufficient number of tokens in input places of the transition.

2) Firing of a transition takes some time. In this way time is introduced to network models. For every transition the TIMSIM generates a procedure which computes the duration of the firing. This may take into account markings of input and/or output places, it may be random, etc. These procedures are also supposed to be edited (for all transitions).

3) For all transitions there are another two procedures. One is called when starting a firing of the transition, the other is called when finishing the firing. In the standard (generated) form these procedures call the procedures of input respectively output places of the transition to remove respectively add tokens. Again these procedures can be edited according to the modelling needs.

Unlike P-nets Time nets use a body of the transition class . The body is a process (in Logos interpretation). In the standard form the body procedure repeatedly evaluates the predicate. If it is true the transition is fired, if not the transition process enters a queue of temporarily suspended processes (in fact it is a new attribute of the Time net class). Places have another new attribute which is a list of transitions that have this place as an input one. Every modification of every place marking activates transitions from this list to give them chance to continue in operation.

The following are some examples of modified procedures (the text in bold has been added, the normal text has been generated by TIMSIM. Comments in { } are not a part of the procedures):

```
{Method S (Simulate) of the object $_T1. $_T1 is a Time net. This method controls the simulation
experiment. After the standard beginning the method prepares the particular experiment. }
TO $_T1.S :N          { the parameter is the Time net number - used to create the names }
( LOCAL "P "I )                                                    { local variables }
```

```
MAKE "P WORD "$_T :N                                  { creating the Time net's name }
IF THING (WORD :P ".EX) = 0                           { test if Time net exists }
  [ PR [TIME NET DOESN'T EXIST YOU CAN CREATE IT] STOP ]
MAKE "$1 :N    { number of the Time net - the variable $1 is used later in other methods }
MAKE "NOMG 0      { model initialization:  NOMG = number of generated messages }
MAKE "NOMT 0              { NOMT = number of successfully transmitted messages }
TYPE [ENTER EXPERIMENT LENGTH : ]
MAKE "$DT RW       { reading the global variable which stores the experiment length }
TYPE [WATCHING ? ( Y / N ) : ]
MAKE "REPO RC         { It is possible to switch on/off (REPO = 1/0) reports about }
PR :REPO                                              { events in the model on the screen }
PR "
IF :REPO = "Y [MAKE "REPO 1] [ MAKE "REPO 0 ]
MAKE "I 1
{ activating all NOT transitions using the Logos activating command !ACT in current time: }
REPEAT THING (WORD :P ".NOT)
     [ !ACT (WORD "$T_T :I) !TIME MAKE "I I+1]
END

{ Method EVAL (Experiment evaluation) of the object $_T1: }
TO $_T1.EVAL :N
PR [EXPERIMENT COMPLETED]
PR (LIST "TIME "NET (WORD "$_T :N) )
PR "                                                  { printing some statistics }
PR (LIST :NOMG "MESSAGES "GENERATED)
PR (LIST :NOMT "MESSAGES "TRANSMITTED)
MAKE "NOMG :NOMT / !TIME
PR (LIST "TRANSMISSION "RATE :NOMG "MESSAGES "PER "SECOND)
PR "
END

{Method DUR (Computing duration) of the object $T_T1. This is the transition t1- see fig 1.}
TO $T_T1.DUR
LOCAL "T
MAKE "T 1.2                          { transmission duration if there is no error }
  { up to 10 retransmissions if there is an error in the message. The error probability is 0.5: }
REPEAT 10 [ IF (RANDOM 9) < 5  [GO "OK]  [MAKE "T :T+1.2] ]
LABEL "OK
OP :T
END
```

There are many more modified procedures but the idea of TIMSIM based simulation is evident: Enter the networks topology + modify methods of objects if necessary. Modification is performed by the standard Logo ED command. After modification the whole environment should be stored to disk.

The following is an example of the simulation (the text in bold is entered by the user, the comments in { } have been added):

```
? LOAD "TNTRA                                         { loading the model }
? !GO
{ This is the TIMSIM prompt: }
* * * ENTER COMMAND ( C P | C T | F | F P N | F T N | S | S P N | S T N | Q )
{ C P    =  create a new Petri net,          C T    =  create a new Time net, }
{ F      =  fix (modify) the last created net,    F P N  =  fix the Petri net number N, }
{ F T N  =  fix the Time net number N,        S      =  simulate the last created net, }
```

```
{ S P N  =  simulate the Petri net number N,  S T N =  simulate the Time net number N, }
{ Q      =  quit to the Logo prompt }
S                                                  { starting the simulation experiment }
ENTER EXPERIMENT LENGTH : 15                                        { model's prompt }
WATCHING  ? ( Y / N ) : Y                                  { displaying messages ON }
TIME 0 : MESSAGE GENERATED
TIME 2.6 : MESSAGE RECEIVED
TIME 2.7 : ACKNOWLEDGEMENT RECEIVED
TIME 2.7 : MESSAGE GENERATED
  { etc. }
TIME 14.4 : MESSAGE GENERATED
EXPERIMENT COMPLETED
TIME NET $_T1
9 MESSAGES GENERATED
8 MESSAGES TRANSMITTED
TRANSMISSION RATE 0.5333 MESSAGES PER SECOND
* * * ENTER COMMAND ( C P | C T | F | F P N | F T N | S | S P N | S T N | Q )
Q
?.DOS
```

Conclusion

The modelling environment is intended as an educational tool. It shows very lucidly the ideas of OOP and OOS. Its disadvantage is a relatively limited speed and memory capacity of the Logo used. Its advantage is simplicity. It is masterable during few laboratory sessions.

REFERENCES

[1] Diaz, M.: *Modelling and Analysis of Communication and Cooperatin Protocols Using Petri Net Based Models.* Computer Networks, no. 6, 419-441 (1982)

[2] Peterson, J.L.: *Petri net Theory and Modelling of Systems.* Prentice Hall Inc., Englewood Cliffs, N.J. (1981)

[3] Pooley, R.J.: *An Introduction to Programming in Simula.* Blackwell Scientific Publications, Oxford (1987)

[4] Sklenar, J.: Environment for Modelling of Petri Net Based Networks in the Logos Language. *Proceedings of the EUROLOGO'91 Conference*, Parma , 385-400, (1991)

[5] Starke, P.H.: *Petri Netze.* VEB Deutscher Verlag der Wissenschaften, Berlin (1980)

[6] Weinberger, J.: Logo, Simulation and Object-Oriented Programming. *Proceedings of the EUROLOGO'91 Conference*, Parma , 219-225, (1991)

AlgoBlock: a Tangible Programming Language - a Tool for Collaborative Learning -

Hideyuki Suzuki[1], Hiroshi Kato[2]

C&C Information Technology Research Laboratories,
NEC Corporation, Japan
Address: 1-1, Miyazaki 4-chome, Miyamae-ku,
Kawasaki, Kanagawa, 216, Japan
tel: +8-44-856-2317 fax: +8-44-856-2239
e-mail: [1] suzuki@joke.cl.nec.co.jp
[2] kato@joke.cl.nec.co.jp

Abstract

AlgoBlock is a tangible programming language for programming education at primary and secondary school. The system is a set of physical blocks that can be connected to each other manually to form a program. Those blocks are then connected to a computer to execute the program. Each block corresponds to a command of a Logo-like programming language. AlgoBlock is developed as a tool for promoting collaboration. It works as an open tool for collaborative activities, and helps learners build programs through social interactions including conversations. In this paper, the AlgoBlock system, its features as a conversational tool, and results of an observational session, in which three 12-year-old students engaged in collaborative activities, are described.

Keywords: collaborative learning, programming education, conversational tool, Logo-like environment

Hideyuki Suzuki *was born in Japan in 1963. He received B. A. in Psychology at Aoyama-gakuin University in 1986, and received M. A.. in Psychology at Keio University in 1988.. After graduation, he joined NEC Corporation. He is currently a researcher of Applied Information Technology Research Laboratory, C&C Information Technology Laboratories, NEC Corporation. His research interests include application of cognitive science to education systems.*

Hiroshi Kato *was born in Japan in 1959. He received B. E. and M. E. degrees all in Electronics Engineering from Keio University, Japan, in 1981 and 1983, respectively. After graduation, he joined NEC Corporation. He is currently an assistant manager of Applied Information Technology Research Laboratory, C&C Information Technology Laboratories, NEC Corporation. He had been a visiting researcher at CSLI, Stanford University, U.S.A. from 1989 to 1990. His research interests include application of cognitive science to education systems.*

AlgoBlock: a Tangible Programming Language - a Tool for Collaborative Learning -

1. Programming languages as conversational artifacts

The authors intend to design a learning environment in which learners at primary and secondary school can improve their skills for planning and building procedures through group work. Group work is essential for learning from the standpoint of situated learning theory [1][3] which considers that learning is a process of enculturation through social interactions.

In designing the learning environment, although social setting for authentic activities [1] is important, it is also important to design the tools for activities, because the structure of our activities is affected by the constraints of the tools in use. Here, the authors focus on the programming languages as a conversational artifact [4][5] which facilitates the learners' verbal interactions.

In our opinion, programming languages for collaborative learning ought to be designed according to the following principles: ***a) easy operation*:** Learners can operate the system easily. Easy operation contributes deeply to the promotion of collaborative activities for the following two reasons. 1) learners do not have to concentrate on how to use the tool, consequently they can be absorbed in group work, e. g., seeking solutions through discussion, building procedures and examining them collaboratively. 2) easy operation promotes trial-and-error activities, which draw conversations among learners. ***b) simultaneous accessing*** and ***c) mutual monitoring*:** all learners are equally eligible to refer to and manipulate the program whenever they want to. And all learners can monitor the process of the collaborative activity mutually. These features contribute to the activation of collaboration, because they promote learners' activities, such as displaying their own ideas to the others when they want to, understanding the others' operation to the program immediately, and intervening against the others' operations on the spot. ***d) natural turn-taking control*:** turn-taking to refer to and manipulate the program should be controlled, since unlimited simultaneous accessing to the representation by all participants inevitably brings out confusion and breakdown of the collaboration. Rules for acquiring control should be natural for learners, because obeying unusual rules prevent learners from collaborating smoothly.

Preceding research has pointed out the possibility of programming languages including Logo as a conversational tool. (e. g., [6]) However, those systems have innate limitations as a conversational tool. The limitations are mainly from constraints on the human-interface of the systems. Generally, the systems use the CRT as a display device; however, only a few viewers can face a CRT, because it is not designed to be shared by two or more users. Therefore, all learners cannot refer to the display simultaneously in collaborative work. This prevents learners from monitoring the process of the works. In addition, those systems employ keyboard or mouse, which allows only one learner at a time to use them to manipulate the programs on the CRT display. Thus, all learners cannot manipulate a program at the same time. Moreover, with these input devices, the turn-taking is controlled by handing over the devices among learners. This forces learners to obey an unnatural rule of turn-taking that is significantly different from the rule of everyday collaborative activity in which turn-taking is unconsciously controlled mainly by non-verbal cues. This limitation can be an obstacle to learners' interaction. Considering these limitations mentioned above, the existing programming languages are not completely suitable for collaborative learning. Based on these investigations, AlgoBlock was developed.

2. What is AlgoBlock ?

AlgoBlock (Fig.1), a Logo-like tangible programming language, is a set of physical blocks that can be connected to each other manually to form a program. Those blocks are then connected to a computer to

execute the program. Each block corresponds to a command of a programming language. Learners can operate a graphically displayed submarine on the CRT (Fig.2) through Logo-like commands, such as go forward, slide to left/right, turn, unfold the legs for landing, and so on. Each one of those commands has a corresponding physical block. In addition, control blocks, i.e., if-then-else and repeat-until are available for logical control. Some kinds of blocks have a parameter switch on them. The switch is used to set a parameter for a command. Learners make a program by connecting real blocks and setting parameter switches. The purpose of AlgoBlock is to allow learners in primary and secondary school to improve their skills for planning and building procedures through group activities.

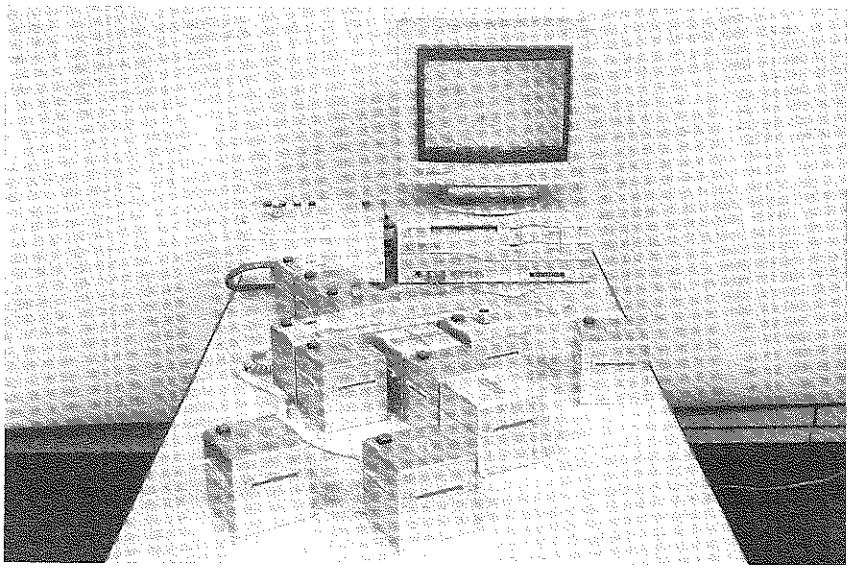


Fig.1 : A tangible programming language: AlgoBlock

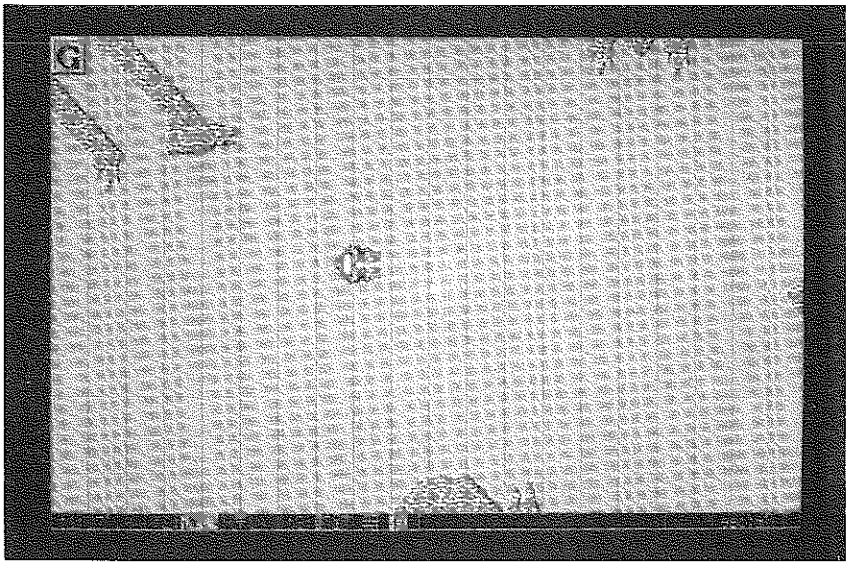


Fig.2 : Screen Design of AlgoBlock

3. AlgoBlock as a conversational artifact

AlgoBlock has been designed as a tool for collaborative learning according to the following principles which deeply support conversation. In other words, it is qualified as a conversational artifact.

a. Easy operation

With AlgoBlock, learners can make programs by handling tangible blocks. This way of making programs is much more direct, therefore, easier than inputting the program with a keyboard and an editor. Easy operation evolves learners in collaborative work, and promotes trial-and-error activities which generate conversation.

b. Simultaneous accessing

Learner's intentions are physically represented on the blocks, and placed on the space of collaborative activities. This tangible representation allows simultaneous pointing and manipulation by a group of learners, so all learners are equally eligible for accessing the program; i. e., AlgoBlock enables learners to share the representations of an algorithm.

c. Mutual monitoring

Manipulating AlgoBlock generates apparent gestures. The gestures indicate their intentions externally. This enables all learners to monitor each other's operations on the blocks throughout the collaborative work. This open interaction stimulates conversation among learners and facilitates collaboration. In other words, AlgoBlock works as an open tool [2], which enables users to monitor their cooperative activities.
Moreover, AlgoBlock shows the program execution dynamically. Each block has a lamp that flashes when the corresponding command is executed. Indication lamps enable learners to trace a programs. In addition, AlgoBlock has a debug mode in which learners can trace the program step by step, by just pushing a button on each block. These functions also help learners to monitor the process of activities.

d. Gesture based turn-taking control

As mentioned above, operating AlgoBlock naturally generates apparent gestures (Fig.3). For example, stretching out one's hand, turning one's body towards the blocks, and other gestures that are inevitably generated in the natural flow of operations on the tangible tool. Consequently, turn-taking of access is controlled by those gestures; e. g., learners get turns and interrupt others' operations by stretching their hands toward the blocks. This sort of control contributes to facilitation of collaborative activities. This is because, rules for acquiring control are equally the same as in everyday activities, thus learners never need to learn artificial and system specific rules. As a result, learners can cooperate unconsciously.



Fig.3 Collaborative work with AlgoBlock (adult)

4. Observation

One experimental session in which three 12-year-old female children were engaged in a group work with AlgoBlock for one hour was executed. The purpose of the session is to examine the effect of AlgoBlock on collaborative work. All subjects had no preceding experience in programming or manipulating computer based tools.

In the session, 3 tasks in which the subjects were required to move the submarine towards the goal were given after a 3-minutes introductory phase. In the introductory phase, they were instructed to observe the work with AlgoBlock which was executed by experimenters for a while.

Although two experimenters attended the session, they never intervened with the subjects' work with the exception of system trouble solution. Subjects' conversation and non-verbal actions were recorded throughout the session. Following is the result of the qualitative analysis of the protocol data.

Throughout the session, it is observed that the subjects made and changed programs easily. Although, it was the first experience for all subjects in manipulating programming language, they could operate the language without difficulty after about 3 minutes of the introduction phase. The following protocols show that AlgoBlock is easy to operate. All utterances in the protocols are originally in Japanese. The notation is also indicated below.

notation
CRT : description of CRT display (program execution)
Subx : Subject identification index
" " : Records of utterances
{} : Records of non-linguistic actions
() : Authors' comments
[] : Index for explanations

protocol 1
CRT : End of execution (submarine failed to reach the goal)
Sub2 : " This is wrong. " {assigned Right45 to the ROTATE block [action-1]}

protocol 2
CRT : End of execution (submarine collided with an obstacle)
Sub3 : " This {pointed a block} might be Left45. This direction. " {moved hand form the upper right to the lower left}
Sub1: " I see. " {changed a ROTATE block's parameter into Left45 [action-2]}

Subjects changed the program immediately after observing the execution. (e. g., [action-1], [action-2]) Easy operation, which is mainly from tangibleness of the tool, encourages learners' trial-and-error activities. Vigorous trial-and-error activities in collaborative work promote generating topics of conversation, thus the collaboration is activated.

protocol 3
Sub1 : " Hey. Is 2 okay? {pointed the parameter switch of the FORWARD block}
Sub2 : " Yah, go forward 2 steps. "
Sub3 : " 2 steps forward, I think so. {moved a FORWARD block to front of Sub2} Then? "
Sub2 : " To right. "
Sub1 : " Rotate, rotate 90 degrees. "
Sub3 : {connected the SLIDE-RIGHT block to one side of the FORWARD block}
Sub1 : " How many steps to right? "

Sub2 : " 2 steps! "
Sub1 : " Then, rotate. "
Sub2 : " Rotate 45 degrees. "
Sub1 : {connected the ROTATE block}
Sub2 : " And then, one, two, three. 3 steps. " {connected the FORWARD block (3 is assigned as the parameter)}

From *protocol 3*, it is observed that they collaborated by sharing a temporary and incomplete representation which was placed on the space of collaborative activities. They never built whole plans in their minds, but they refined their plans incrementally through conversations. In addition, all the subjects were equally eligible to manipulate and refer the tangible representation in the activities.

Protocol 4
Sub2[A]: " Has folded the legs (legs for landing), now it (the submarine) can slide. (booster for sliding right and left are available only when the legs are folded)" {pointed the LEG-FOLDING block [action-a1]}
Sub1[B]: " These two must be reversed." {turned body towards the blocks [action-b1] , and pointed the LEG-FOLDING block and the ROTATE block [action-b2]}
Sub2[C]: " No. This way. " {tried to change the parameter for the FORWARD block [action-c1], and prevented Sub1's pointing (against [action-b2]) by stretching elbow}
Sub1[D]: " No! First, we must make the legs fold, and then rotate 90 degrees. " {shouted, leaned towards the blocks [action-d1], and pointed the ROTATE block [action-d2]}
Sub3[E]: " I think so. "
Sub1[F]: " Because the legs aren't folded now. " {tried to make the blocks, i. e., LEG-FOLDING block and the ROTATE block, reversed [action-f1]}
Sub2[G]: " No. This must be (not clear) "
Sub1[H]: " I'm certain." {made the blocks reversed [action-h1]}

Manipulations on AlgoBlock generate apparent gestures, e. g., [action-c1], [action-f1]. This tendency might be much more noticeable than the situation in which the subjects use the keyboard or mouse for input. Utilization of tangible tools enable all subjects to monitor the collaborative work. For example, Sub1's [F] operation on the blocks are indicated clearly as obvious gesture [action-f1] to all subjects. Sub2 grasped the intention of Sub1 from Sub1's action, and made objection to the operation [G] immediately. This open interactions stimulated conversation among the subjects, and facilitated collaboration.

Protocol 4 also shows how turn-taking to access the program was controlled with AlgoBlock. Sub1 [B] tried to manipulate the program by turning her body towards the block s and reaching out a hand to the blocks [action-b1/action-b2]. Against Sub1, Sub2 prevented the action by swishing away Sub1's hand [action-c2]. After that, Sub1 leaned towards the blocks again [action-d1/action-d2], and finally succeeded in operating the program [action-h1]. Here, the turn was shifted by the subjects' actions on the program which are generated naturally in the activity. This gesture-based control facilitates collaboration by reasons mentioned in the preceding chapter.

Moreover, it was observed that the subjects used abridged expressions frequently. For instance, Sub1 [B] said "These two must be reversed" by pointing the LEG-FOLDING block and the ROTATE block directly [action-b2], instead of saying "How about reversing this LEG-FOLDING block and that ROTATE block". In this way, they expressed their intentions immediately and easily, which supported learners' conversation. These abridged expressions make sense when they set consciousness onto the same objects. AlgoBlock supports learners in the sharing of the orientation of consciousness because of its tangibleness which provides apparent targets of mutual consciousness-setting. In other words, AlgoBlock has competence to generate collaborative space successfully.

5. Conclusion

In the preceding chapters, the role of programming languages as conversational artifacts was emphasized, and requirements for conversational support were examined. Based on this investigation, the authors proposed the concept of AlgoBlock which is a tangible programming language for children, and mentioned its aspect as a conversational artifact. Results of observations were also presented. Although it is incomplete and more analysis is needed, the results show that AlgoBlock is qualified as a conversational artifact.

AlgoBlock is an adequate tool for conversation, however it is not the only solution for conversational support. The authors intend to propose principles for designing collaborative learning environment from aspect of tools as well as social settings. In this perspective, AlgoBlock is a first step of our research.

REFERENCES

[1] Brown, J. S., Collins, A. and Duguid, P. (1988):"Situated Cognition and the Culture of Learning", IRL Report No. IRL88-0008

[2] Hutchins, E. (1990): "The Technology of Team Navigation", Intellectual Teamwork, Galegher, Kraut and Egido (Eds.) pp.191-220, Erlbaum

[3] Lave, J., and Wenger, E. (1991):"Situated learning - Legitimate peripheral participation -" Cambridge University Press

[4] Pea, D.R., Sipusic, M. and Allen, S.(1990):"Seeing the light on optics: classroom- based research and development of a learning environment for conceptual change", Proceedings of 7th Tel Aviv Workshop on Human Development: Development and Learning

[5] Roschelle, J. (1990): "Designing for conversations", AAAI Symposium on Knowledge-Based Environments for Learning and Teaching (1990)

[6] Schibeci, R. A. (1988):"Logo in preservice and inservice teacher education", Proceedings of 6th Australian Computers In Education Conference

Preschoolers Logoworld

Stathis Triantafillou

Theotokos Foundation
Nea Liosia Attiki
131 22 Greece
tel: (301) 2311070, fax: (301) 2314888

Play & Learn Computers
Georganta 36 & Alonion
Kifissia - Greece
tel: (301) 8011369

Abstract

It is possible to construct a functional microworld for preschoolers. How the gap between concrete and abstract forms is bridged? What type of methodology should be followed in order to get the most from the program? Those are the questions which this paper tries to through some light, based on field - experience with preschoolers.

Keywords: Logo, developmental microwords, turtle peripherals

Stathis Triantafillou *is a B.A Psychology graduate of a Canadian University. He has been working in the field of Rehabilitative Technology for the past 10 years. His work specifically concerns software development and special peripherals for normal pre-schoolers & mentally and physically handicapped children. Work places: Theotokos Foundation, Social Work Foundation, Play & Learn Computers, Sakellariou-Kavaratzi School and Institute of Informatics Crete.*

Preschoolers Logoworld

Introduction

As Piaget has described the preoperational period (2 - 6) is a tie of transition. Is the time that the child, uses a mental picture of an object to represent it, perceives two things that happen together to have some causal relationship and is heavily influenced by her own wants and desires. Further more in the minds of this period reversibility and classification begin to operate.

Taking in consideration the characteristics of the pre operational period, its potentials and the potentials of Logo environment we developed a stimulating micro world which enables pre-schoolers to manipulate efficiently the turtle concept. Our seven years of experience are not concerning only the normal pre-schoolers (kindergarten "Play & Learn" Computers), but also the non-gifted pre-schoolers (Theotokos Foundation - mentally handicapped children) which are differentiated from the others by the extended preoperational period.

When we first applied onekey Logo to ages below 8 years old, we realised that modifications into the software of the program and the development of some peripherals in order to make the program accesible to preschoolers were inevitable.

The basic difficulty which the program sets on that age is the position of the turtle in relation to space. The desired prerequisite set by S. Papert ie, identification of the child 's own body with the turtle it was a hard accomplishing task. Another difficulty is the computing of distance, particularly the estimation in logosteps which does not offer experience with the real measuring in life situation. Further more floor turtle is quite, mute, slow and unable for adventurous motion. Pre-schoolers of our times and especially boys demand action, noise, flashing lights, speed and power. The poor turtle has to compete the 4 * 4 wheel drive and spacecrafts. Surely only the turtle has the fine move and the turn but is that enough to motivate pre-schoolers?

So, as anyone can see, apart from the methodological problems the program has to compete the massive computer & video games. One might differentiate correctly educational programs and games, however children 's perspective and expectations are quite different than the ones the adults usually promote. And it is obvious and inevitable that children will compare Logo and Games with their own criteria.

Confronting this "crisis" of our times the task of designing and applicating Logo becames more demanding and difficult. Therefore the goal is to develop microworlds which are understandable and stimulating to preschoolers. The first requires analytic, successive and hierarchical methodology while the second requires high motivational level, adventurous notion and entertainment.

With this scepticism in mind we designed a Logo environment for preschoolers based on two basic microworlds, the *floor-turtle microworld* and the *screen-turtle microworld*. Each microworld includes a different learning environment which is supported with additional commands, special peripherals and a methodological approach relative to cognitive capacities of the preschoolers.

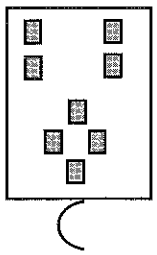
Our effort was to design microworlds which would be functional, interactive and joyfull. In other words we had to design an educational enabling technological microworlds.

Floor-Turtle Microworlds

At this stage the trainer has a heavy task to organise a highly stimulating micro world. Is the time which

the trainer has to become a creator and design the first introductory microworld. Few obstacles or a piece of paper are not enough to stimulate and motivate the child. I usually refer to my colleagues for this intoductory microworld as the "*Logo Disneyland*". For example if you are running the village project the microworld should be of areal village. It should include little houses, shops, playmobils, animals etc. All these should be available to children because at this stage they are called to operate as directors and actors. In other words they will develop a story (scenario), encouraged by the trainer, and will place the material according to the scenario and will interact with the basic hero the floor-turtle.

Ofcourse through the process due to the innitiative of the children the basic story will be modified but this is part of the play. The trainer has to be alert and promote indirectly various stimulating scenario 's which will justify also the methodological approach. The succesive introduction of the commands is paired developmentally with turtle free move, maze move, pen move and with the use of the **hand turtle**. Hand turtle is a simple peripheral, equipped with the four directional arrows plus four other commands. It is first used on the top of the floor turtle and secondly is placed in the hands of the child which directs the floor-turtle by walking directly behind her.



Hand Turtle: A peripheral equipped with light keys and is linked to the PC via an interface switch system (Inter 800).

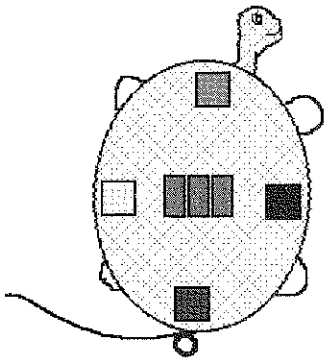
When the children exhibit learning behaviour in relation to position in space and to the particular command system, are ready for the next step. They are ready for the transition.

Screen-Turtle Microworld

The transition from the floor environment to the computer environment is always the hard step. Suddenly, the three dimensional stimuli is replaced with the two dimensional stimuli, the quite big floor turtle with a tiny screen turtle, and on the top, the complexity of the keyboard. *Lacking in the transformation of simulation, assimilation pays the price.* It seems that there is a gap between the floor turtle and the screen turtle, and this gap we will try to bridge by introducing the **desk turtle**. Desk turtle is a simple peripheral like the hand turtle, it is actually a small turtle equipped with the four directional arrows plus three other buttons. As well as with a little pen on the backedge.

Desk-turtle is located directly below the screen on the top of a hard paper. The child is instructed that the desk turtle is a kind of relative to the turtle on the screen and is directed by pressing the command he wants and pushing her towards the direction of the turn or the move.

The advantage here is that the child directs the desk turtle by coordinating his/her move with the one on the screen turtle. This is the time that the three dimention microworld is paired with the two dimention microworld in order to increase simulation and assimilation.



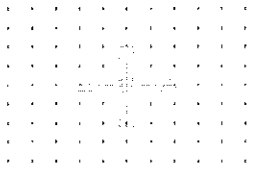
Desk Turtle: A peripheral equipped with four directional arrows plus three other buttons and a little pen on the backedge

The instruction here is under a slow pace so that the child is able to conceptualize the functional environment. Desk turtle here plays the role of the intermediate tutor. At a second stage the child may feel confident enough to direct the screen turtle without moving the desk turtle. When this takes place will be the right time to move to keyboard.

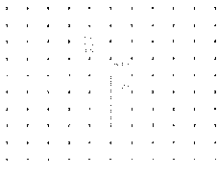
Apart from the contribution of the desk turtle few other supportive techniques could be very usefull:

Inclined Screen and Turtle move: If the monitor is placed horizontally on a 60º angle the commands forward / backward are not replaced with up / down. It helps move if the turtle move 1 cm, instead of Logosteps. This move initiates children into measurment as it is applied in life situations.

Grid Screen and Instant Compass: Tiny dots placed in each cm on the screen offer a functional visual template which enables children to estimate the total lenght of their side. A compass which reveals the four directions for 60 seconds is on the child's availability by pressing a single button (icon 3).



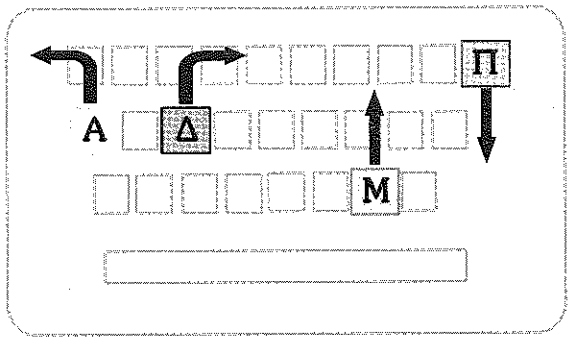
icon 3
Grid Screen and Compass



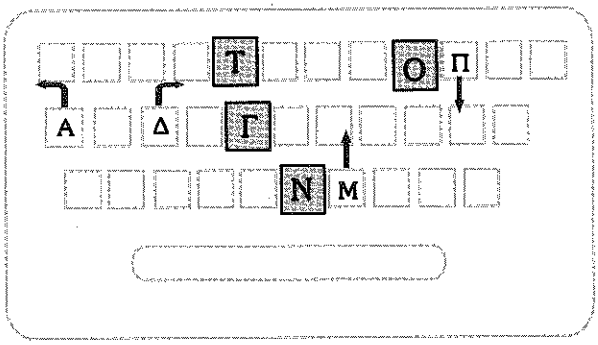
icon 4
Side and Angle Estimation

Side Estimation and Angle Estimation: When a line is completed, an indication about its length is placed besides it on the screen. So, the child is informed about the total length of the line.
When a turn is completed, an indication about the total degrees is placed on the screen. So, the child is informed about the total turn of the turtle (icon 4).

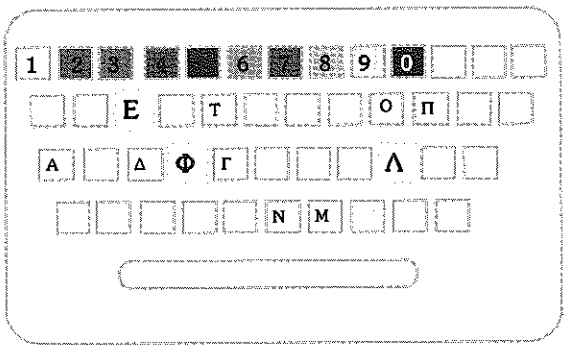
Hierarchical Menus: Hierarchically-based menus are necessary, in order to introduce successively the commands of LOGO. The Onekey commands are introduced successively in six menus. Colored stickers are used at the top of the keys, in order to enable discrimination and right key punch.



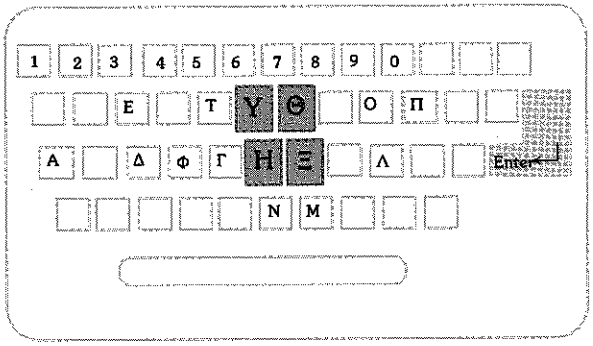
1. forward / backward / left / right



2. penup / pendown / penerase / circle

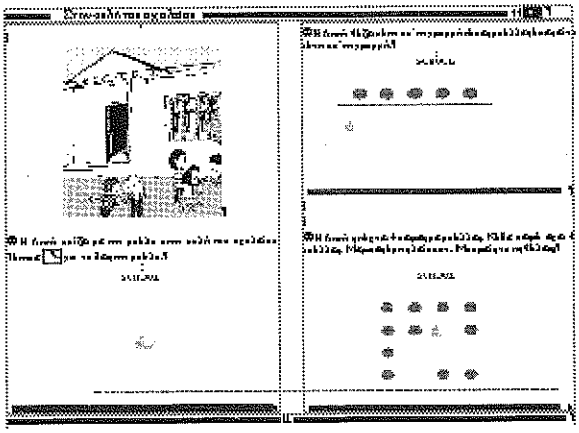


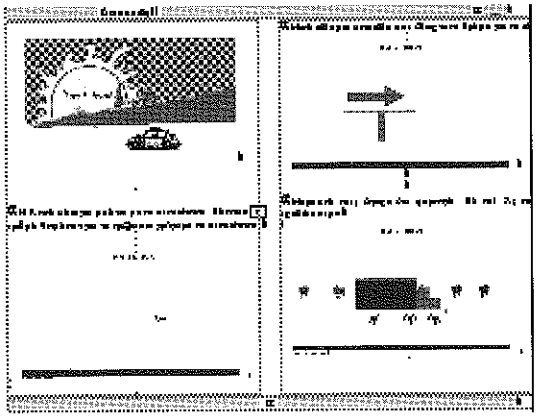
3. Pencolor, screencolor, fill, colors 1-0



4. Stamp, shape, clearscreen, fillshape, enter

Projects: Scenario is the keyword here as well. The Philosophy of the onekey Logo projects (in the language) was to pair the instruction of the commands with a stimulating and appealing stories. Leto is the name of the turtle and the child assists her from birth to school (Onekey - A series 31 work cards). Through obstacles and adventures the child directs Leto by practicing onekey commands.





DISCUSSION

Our effort was to design a dynamic Microworld with clear stimulants, joyful scenarios, challenging and easily manipulated by preschoolers. In the process, we discovered that the "traditional" Floor Turtle environment had to be supported, in order to be manipulative and understandable. Continuing our exploration, we found out that the transition from the Floor Turtle environment to the computing had to be bridged with an intermediate stage. In this stage, Onekey LOGO gets very specific and detailed, in order to match the cognitive potentials of the pre operational child. The peripheral, which was used here (Desk Turtle), increased stimulation and enabled the users to "see" that the tiny Screen turtle moves the same way as the Desk Turtle and as the Floor Turtle. In few words, the preschooler user interacts at various stages with various turtles and understands that all these turtles belong in the same family.

What we actually have done was to support the transition from the very concrete to less concrete by constructing Microworlds, which preschoolers could understand and interact with.
We have done that by translating the "cues", which are young users indirectly/directly have shown during their interaction with the Microworlds.

Aknowledgement

Special acknowledgement to Microelectronics Educational for the technical support and the development of the two basic peripherals (hand turtle - desk turtle).

BIBLIOGRAPHY

Center for Special Education Techology (editor) (1989), *Tech Use Guides - Using Computer Technology,* USA Department of Education.

Calabrese E. (1991), *Proceedings of the 3rd European Logo Conference*, EUROLOGO '93, Parma, Italy.

Kotsanis Y., Doukas C., eds (1991), *Proceedings of the 1st Conference on Educational Computing*, Athens, Doukas School

Lawler R. (1985), *Computer Experience and Cognitive Development*, Wiley, New York.

Piajet J. (1963), *The origins of intelligence in children*, Norton & Co., N. York.

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

Parallel Programming in Logo

Large Darko Zupanic

Jozef Stefan Institute
Computer Science Department
Artificial Intelligence Laboratory
Jamova 39, 61000 Ljubljana, Slovenia
tel: (+38)(61) 159 199, fax: (+38)(61) 161 029
e-mail: darko.zupanic@ijs.si

Abstract

One of the most powerful programming techniques is the parallel programming. This is a very useful technique especially when we are dealing with robots, toys, multiple turtles,... Since the parallelism is not supported by Logo itself, we've written a module which defines commands that make the parallel programming possible. The present paper indicates the implementation of this parallel programming module, describes the supporting commands, and gives an example of a semaphored crossway. The module was written in LogoS. It uses some LogoS dependent commands, but with a few changes it can be made suitable for other Logo interpreters.

Keywords: processes, synchronisation, parallel programming

Darko Zupanic *is a student of the Faculty for Electrical Engineering and Computer Science in Ljubljana. His main topic of interest on Jozef Stefan Institute is Machine Learning. He started to work with Logo four years ago when the Slovenian Logo project (LogoS) was started.*

Parallel Programming in Logo

1. Introduction

Because the parallelism is not supported by Logo [6] itself it is impossible to make for instance a Modula-2 [7] like parallelism. However, it is possible to make some kind of approximation, which is good enough to help resolving some problems.

In order to achieve this purpose it is necessary to define some new commands. These new commands are divided into three groups. The first group contains commands which handle processes, the second group takes care of the process synchronisation and the third group introduces the control commands.

In a real parallelism [5] (language implemented) the switching between the executing processes is automatic. This is commonly known as *time slicing* . If the switching is done quickly enough, the process appears to be running concurrently.

In the Logo parallelism we don't have an automatic determination of a time slice. The decision, what the one slice is, is the programmer's. The base unit of one time slice is a single command. Because this command can be a loop (which is usually time consuming), we have introduced some control commands.

2. Description of the parallel programming module

Every user command in Logo can become a process under certain conditions. First, the command body must be divided into slices; slices are syntactically separated with brackets. The second step is the initialisation of that process.

The initialisation of the process is in fact a transformation of a command into a list of slices and an inclusion of the process into the *schedule chain* .

The schedule chain contains the following data about each process: the list of slices, the currently active slice, the process state and the process turtle.

The important part of the parallel programming is of course the synchronisation, which takes all the care of the coordination between processes. This aim was achieved with using the *signals* through the process *state* .

The execution of processes and the switching between them is very simple. We just observe which process's turn is it, then check its state and according to the state the active slice is *Run* .

All the processes are independent in the sense of inheritance and they all run at same priority.

2.1 Processes

The central part is the process scheduler. When you make a call to the *RunProcesses* , you start the process scheduler. During the execution each process is in one of the following states:

- *Running* - this is the ordinary state. A process is waiting for its turn and when the scheduler switches to it, one slice is executed.
- *Suspend* - the process is suspended. The scheduler 'ignores' it.
- *SignalNm* - the process is waiting for a signal with the appropriate name.

The scheduler is stopped when the last active process is terminated.

The commands dealing with processes are:

InitProcess ProcessNm

Description: The command *InitProcess* prepares the body of the procedure *ProcessNm*, that means it resolves *labels*, and includes the procedure in the schedule chain. Thus procedure becomes a process with the state *Running*. The process is also given its identification number.

Parameters: *ProcessNm* is a word used to denote a command.

Returned value: A number identifying the process. This number is used as a parameter to some other commands.

KillProcess ProcessId

Description: The command *KillProcess* terminates the process identified by *ProcessId*.

Parameters: *ProcessId* is a number identifying the process to be killed.

SuspendProcess ProcessId

Description: The command *SuspendProcess* suspends the process identified by *ProcessId*. The process will not be scheduled again until it is resumed by the command *ResumeProcess*.

Parameters: *ProcessId* is a number identifying the process to be suspended.

ResumeProcess ProcessId

Description: The command *ResumeProcess* resumes the process identified by *ProcessId*. The process can again be scheduled.

Parameters: *ProcessId* is a number identifying the process to be resumed.

GetProcessId

Description: The command *GetProcessId* returns the current process's identification. Thus it makes possible to handle the process dependent matters.

Returned value: A number identifying the process. If the current process is just being killed proceeding, zero value is returned.

SetProcessTr TrNum

Description: The command *SetProcessTr* assigns a turtle with the identification *TrNum* to the current process. Each turtle command in the process then refers to this turtle.

Parameters: *TrNum* is a number identifying the turtle.

2.2 Synchronisation

The synchronisation of processes is provided by using the concept of a *Signal*. A process can *Wait* until another process *Sends* a signal.

When a process sends a signal, the first of the processes currently waiting for the signal is allowed to continue. There is a *waiting queue* which contains all waiting processes' data. The process that has been waiting the longest is resumed when a signal is sent. If there is no process waiting for a signal when it is sent, the signal is inserted into the *signal queue*. For each time the signal is sent, one process that waits is allowed to continue immediately.

The signal type is determined by a name, so you can have an arbitrary number of synchronisations at a time.

The commands for the synchronisation are:

SendSignal SignalNm

Description: The command *SendSignal* sends the signal *SignalNm*.
Parameters: *SignalNm* is a word denoting the signal to send.

WaitSignal SignalNm

Description: The command *WaitSignal* causes a process to wait for the signal *SignalNm*.
Parameters: *SignalNm* is a word denoting the signal to wait for.

AwaitedSignal SignalNm

Description: The command *AwaitedSignal* checks whether a process is waiting for the signal *SignalNm.*
Parameters: *SignalNm* is a word denoting the signal to check on.
Returned value: The word *True* if one or more processes is waiting for the signal *SignalNm*, and the word *False* otherwise.

2.3 Control

A command is base unit of a time slice. That also includes very complicated and time consuming commands. The programmer is able to prepare (cut up in slices) any of the user commands in order to make a process. The problem is with the primitive control commands such as loop and decision commands.

We've solved this problem with introducing two new control commands: *Label* and *GoToLabel.* These two command make possible to replace the critical primitive control commands.

The commands *Label* are resolved within the command *InitProcess*, while the command *GoToLabel* is an usual one (it is interpreted each time it is called).

Label LabelNm

Description: The command *Label* marks the position in the process in order to allow you to redirect the execution with the command *GoToLabel*. The command *Label* must be the only command in a time slice.
Parameters: *LabelNm* is a word denoting the marked position.

GoToLabel LabelNm

Description: The command *GoToLabel* directs the execution of the process to the slice that

immediately follows the slice containing the command *Label*. If there are any other commands after the command *GoToLabel* in that same slice, they will be executed as well.

Parameters: *LabelNm* is a word denoting the label to jump on.

3. Example

The following example is a typical problem which can be solved efficiently using the parallelism. LogoS [1, 4, 2, 3] supports eight turtles, so we may have at most eight vehicles. Each vehicle is to drive through the semaphored crossway. One possible situation is shown on figure 1.

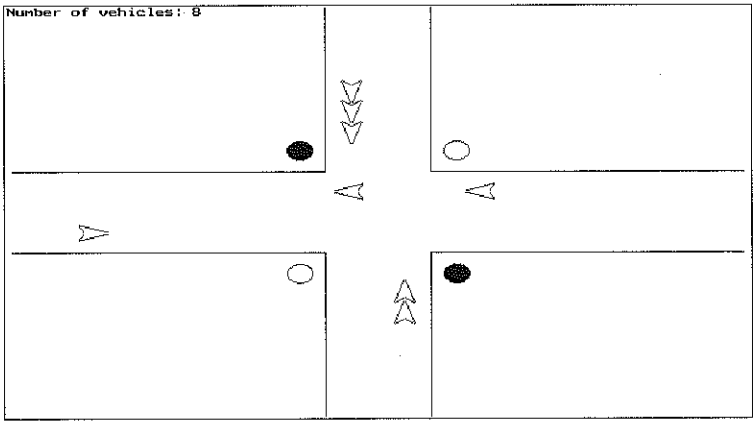


Figure 1: Semaphored crossway situation

The problem is divided into two independent processes. One process controls the sema-phore, and the other controls the vehicles.

Here is the source code of the semaphore process:

```
To DriveSemaphores
  [ Label "Drive ]
  [ GreenLightSignals
    Make "SemaphoreCnt :SemaphoreCnt - 1
    If :SemaphoreCnt > 0 [ GoToLabel "Drive ][] ]
  [ Label "Delay ]
  [ Make "SemaphoreCnt :SemaphoreCnt + 1
    If :SemaphoreCnt < :SemaphoreDelay [
      GoToLabel "Delay
    ][
      SetSemaphores
      GoToLabel "Drive
    ]]
End
```

Actually, this process has only two slices, because all the *Label* commands are solved and removed in the initialisation of the process. The first slice is a loop, which is repeated until the time allowed for the green light to be on is exceeded. The command *GreenLightSignals* sends a signal to all the vehicles that want to cross the crossway, if the green light is on. When the time for the green light is over all the lights become red. The next slice takes care of the vehicles that were in the middle of the crossway at the moment when red lights were turned on. After the *SemaphoreDelay* is reached, the same procedure with the other direction is repeated.

The only task of the vehicles' control process is checking the number of vehicles.

```
To CheckVehicleNum
  [ Label "Check ]
  [ If :NumOfVehicles < :MxVehicles  { GenVehicle } {}
    GoToLabel "Check ]
```

If the number of vehicles is less than the maximal allowed number of vehicles, the process executes the command *GenVehicle* , which generates with certain probability a new process *DriveVehicle* .

```
To DriveVehicle
  [ SetProcessTr GetProcessId - 2 ]
  [ InitVehicle ]
  [ Label "Move ]
  [ MoveVehicle
    GoToLabel "Move ]
End
```

First we assign a turtle to the process, then we initialize a vehicle and move it over the crossway.

When a vehicle arrives to the traffic light it waits for a signal. If the light is green, it receives the signal immediately (as described in the semaphore control process), otherwise it waits until the lights are exchanged. The vehicle then moves to a special position, which causes the process to kill itself.

All together, we have running two constantly present processes and up to eight temporarily present processes.

4. Conclusion

We present, that some problems with parallel nature, which are quite complicated to solve using only usual techniques, can be solved simply with the use of some additional commands.

The parallel programming techniques in Logo may turn out to become very important as Logo is mostly intended to be used by children and they are especially enjoying themselves when using Logo in connection with toys, robots, LEGO, etc.

REFERENCES

[1] Batagelj, V.: *LOGO-S, conceptual outline* , Ljubljana, October 1989, (in Slovene)

[2] Batagelj, V., Grobelnik, M. and Zupanic, D. (1991) The design of LogoS, *Third European Logo Conference Proceedings* , pp. 287-296, Parma, Italy.

[3] Grobelnik, M. and Zupanic, D. (1991) Inside LogoS implementation, *Third European Logo Conference Proceedings* , pp. 297-311, Parma, Italy.

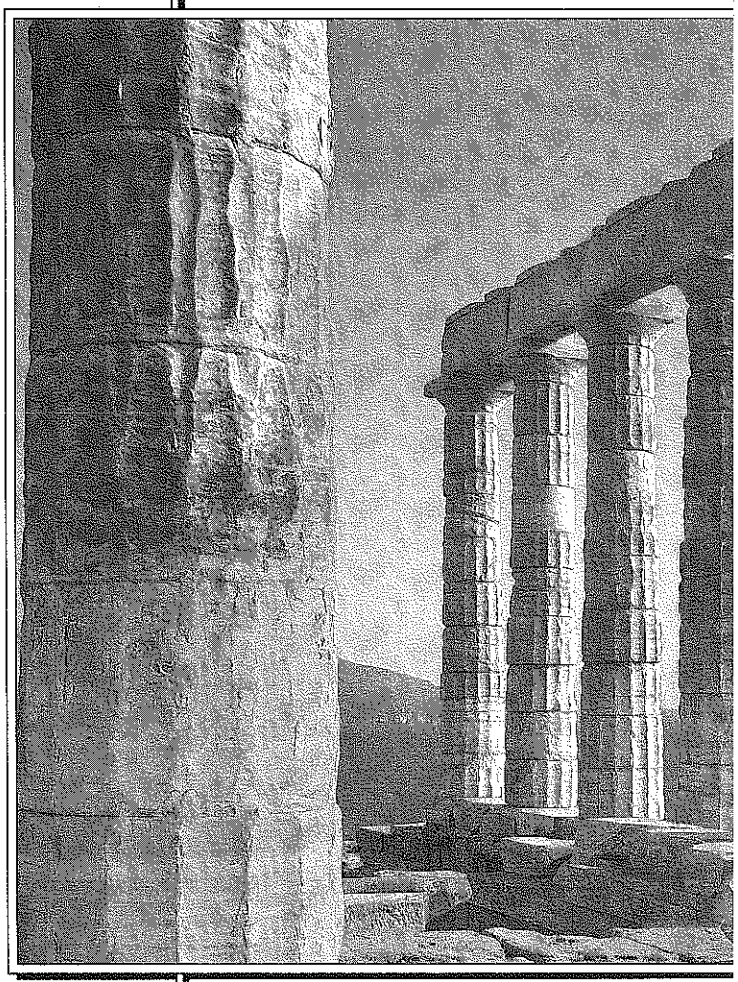
[4] Grobelnik, M., Zupanic, D.: *LOGO-S, version 1.0; technical documentation* . Ljubljana, January 1991, (in Slovene)

[5] Hoare, C., A., R.: *Communicating sequential processes* . Prentice-Hall, 1985

[6] Ross, P.: *Logo Programming for the IBM-PC* . Addison-Wesley, 1986

[7] *Language and Runtime Library, Reference* . Stony Brook Software, 1989, 275-287

Demonstrations



DEMONSTRATIONS

319 • *Enkenberg J., Lakotieva A.*
Modelling-Based Learning in Lego-Logo Environment

321 • *Morato P.*
Logo Environment & the Development of Spatial Cognition

325 • *Papadopoulou D, Triantafillou S.*
Logo for Motor Disabled Children

327 • *Sendov B.*
Geometric Exploration through Visual Modelling

329 • *Triantafillou E., Koulouris C., Kotsanis Y., Marouka H.*
The Development and Testing of a Greek Educational Environment ...

Modeling-based Learning in Lego-Logo Environment

Jorma Enkenberg, Aimo Lakotieva

University of Joensuu, Finland

Keyconcepts

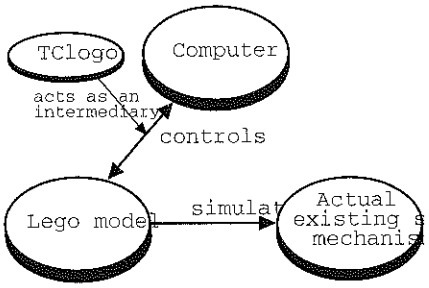
Cognitive apprenticeship, constructionism, design, complex problem solving, practical reasoning, knowledge representation, modeling, situation graph

Theoretical background

The purpose of the study has been to promote the mediation of the thinking skills and heuristic processes of an expert in working in a complex technology environment. The objects of learning have been auto existing servo mechanisms, the structure and function of which have been simulated by the means of LEGOTClogo. In order to organize the learning process the central starting points of the theory of cognitive apprenticeship and the constructionistic conception of learning have been applied.

Goals for the development and research work have been:
a) to develop a method based on cognitive apprenticeship learning theory for learning of control technology and
b) to develop tools for representing student's mental models about the auto existing servo mechanisms so that they could be used in describing, organizing and reorganizing of students' conceprtions so that they help the plannning and its implementation during model construction and programming.

Learning enivironment

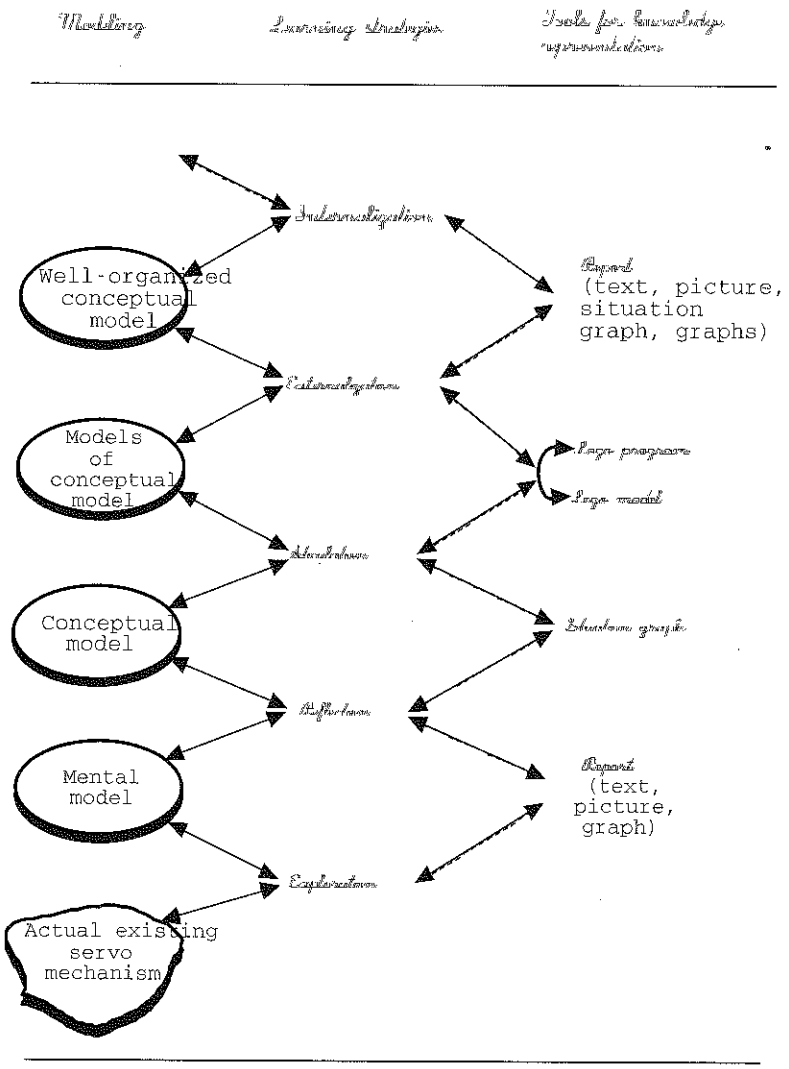


From theory to practice

Three different kind of models for the project work have been developed through research. Main emphasis in supporting of students' learning processes has veen in spontaneous investigations, representation of knowledge in different forms, construction and programming the model.

The main tool for organizing of students' knowledge structure has been the situation graph in the form of tree structure.

Phases of modeling, learning strategies and tools for knowledge representation



Logo Environment and the Development of Spatial Cognition in Children with Trisomy 21

Parrot Morato

Technical University of Lisbon
Portugal

Abstract: In this presentation we'll broach the specificity of the development and learning of children with Trisomy 21 (T21-"Down Syndrome"), at light of Morss' theory of developmental difference (1983). Beginning with Piaget's sensorium-motor theory about the genesis of spatial representation and with the recent studys of Pick & Lockman (1981), Newcombe (1982), Acredolo (1985), Cohen (1985), Spencer, Blades & Morsley (1989), about the role of children's active experience (manipulation and locomotion) in the development of spatial cognition, we'll point out the importance of the non-verbal nature of this assembly as a pertinent hypothesis to the study of the development and learning of space of children with mental deficiency/T21. Due to the scarcity of literature about the acquisition of spatial concepts by children with T21, we accomplished two studys with the purpose to contribute to one methodology of educational intervention. This study framed the matching of two groups of children with T21 phenotype (N= 9+9), on a educative context, with the purpose to analyse the effects of different Environments of Learning, one Conventional, centered on the visual resolution of problem situations (paper and pencil), the other one Experimental, centered on the resolution of situations by the actions of experimentation and concretion on the physical space. Trying to validate empirical experiences on the utilization of Information and Comunication Technologies in Special Education we used, on the Experimental Program, a robotic application of LOGO language, because of its implications in development of spatial cognition.

Introduction

We may say that people with trisomy 21 (T21-"Down syndrome") are virtually always mental handicapped to some degree. It is true, but to be realistic, T21 children should be viewed as children with certain limitations but who, given the right kind of special education, could reach a stage of development which would enable them to have a semi-independent existence and to enjoy a reasonable life-style within their limitations (Stratford, 1985).

Why LOGO with T21 Children? If Logo is a language for learning how to think and children with T21 have a poor and complex mental function, what is the reason than we think that LOGO can be useful for those people.

With Stenberg (1987), We must be more persistents in the study of qualitatives aspects of mental development, because mental deficiency after Inhelder (1963), result a human stage of development stoped on concrete operational thinking.

Children with T21 have long histories of learning failure, are easily distractible, lack confidence in their learning abilities, show low self-estem, and most important, exhibit a caotic approach to problem solving. During primary-school years (age five to ten) the average child with T21 remains very concrete in his thinking, and understands things in a literal sense. He understands everything around him in the light of a internal model of the world, without modifying this on basis of his experiences (Gibson, 1978). According to piagetian theory, it is primarily trough active involvement with stimuli that a child knowledge advances through the stages of cognitive development, this action dimension assumes particular importance for the understanding how children gather information about spatial domain, and his awareness the every day situations.

Spatial cognition is a domain where the child think about the relation and distribution of things. We can see this fact in everyday situation of involvment where the concepts of positon and orientation from own body to the objects is a natural way to think about.

The problem is how children with mental deficiency can manage spatial concepts beginning their own body active experience. Positional concepts like front and back, orientation concepts like left and right are the primarily of a cognitive conflict that children have to think.

One way to rectify this lack of conceptual clarity consists of examination the children's problem-solving skills with LOGO by a robotic application. Our understanding about mental deficiency depends of experiences where the child conflict with himself about the way to do something.

Spatial cognition is a variable or a component of human inteligence that has not yet been sistematically examined in the context of the development of cognitive-mapping skills even in those recent studies in which observations of children behaving in real-life spaces have replaced map drawing with paper and pencil spatial measures Spencer et. al (1989).

Indeed, it would seem reasonable to assume that such behavioural measures would be particulary influenced by the mode of exploration (active versus passive exploration).

However much of what little information we have from such studies seems from procedures in which the child has been limited to passive movememt through the experimental space.

The question about active exploration and concretion in the space and his representation is a very good way to study problem-solving skills in children with intelectual disabilitty like T21 children.

According with Morss' theory of developmental diference it is necessary to understand how children with trisomy 21 learn; what kind of errors they make; what kind of strategies they employ in their learning Morss (1985).We do need to look for reasons why children with T21 behave in the way they do in their learning strategies.

According Piaget, spatial cognition beginning on self experience with own body and the objects into a graphic space. It is the development from action to representation or from concrete space to abstract. According Acredolo and Piaget self directed activity serves to encrease attention of preoperational children to relevant topological cues in the environment.

To explore this subject of learning the environment LOGO by robotic application (Jeulin T3) is a way very concrete to promote the mobilization of the cognitive capabilities. The command by infra-red signals from the communicator are converted into moves, turns and pen action. This command by perforate card is a pedagogic way to put child in charge, without any difficulty about the normal computer keyboard. There is not any problems with the simbolism (numbers and letters), because the perforate cards contents the order by a simbol like an arrow (little or big) to forward or backward.

The Robot is easy to use. So children do not think in abstract terms, and because for a child with T21 think in abstract is normal very hard, they can understand things because they touch and see the robot himself in the space. The child with T21 can learn with robot the changes in the space by to think with moves. After some experiences with the robot, the child must think what he want to do before the command the robot, than the result show what he thought and what is the reality of his thinking.

We know that motivation is believed to be a powerful variable affecting learning. With robotic application we assume that motivation plays a very important role in learning children with intelectual disability because active participation like learning by discovery is believed to be a more meaningful type of learning than other types. We think that LOGO robotic application can help children because children learn to organize concepts and knowledge. Since 1989, we begin a study about development spatial concepts with children

to 5-10 years by a experimental program to develop spatial concepts by metacognition intervention using a LOGO learning environment by robotic application.

We can see in video some aspects of what LOGO can do for children with T21 in his learning of spatial concepts.

Results

From the analyse of the Experimental Program, we verified that it produced statistically significant effects (Measures front/back pre-post T= 5.06 to p=.001 and right/left T= 3.16 to p=.003) on the study group liable to its practice, in opposition to the Conventional Program that didn't produce statistically significant effects (Measures front/back pre-post T= 1.26 to p=.242 and right/left T= 1.58 to p=1.54).

From the comparison between the effects of the Learning Environments, we must conclude that both caused gains of learning without significant differences (Measures front/back pre-post F= 3.96 to p=.063 and right/left F=2.67 to p=.109).

However, the gains of learning of the study group liable to the Experimental Environment, revealed superior with a difference tending to the critic value p=.05. This results suggest that the utilization of LOGO robotics in the Experimental Program promoted the mobilization of the cognitive capabilities, by the relation of corporal concretion inherent to its manipulation, proportioning one environment of operative learning decisive to the development of the children involved.

Video

Title: "LOGO environment and spatial cognition in children with T21".
Duration: 7 min. color, produced by Sofia Fonseca and Pedro Parrot Morato

Logo for Motor Disabled Children

D. Papadopoulou, S. Triantafillou

Chatzipaterion - Criton, Microelectronics Educational (Greece)

Abstract: In order to make Logo accessible to children with motor disabilities a special system was developed utilising switches via an interface and relative software. A small group of children with motor disabilities was introduced into Greek LogoWr projects and the system enabled the users significantly to interact with the program. It seems that the system worked positively for the motor disabled children, however the experiment also indicated which way the system could be modified in order to work for bigger sample.

Introduction

Motion sets the cues of learning. Children who are deprived of motion have problems in the acquisition of learning. Cause and effect relationships are not clear due to the absence of experience. Logo offers concrete experience on cause and effect relationships and above all it offers a microworld environment which stimulates positive learning behaviour and responses.

Our intention was to organise an operating system which would enable children with motor disabilities to interact with the environment of Logo. Obviously the keyboard is not the ideal way interaction for the child with motor problems. In addition, because many times these particular children have learning defects the peripheral which is going to work should be functional and simple. That is why we have decided to work with simple switches and inside the onekey environment.

Of course this is a pilot project which lasted 3 months (April to June) and aimed to evaluate the functionality of the operation switch system in relation to one key environment and with the use of Greek Logo projects. From the feedback one might set the grounds for future research in this particular area.

The research took place in the Social Work Foundation, into the centre for the rehabilitation of spastic children (RCSC) and particularly in the Micro Lab which operates since November of 1993.

Subjects

Subjects were 5 children with motor disabilities (4 boys, 1 girl), ranging in age from 4 1/2 to 7 years.

Apparatus

Seven switches were connected via the Inter 800[1]. Each switch corresponded to a command from the onekey environment: F (Forward), B (Back), L (Left), R (Right), D (Pen Down), U (Pen Up), E (Pen Erase). Every switch was identified by a special graphic that was placed on the top, i.e. Left = a turn on its left turn, U = penup an air plain etc.

Procedure

A. Take in consideration the medical report or any other report relevant.
B. The child 's introduction and evaluation by the therapist in the Micro Workshop takes in consideration the following categories through clinical observation.

1) Motor abilities
2) Perceptual abilities
3) Sensory abilities and disabilities
4) Communication capacity
5) Self-care abilities
6) The motor and conceptual abilities lacked by the child are estimated and special treatment is used to develop them.

C. Planning out a therapeutical programme.

1) Therapeutical and rehabilitational aims and selection of the most appropriate technique.
2) Decide on the right work position
3) The peripheral needed and the suited place for the child to use it (head, hand, etc.).
4) Selection of the training programme.

D. The programme.

When the above procedures are completed the child is then introduced to Logo, as follows :

- The child takes the right work place
- Gets familiar with what we expect from he/she.
- We explain the usage of each switch.
- The child looks at the work cards while we narrate the adventures of Leto[1], the turtle.

The child has the opportunity to have a clear view of its own work, learns gradually the onekey commands and tries to accomplish the drawings suggested by the work cards, on the screen. When we introduce a new work card, which may be placed on a book-stand, we read the titles and the text and we ask him/her to execute the command.

The therapist observes:

- The comprehension of the command
- The work out of the command
- The understanding of the four directional commands
- The time needed to finish the project
- The feelings caused
- The degree of concentration
- If the project was up to his/her capacity

Considering the conclusions of this observation we intend to find solutions by simplifying the process up to the point we succeed with our original goal. We can also create new "exercises" to help solve the problems present.

Our primary scope in this project was:

- To make the child direct the turtle in the screen as himself in three dimensional space.
- Be able to follow the right direction
- To co-ordinate movement with the eye and hand
- To control body movements
- To learn to concentrate and program his/her work.

Results - Discussion

The reaction of children interacting with the system seemed to be positive. It was probably the first time that the children were able to control the educational process and to execute a design simply and accurately. Their effort was obvious in trying to direct "their" turtle in such a way in order to meet the demands of the design.

The projects worked positively and were easily understood. The adventures of Leto, although the assigned projects were only 5, stimulated motivation and creativity. The pictures or the symbols which were placed on the top of switches increased command understanding.

The application of the switch system supported a lot the operation of the program. In contrast with the keyboard, hand movement was more efficient and the useless moves were minimised.

The high sensibility of the switches created, at times, some problems in controlling the turtle. Children with no use of their hands were unable to operate more than four switches. The number of switches was a barrier due to the design of the system. However, we wanted to evaluate the response of the children to the program in order to go ahead with a more sophisticated design. What we have in mind, is a visual menu of the commands on the screen which will operate with two switches, one for moving the lighted frame and the other for choosing the command.

Technical Support : *Microelectronics Educational*

1 Inter-800 is an 8-switch PC interface and is a trademark of Microelectronics Educational, Athens

2 "Adventures of Leto" is a Logo Educational environment and is a trademark of Microelectronics Educational, Athens.

Geometric Exploration through Visual Modelling

Bojidar Sendov

Faculty of Mathematics and Informatics
Sofia University
Sofia 1126, Bulgaria

The goal of the demonstration will be to suggest worthwhile mathematical activities originating from real-world problem situations to be performed in GEOMLAND - a Logo-based environment of a laboratory type. The demonstration will cover such mathematical topics as:

- Geometric Constructions
- Loci
- Envelopes
- Extremal Geometric Problems
- Geometric Transformation

The focus will be put on showing appropriate situations for the mathematics classroom in which students might be involved in activities enabling them to show their creativity in discovering and exploring mathematical ideas.

The solving of the problems to be demostrated follows the natural stages of an investigation:

- verbalization of the problem;
- creating a mathematical model in GEOMLAND;
- explorations with the model;
- looking for patterns and interesting properties of the model;
- making conjectures, gathering evidence and building arguments to support them.

The teacher's role (which is not any more to dispensee "truth" but rather to help and guide the students when making observations, searching for patterns, formulating and verifying conjectures) will be discussed on the basis of an educational experiment in some Bulgarian schools.

The Development and Testing of a Greek Educational Environment Based on LogoWriter

S. Triantafillou, C. Koulouris, J. Kotsanis, E. Marouka

GREEK LOGO USERS GROUP

Microelectronics Educational
Athens, Greece

Abstract: The aim of the project is to make a story board approach to a developmental cognitive schema for the introduction of pupils to the world of LOGO. The schema is analysed in a series of work cards, that are interrelated and have a dual target of progressing the theme, as well as introducing new commands.

To implement this project, a Greek One-key system was devised to help "Leto", the turtle, in her adventures. The "Adventures of Leto" are currently tested on 5-7 year olds. The feedback is positive and will significantly assist in the formation of the final version. Projections suggest that this system will prove highly effective in teaching children the methodology needed for an initiation into LOGO worlds.

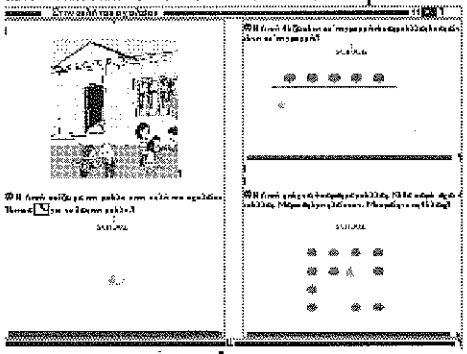
Introduction

Having considered the necessity to introduce pupils to Logo and assist them in their first attempts to create designs, a whole environment was developed using work cards, software and various support cards.

The effort was to make Logo part of a unified educational framework with clearly defined aims and to analyse it in substantial didactic units. These units had to lead to primary, direct scopes, given hierarchically, but also presented in such a way to combine with other scholar or / and social experiences of the pupils so that the entire environment would be adventurous and friendly. Each scope was analysed in one or more short "stories" in the work cards. Further more each "story" causes a meaningful "problem situation", and requires from the child certain kinds of abilities and cognitive skills for producing a satisfactory response. The purpose was to make "stories" with the response needed, embodied in their own form.

The out come was a series of 31 work cards. Each one plays a unique role in an ongoing scenario that correlates them all by narrating the story and adventures of Leto (the name of the turtle in the leading role). Children accompany Leto from her birth and first journey in the nearest sea and islands to a village, in school, at Christmas holidays, to her first party and finally to summer holidays. Work cards also present successive onekey commands in such a way that pupils can not complete the suggested projects without using them.

Work cards contain pictures and text (icon 1). There are two kinds of pictures. The upper left picture plays a decorative role and is related to the progress of the story. The other pictures are screen images from LogoWriter, same as the ones the child sees on the computer, but also representing the desired results. Text consists of small phrases describing the "intentions" of Leto and simple instructions on how to fulfil them.



icon 1

In order to make the whole project even more simple, all onekey commands were separated in smaller groups of the same type and assignment (ie group of turtle movement: M, Π, A, Δ, group of pen commands: N, T, Γ, etc). Work cards introducing commands of the same group have a common colour in the indicating ribbon that surrounds them (pen commands - blue ribbon etc). (icon 1)

Field Trial

Test Population

The subjects were 32 children, pupils of second grade elementary; eleven girls and twenty-one boys.

Method

Due to their big number the subjects were divided in two groups. The Tuesday 's group that numbered 15 pupils and the Thursday 's group of 17 pupils. Each group was distributed in the 11 computer systems of the lab. Furthermore each system accepted one or two children of the same Academic performance that was classified by A, B, or C level. Class Academic profile contained 16 A 's, 10 B 's and 6 C 's. (Figure drawn from monthly reports). The children were coming once a week and each session lasted for 25 to 30 minutes. Within each session we were able to run one or two projects the most. Each session could be described as follows:

At the beginning the children were asked who was going to read the story from the work card. The child which was selected read the story loudly and the echer encouraged the children to go ahead with the story if they had no questions to ask. Enough time was given for the completion of the story which was followed by saving the file and evaluating the work, by giving A for successful completion, B for partial completion and C for completion with mistakes. At the beginning of every session the trainer requested from the class the commands which were taught the previous weeks. After this short overview additional help was offered in its minimum, due to the evaluative feature of the research.

Results

Among the 32 pupils, the scores were 31 A 's with analytic grading ranging from 9.000 to 9.96 and 1 B 8.93. Those were the average scores drawn from the 12 introductory projects which were applied for a period of 3 months (March to May). This score is much better than the score of the Academic performance in the school lessons, which contained 16 A 's, 10 B ;s and 6 C ;s. (See comparative scores, Appendix A).

Discussion

Perhaps the most dynamic contribution of the research was that it constructed a stimulating computing environment, which allowed young children to be able to interact with a microworld relative to their cognitive abilities. Influenced by the massive computer games of our times, grade B children were more than ready to interact with computers positively. With this psychological background our users entered Leto 's microworld.

Children's comments on Leto 's projects seem to justify their design, meaning that they found the work cards pleasurable, easy and adventurous. Some comments were: "How nice it is to play with Leto", "Leto goes to the city! And goes to school too!", "To cross the river? That 's easy", and many more. Of course they met some difficulties when the demands for designing were increased, i.e. work cards 11 and 12, but by that time were highly persistive and would not give up.

Children who were sitting together and in their academic performance had A 's, worked less collaboratively than children who had B 's and C 's. Obviously collaboration exists when there is a need and personality traits that permit such a behaviour.

Keyboard discrimination did not create any problems. The children easily recognised the Greek characters on the keyboard and quickly learned the "instant" press of the computer key.

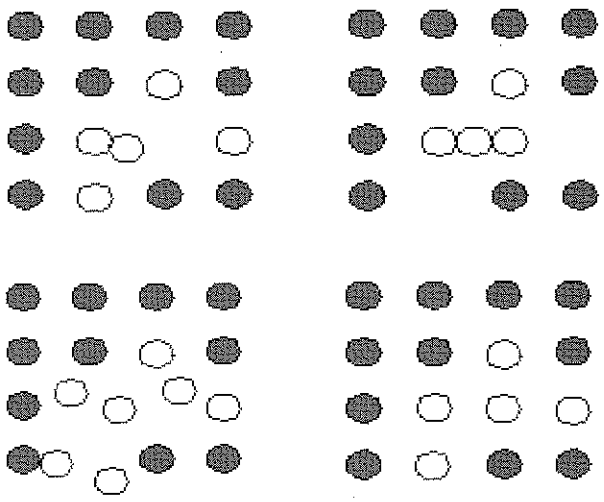
The research was beneficiary to the projects as well. Quality of questioning, adjustments of colours, appearance and positioning of shapes, hierarchical chaining and commands elaboration, were the areas tested and improved significantly.

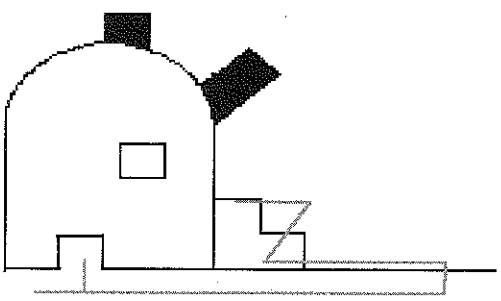
Above all the children indicated directly and indirectly that they understood Leto 's adventures and enjoyed the process. This successive methodological schema for commands seems to be functional and further research may support the present findings.

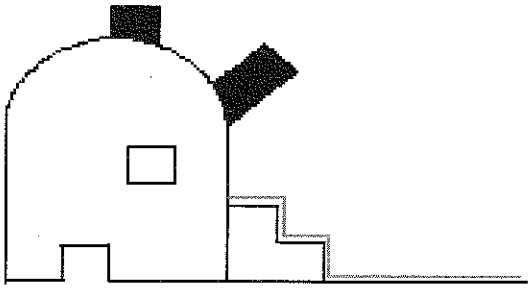
Appendix A

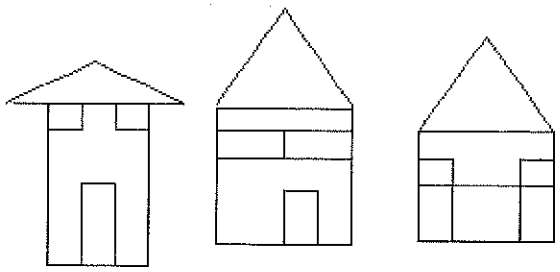
Score of Academic performance	Precise score in the projects	Score in the projects
A	9,73	A
A	9,86	A
A	9,65	A
A	9,20	A
A	9,32	A
A	9,79	A
A	9,96	A
A	9,93	A
A	9,48	A
B	9,37	A
B	9,83	A
B	9,41	A
B	9,61	A
B	9,79	A
B	9,93	A
B	9,62	A
B	9,32	A
B	9,37	A
C	9,72	A
C	9,24	A
C	9,00	A
C	8,93	B

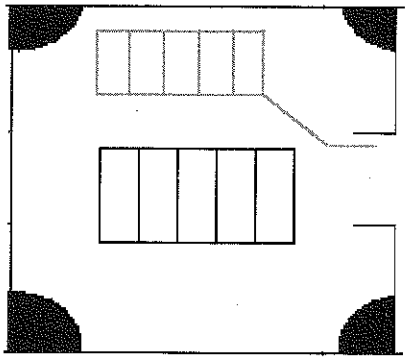
Appendix B





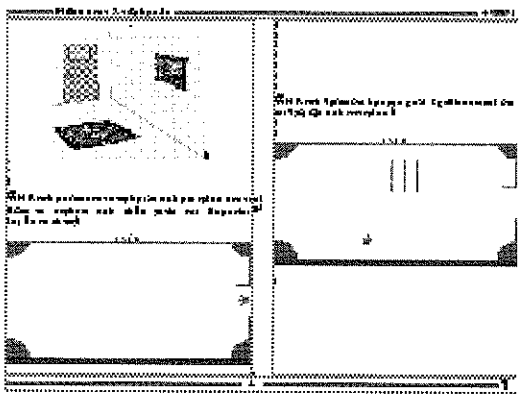




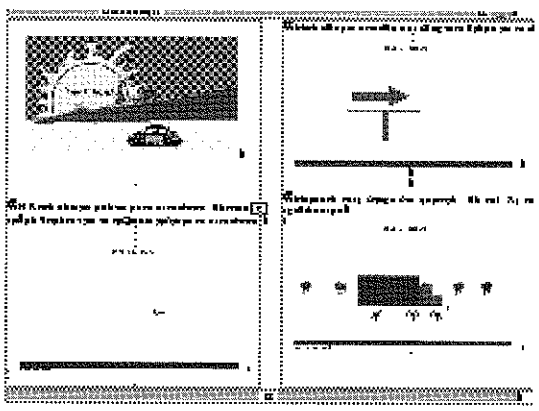


Appendix C

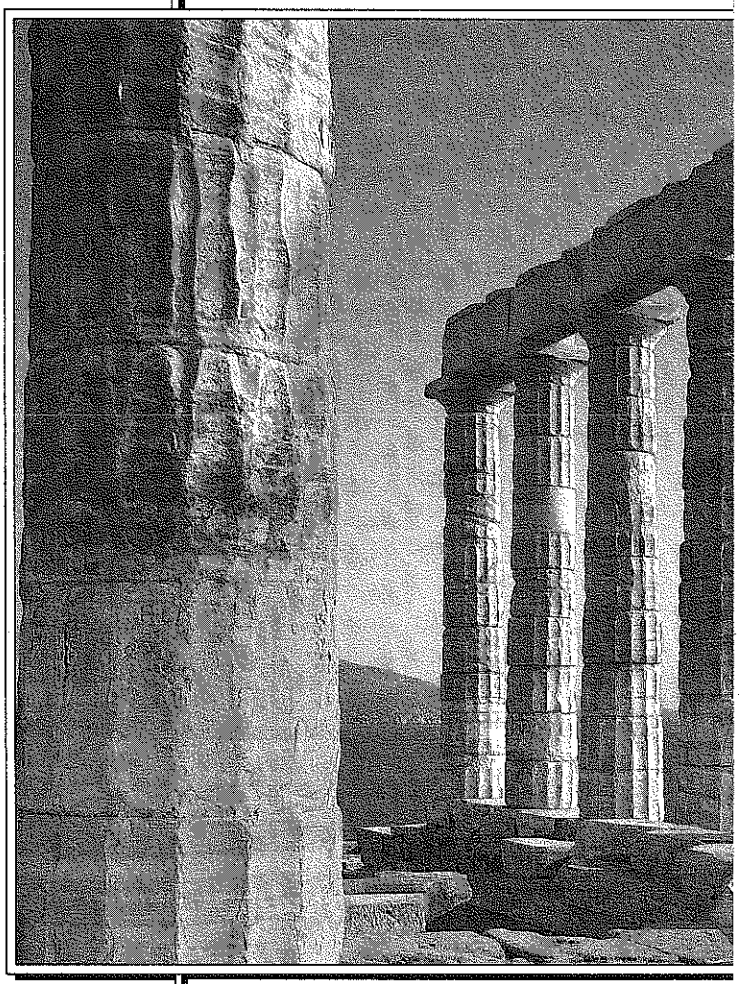
Work card 4



Work card 22



Plenary Addresses



Plenary Addresses

335 • *De Corte E.*
Toward Embedding Enriched Logo-Based Learning Environments in the School ...

351 • *Di Sessa A.*
Collaborating via Boxer

359 • *Harvey B.*
Symbolic Programming vs. Software Engineering - Fun vs. Professionalism - ...

367 • *Hoyles C.*
Exploiting Logo-like Environments for Learning Mathematics

377 • *Kontogiannopoulou G., Kynigos C.*
An Educational Perspective of the Socio-Cultural Pre-requisites ...

391 • *Makrakis V., Papavassiliou E.*
Learning Process in an Exploratory Data Handling Environment

397 • *Maritsas D.*
The Integration of the Use of Computer Technology in Greek General Education

405 • *Noss R.*
The Politics of Logo

Toward Embedding Enriched Logo-Based Learning Environments in the School Curriculum: Retrospect and Prospect

Erik De Corte

Center for Instructional Psychology and Technology (CIP&T)
University of Leuven, Belgium

Abstract: The major claim of this article is that to fully exploit Logo's potential for children's acquisition of knowledge and skills, it is necessary to embed the language in curriculum domains as a medium and a resource for learning and teaching. In addition, it is argued that Logo's utility and attractiveness for school education can be increased by enriching it with built-in instructional capability and supporting tools for learning and problem solving. This standpoint is elaborated starting from a brief retrospective analysis of a representative sample of studies on the cognitive effects of Logo experience on children. An explanation for the moderate positive results is presented by identifying instructional aspects of the learning environments designed in a number of successful investigations, that can account largely for those outcomes. Then, a project is described in which an attempt was made to enrich the Logo world by building in it a part of the necessary instructional support and tools. Finally, a prospect is outlined with respect to the integration of enriched Logo environments in the subject-matter fields of the school curriculum. As an illustration a recently initiated project aiming at the design of a Logo environment for learning geometry is briefly discussed.

Introduction

Although Logo has certainly become the number one programming language for education, there are some signs that in classroom practice the interest in and the use of Logo are waning (see e.g., Doyle, 1992; Kaput, 1992). This is in some sense surprising taking into account several interesting developments with respect to Logo such as LogoWriter, object-oriented Logo, control technology and Lego/Logo, on the one hand, and the accumulating body of research findings showing at least moderate positive effects of Logo programming on children's problem-solving and thinking skills, on the other. There are certainly a variety of reasons for this situation. For example, it has often been claimed that an advantage of Logo is that it has no floor and no ceiling. But it has recently rightly been argued by Picciotto (1993) that, while it is indeed rather easy to get started in Logo, serious problems raise along the road. Doing significant and interesting things with Logo in mathematics, for instance, soon requires a level of programming skill that is beyond children's, but often also teachers' mastery of the language. Without ignoring the importance of other reasons, this article claims that a major cause for the relative failure of Logo lies in the fact that until now we have not been sufficiently successful in integrating or embedding it within the curriculum of the school. Integrating does not mean here adding on Logo as one more subject to the already existing series of curriculum domains, but using it as a medium and a resource in the teaching of those domains. In addition, it is argued that Logo's potential and attractiveness for educational practice could be improved if the language is part of a software environment that has built-in supporting tools and instructional capability, and that is intelligible and easy to handle by teachers.

This stance will be elaborated in the following sections. The first section summarizes in a rather schematic way the research on the impact of Logo experience on children's thinking, and shows that there is at present fairly substantial support for the so-called "cognitive effects hypothesis", according to which learning computer programming in general, and learning Logo in particular is conducive for the development of important cognitive and metacognitive skills. The second section tries to identify characteristics of the

Table 1. Schematic overview of some major aspects of the Logo studies reported in the volume edited by Mayer (1988)

Study	Grade(s)	Conditions	Duration of instruction	Cognitive skills	Results
Carver (1988)	3-6	- Logo pr. solv. : graph. + lists - Logo pr. solv. : lists + graph.	50 hrs	Debugging	+
Clements & Merriman (1988) (see also Clements, 1990)	3	- Logo pr. solv. + transfer instr. - Control	58 hrs	Deciding on nature of problem Choosing performance strategy- Selecting representation Monitoring	+/- + +
Lehrer et al. (1988)	3	- Logo pr. solv. - Logo geometry - Control	58 hrs	Problem solving (Tower of Hanoi) Planning Metacognition	- + -
Littlefield et al. (1988)	5	- Logo pr. solv. - Logo pr. solv. + transfer instr. - Control	25 hrs	Following directions Identification of errors Planning ahead	+ + +

Table 2. Schematic overview of 18 studies on the effects of Logo published in the Journal of Educational Computing Research, 1985 (Vol. 1) - 1992 (Vol. 8)

Study	Age/Grade of subjects	Skills/concepts	Results
Watson, Lange, & Brinkley (1992)	4-5 years	learn and transfer of spatial problem-solving skills	+
Miller & Emihovich (1986)	5 years	comprehension monitoring abilities	+
Howell, Scott, & Diamond (1987)	5-7 years		-
Clements (1987)	1st grade	metacognitive skills	+
Lehrer & Randle (1987)	1st grade	metacognitive skills, esp. monitoring	+
Campbell, Fein, & Schwartz (1991)	1st grade	development of a measure scheme to estimate distance	+
Au & Leung (1991)	8-12 years	general problem-solving skills such as planning, analysis, monitoring, evaluation	+
Burns & Hagerman (1989)	3th grade	incremental conception of intelligence (mastery-orientation)	+
Gallini (1987)	4th grade	formulating directions	+
Swan (1991)	4th-6th grade	development and transfer of four problem-solving strategies: subgoal formation, forward chaining, systematic trial & error, analogy strategies	+
Catchart (1990)	5th grade	divergent thinking	+
Dalton & Goodrum (1991)	5th-6th grade	problem-solving skills (solving word problems)	+
Turner & Land (1988)	5th-8th grade (middle school)		-
Thompson & Chen Wang (1988)	6th grade	plotting points in a cartesian coordinate system & transfer to real life situations (e.g., map reading)	+
Ortiz & MacGregor (1991)	6th grade	long-term retention of understanding of variable	+
Many, Lockard, Abrams, & Friker (1988)	7th-8th grade	reasoning and inquiry skills in language	+ (o)
Kurland, Pea, Clement, & Mawby (1986)	10-12th grade		-
Grandgenett & Thompson (1991)	college freshm college juniors	ananalogical reasoning	+ -

learning environments designed in those investigations that can account for the positive results obtained. A feature of some of those studies is that an attempt was made to enrich the original Logo software by incorporating part of the instructional support into the programming environment. The third section describes a project in which we have developed in a more systematic way a new Logo system aiming at fostering the acquisition of programming ability and general cognitive skills, that involves such built-in supporting tools and coaching. Finally, the fourth section makes a plea for a more systematic embedding of Logo in the school curriculum in the sense indicated above. As an illustration an ongoing American project is briefly introduced, in which an enriched Logo environment for elementary geometry is designed. Remarkably, this Geo-Logo project is in some respect a hybrid combination of Logo programming with dynamic capabilities of already existing conjecturing software tools for geometry.

Logo fosters the acquisition and transfer of cognitive skills

In his well-known book *Mindstorms. Children, computers, and powerful ideas* Papert (1980) advocated Logo learning in children on the basis of two major arguments representing two variants of the so-called "cognitive effects hypothesis". First, Logo programming experience is conducive to the acquisition of general thinking and problem-solving skills, that can be transferred to other content domains. Second, Logo provides an ideal environment for learning basic mathematical concepts such as angle, polygons, variable, recursion, etc. Moreover, Papert argued that Logo programming skills can be acquired according to a self-discovery approach, analogous to the spontaneous and natural way in which the child learns to speak. In this respect, Papert's standpoint approaches the radical version of the constructivist view of learning (Cobb, in press).

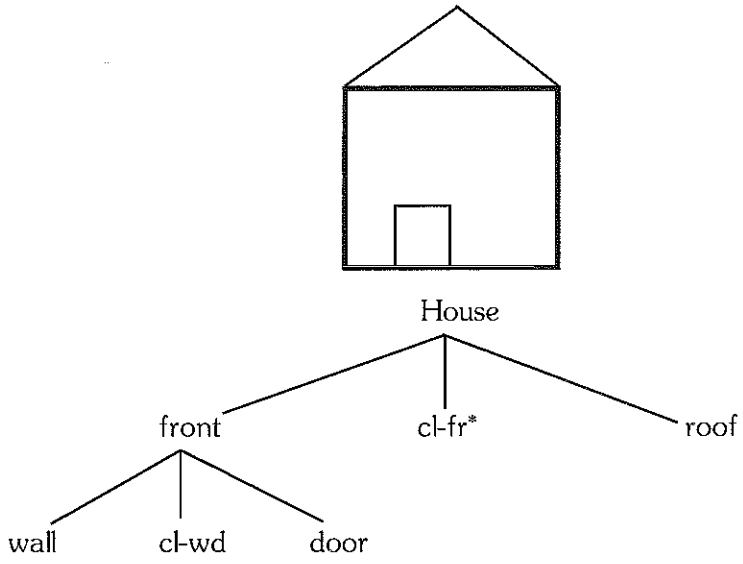
This ideas are at the origin of a substantial number of studies aiming at testing the cognitive effects hypothesis. A series of those investigations, carried out in the first part of the past decade according to Papert's self-discovery pedagogy, did not report any supporting evidence for the cognitive effects hypothesis (for a review of a representative sample of studies, see De Corte & Verschaffel, 1986, 1989). Most illustrative of this early research is the work of the Bank Street College of Education in New York (see Pea & Sheingold, 1987). Notwithstanding the lack of positive results, the cognitive effects hypothesis was not rejected, but instead the discovery approach was questioned. Consequently, the view that the acquisition of the programming skills would somehow "happen" to the children due to the unique characteristics of the Logo language (Leron, 1985), was largely abandonned, and a broad consensus emerged that Logo environments - while still stimulating students' own construction of their knowledge and skills - should at the same time provide systematic guidance and mediation to support their acquisition of problem-solving skills in programming, and possibly their transfer to other domains and situations (see e.g., Noss & Hoyles, 1992).

Taking the latter view as a starting point, more powerful Logo learning environments have been designed in subsequent investigations. Interestingly, many of these studies reported at least moderate positive results with respect to mastery of a variety of concepts and thinking skills within the Logo environment, but also with regard to (near) transfer of cognitive and metacognitive skills to new content domains. Space restrictions do not allow a detailed discussion of this research. Therefore, Tables 1 and 2 below only give a selective and schematic overview of a representative set of the available studies based on two sources, namely a volume edited by Mayer (1988) on *Teaching and learning computer programming. Multiple research perspectives*, in which half of the chapters reporting empirical work relate to Logo (see Table 1), and the *Journal of Educational Computing Research*, vol. 1 (1985) till vol. 8 (1992) in which 18 relevant studies were identified (see Table 2). Although the information in both tables is rather rough and sketchy, it nevertheless clearly illustrates the overall positive trend in the findings of the investigations considered. (For a more detailed review of the work reported in the Mayer-volume see De Corte, Verschaffel, & Schrooten, 1990; De Corte, Verschaffel, & Schrooten, 1992b.)

In our own research we have also obtained positive results in a project in which we developed, implemented and evaluated in sixth-grade classes a Logo teaching-learning environment, focussing on four thinking skills that are considered to be sensitive to programming experience and instrumental in acquiring competence in

programming : two metacognitive skills (planning and debugging), and two heuristics (problem decomposition and construction of an external problem presentation). The crucial part of the environment was the teaching of a Logo programming strategy, consisting of two main phases: a planning phase, carried out independently from the computer, and an integrated coding-and-testing phase at the machine (for a more detailed discussion see De Corte, Verschaffel, & Schrooten, 1992a).

In the planning phase a tree-like diagram (see Figure 1) is constructed in which the complex drawing is subdivided in building blocks that are easy to program (e.g., rectangles and triangles). In this phase planning as a metacognitive activity takes place, and elicits also the application of the two heuristics, namely problem decomposition and construction of an external representation.



(cl-fr means: connecting link between front and roof)

Figure 1. Tree-like diagraam for the drawing of a house

Once the planning is completed, the integrated coding-and-testing phase at the machine can begin. This activity, which involves debugging, is guided by two principles, namely top-down programming and immediate testing and debugging. Top-down programming means that the children are taught to start with the most global procedure, called the "mother procedure" consisting of the names of all the subsequent parts from the second level of the tree-like diagram. Subsequently, each component of this procedure is specified until the lowest level of the tree-like diagram is reached. The second principle consists of the immediate testing and debugging of each newly defined procedure after defining it. By calling the "mother procedure" the result appears on the screen and can instantly be evaluated; furthermore the error message ('There is no procedure named...') indicates which procedure has to be written next.

Instruction of this programming strategy was characterized by a good balance of exploratory learning activities, on the one hand, and systematic teaching aiming at mastery of the domain-specific Logo concepts and programming skills, on the other. The programming strategy was first demonstrated as a whole by a member of the research team, and the different steps were explained with a view to helping students to build a conceptual model of the processes required to carry out the task. Then, each component of the strategy was treated and practised separately; the teacher provided direct support whenever necessary. Finally, the children were given ample opportunity to practise the whole strategy in small groups of three pupils working on progressively more difficult problems. In the beginning the groups were guided intensively using hints, feedback and support, but the interventions were removed gradually as children's mastery of the skills increased.

The results of this study showed that it is possible to teach twelve-year-olds successfully a metacognitive strategy for programming, involving the systematic application of heuristic methods such as decomposing a problem into subgoals. Moreover, we found that mastery of those cognitive skills within the domain of Logo programming is sufficient to achieve near transfer (De Corte et al., 1992a).

What makes powerful learning environments powerful?

The positive results with respect to the acquisition and transfer of cognitive skills obtained in the Logo studies mentioned above, raise the question as to which characteristics of the learning environments designed in those investigations makes them powerful, i.e. can account for the fact that these environments elicit and support in students the appropriate learning activities and processes for achieving the intended outcomes ?

Without trying to be exhaustive, this section identifies a few such key characteristics based mainly on an analysis of the learning environments created in a sample of the studies referred to above, namely those listed in Table 1 (Carver 1988; Clements & Merriman, 1988; Lehrer, Guckenberg, & Sancilio, 1988; Littlefield, Delclos, Lever, Clayton, Bransford, & Frank, 1988), the one by Swan (1991) listed in Table 2, and our own teaching experiment summarized in the final part of the previous section (De Corte et al., 1992a).

Guided constructive learning in an apprenticeship-like environment

In a review of Harel and Papert's (1991) book *Constructionism*, Littlefield (1992) argues against the assumption that there are educational "magic potions", and claims that effective learning still requires good mediation and teaching. The latter point of view underlies the six successful studies mentioned above. They did not consider Logo as a magic potion as was largely the case in the earlier studies, but designed environments that were characterized by a good balance between discovery learning and personal exploration by the students, on the one hand, and systematic guidance, mediation, and instruction, on the other. A crucial common component in those teaching-learning environments was the direct instruction of problem-solving skills within the Logo context. A first step in designing such an instruction taken by all the investigators, is the identification of a particular set of skills to be trained. Some authors (e.g., Carver 1988 and Swan, 1991) went even one step further; they not only identified the skills to be trained, but developed also detailed performance models of those skills using concepts and techniques of cognitive psychological research.

The teaching strategies used for systematic instruction of the thinking skills differed in each study. However, at a more general level, there are a lot of communalities; most approaches can be described in terms of a combination of the strategies for problem-solving instruction identified by Collins, Brown and Newman (1989) in their apprentice-ship-like appraoch to teaching:

- modeling involves the observation by the student of an expert who is performing a certain task; this allows the student to construct an appropriate mental model of the activities that are required for skilled performance;
- coaching refers to the observation of the student by the teacher during task execution as a basis for giving hints and feedback with a view to improving performance;
- scaffolding consists of providing direct support to the student while (s)he is carrying out the task; this method derives from the Vygotskyan concept of the zone of proximal development;
- articulation refers to any technique that helps students to spell out and make explicit their knowledge and problem-solving procedures;
- reflection leads students to compare their own cognitive strategies and solution processes with those of experts, of other learners, and ultimatley, with a model of expert performance;
- exploration intends to increase the learner's autonomy in skilled problem solving as well as in discovering, identifying, and defining new problems. (For a more detailed discussion of the application of these teaching strategies in our teaching experiment see De Corte, Verschaffel, Schrooten, Olivié, & Vansina, 1991.)

An interesting characteristic of some studes is that efforts were made to enrich the original Logo-software, so that a part of the problem-solving instruction is provided by the computer. As an example, we refer to Clements (1990), who provided a powerful scaffold for debugging by allowing the child to define a procedure, and simultaneously watch it being executed, and to our own study, in which children were offered a Logo micro-environment that guided the process of decomposition through well-chosen questions (De Corte et al., 1992a). This raises the interesting question, whether and to what extent, the necessary problem-solving instruction can be built in the programming environment, and which part of it can better remain in the hands of the teacher.

Another feature of most of these six successful studies, but also of many of the investigations listed in Table 2, is that they make ample use of collaborative learning. In fact, several teaching strategies of the apprenticeship approach discussed above such as articulation and reflection, induce interaction and collaboration. Indeed, group work is a natural context for the explicitation of and reflection on problem-solving strategies. Nevertheless, in our own work we have observed that initially children experience great difficulty in talking about their problem-solving activities. However, we also found that this can be easily facilitated by giving special tasks, such as having students take alternatively the role of executor and of controller when they work in dyads. Another technique that we, but also Clements and Merriman (1988), applied is the use of cartoon-like figures (or homunculi) that represent the different thinking skills (see Figure 2). To stimulate communication about the skills, the teacher refers to the corresponding figure; for instance: "Here is an error. Let your garage keeper (representing the debugger) work on it and tell us what he is finding and doing".

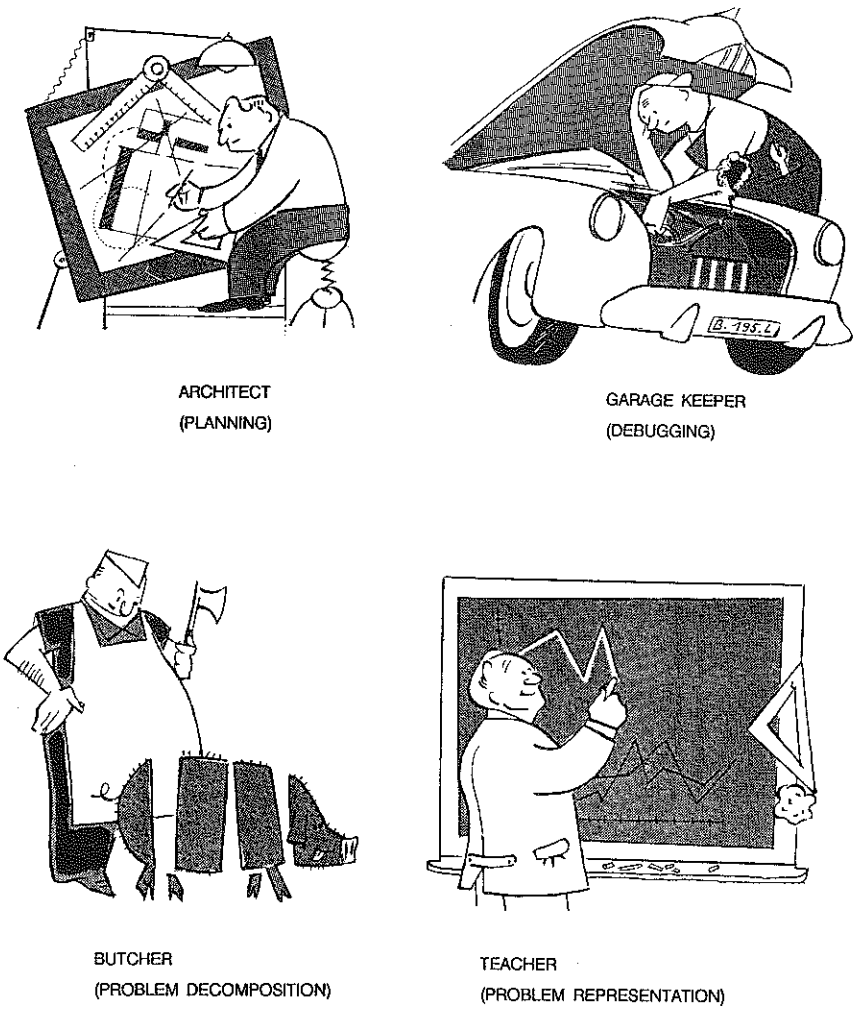


Figure 2: Cartoon-like figures representing four thinking skills

Explicit teaching for transfer of cognitive skills

In each of the six successful studies transfer was pursued intentionally and explicitly. Using Salomon and Perkins' (1987) terminology, two aspects can be distinguished in this efforts to teach for transfer : explicit, *mindful abstraction* of the concepts and skills one wants to transfer, and *decontextualisation* of these skills by demonstrating their usefulness in other situations and domains. This distinction is echoed in the two instructional components of Littlefield et al. (1988; see also Littlefield, Delclos, Bransford, Clayton, & Franks, 1989), namely, *framing*, involving "the act of relating a specific set of behaviors to a broader framework of problem solving", and *bridging*, referring to "the act of relating processes that occur within one context to similar processes occurring elsewhere" (Littlefield et al., 1989, p. 356).

The studies differed with respect to their focus on one or both of those components of transfer instruction. In most investigations only the first aspect was pursued; the skills learned in the Logo context were spelled out at a more general, abstract level. Three studies tried to realize both abstraction and decontextualisation, namely Clements and Merriman (1988), De Corte et al. (1990), and Littlefield et al. (1988). In the study of Littlefield et al. (1988) the first instructional strategy was, for instance, realized in relating the notion of using subprocedures in Logo programs to the general context of analyzing problems and breaking them down into manageable components. The second technique was applied in linking problem-solving strategies such as problem decomposition in Logo with familiar actions of Indiana Jones searching for a golden idol in the jungle in the movie "Raiders of the lost ark". In our own research (De Corte et al., 1990), mindful abstraction (or framing) was realized by explicitly labeling the thinking skills that were embedded in the trained programming strategy. Decontextualisation (or bridging) was pursued in teaching the children how to apply the learned skills while solving complex word problems. Clements and Merriman (1988; see also Clements, 1990) used the homunculi device to abstract the learned skills; he then offered problems in a diversity of contexts to bring about decontextualisation. Taking into account the available research, it seems reasonable to expect most perspective from studies in which both elements are pursued in teaching for transfer.

Enriching the Logo environment with supporting tools and instructional capability

The research reviewed above clearly shows that acquiring programming ability and more general cognitive skills in a Logo environment requires instruction and mediation. So far such help to children learning Logo was in most studies provided by a teacher and by written material. An additional outcome that derives from the available literature is that this teaching of programming and thinking skills is very time-consuming and effort-intensive. Therefore, an important question already raised in the previous section is, whether it is possible and desirable to incorporate a part of the necessary instructional support in the Logo software itself. Taking into account the advances in informatics there is no doubt that this is technically feasible. From an educational point of view it is plausible to anticipate that a form of "shared instruction" between teacher and computer will be beneficial for the teaching-learning process, assuming, however, that the computer support is well understood and easy to handle by the teachers.

Starting from this ideas we have worked over the past years on the development and implementation of an enriched Logo environment for pupils at the end of the primary school (12 year-olds). The enrichment means that the Logo-based system has built-in instructional capability in the form of scaffolding tools and coaching facilities to guide and support children's learning of planning, programming and debugging skills. More specifically, this instructional assistance focusses on facilitating children's acquisition of the Logo programming strategy discussed before, which consists of two main phases, namely a planning phase and an integrated coding-and-testing phase. In fact, the basic concept of the system was already presented at the Eurologo 91 Conference (De Corte et al., 1991). Meanwhile the environment has been implemented, and is now ready for try-out and evaluation in the classroom with pupils from the sixth grade on. The available space allows only a brief description of the components of this new Logo environment (see De Corte, in press and De Corte et al., 1991 for a more detailed overview of the system).

The Logo-based computer system consists of three levels. The first level corresponds with the traditional Logo turtle graphics and has two components, namely an execution part and an editor; the pupil can write

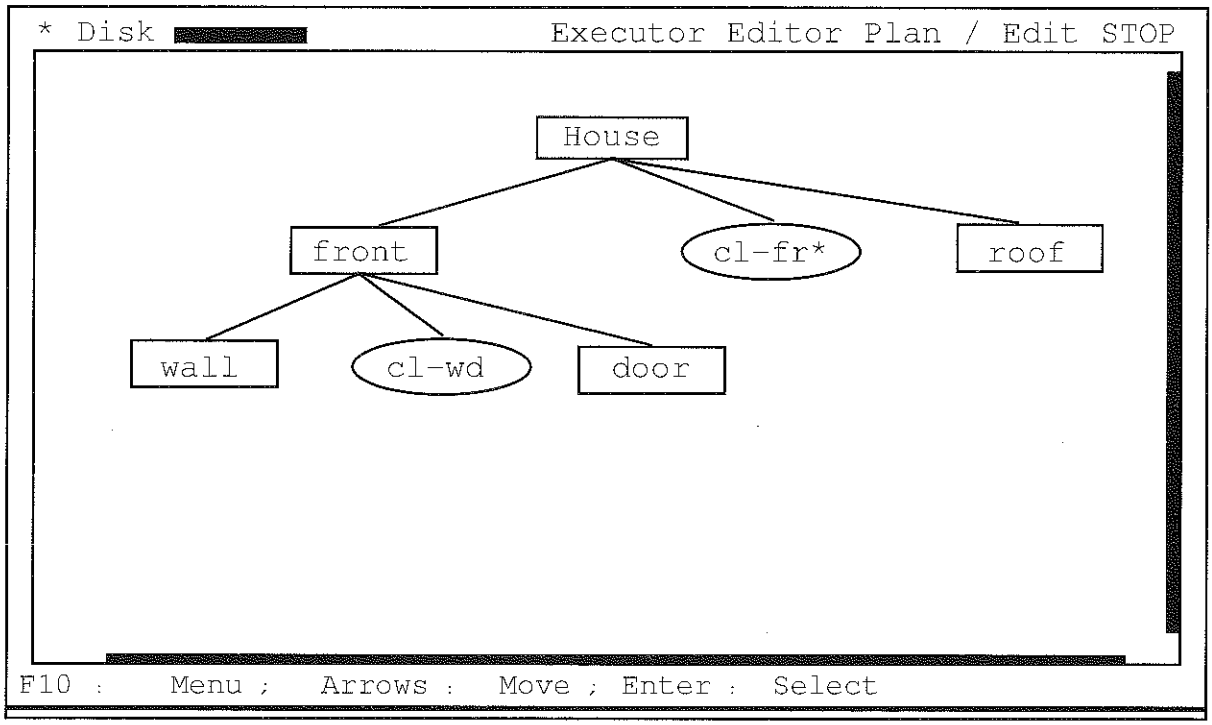


Figure 3. Planning tool: Tree-like diagram for writing a program that draws a house

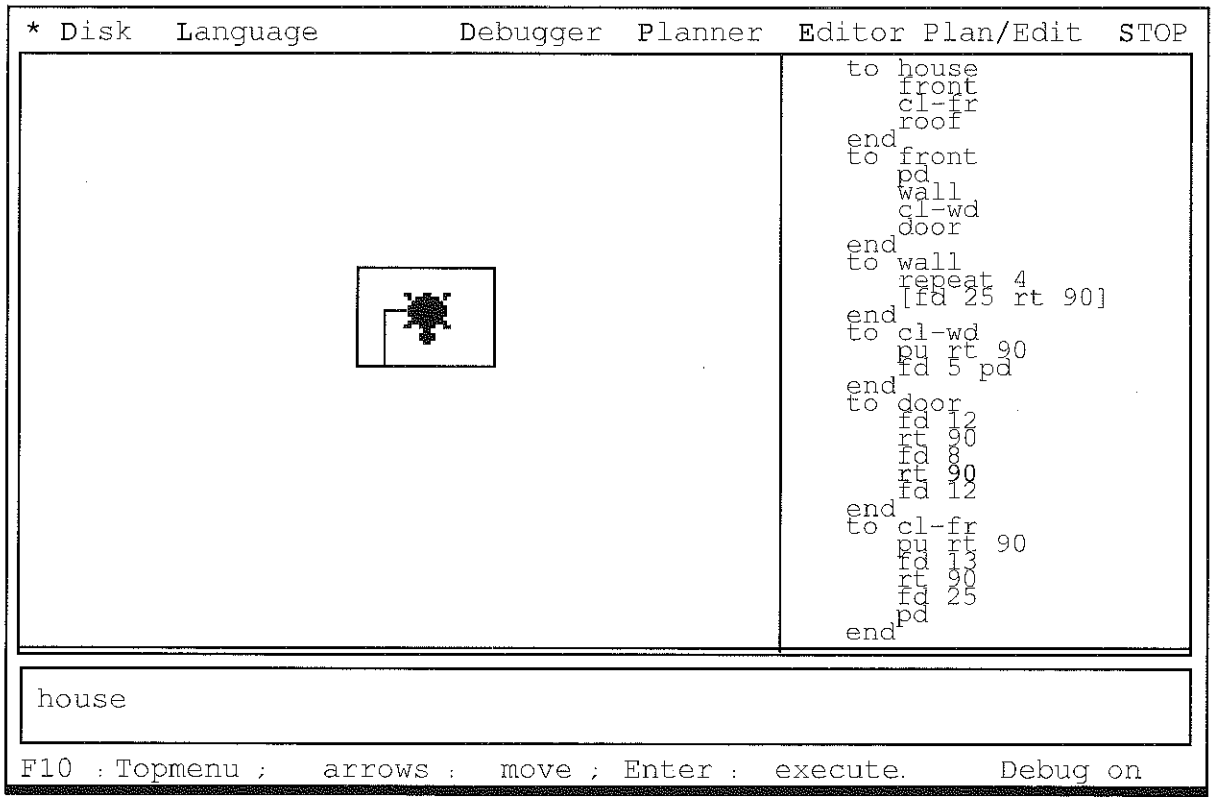


Figure 4. Debugging tool showing the program code and the graphical output
(The program line "rt-90 " in the procedure "to door" is highlighted on the screen)

Logo commands and programs for making drawings. At the second level two tools are available as scaffolds that stimulate and facilitate planning and debugging actitvities. At the third level the pupil's activities are analyzed by a computer coach; if necessary the coach gives comments and orienting help. Because the first level is similar to the usual Logo software, only the second and third levels will be somewhat explained.

Planning and debugging tools

At the second level Logo is supplemented with two tools that support planning and debugging activities. However, use of this scaffolding tools is not compulsory and should mainly depend on the level of competency already acquired by the pupil.

The *planning tool* is a graphical editor that allows the construction and manipulation of a graphic representation of the structure of a program in the form of a tree-like diagram. This graphical editor has special boxes for the following Logo constructs: procedures, procedures with parameters, iteration, recursion, and conditional statements (see Figure 3).

With the help of this tool the pupil can add, delete, replace, and rename parts of the tree-like diagram. Doing this at the computer has major advantages over paperwork. For instance, during the construction process at the machine the arrangement of the tree-like diagram as a whole stays clear and tidy, and at the third level the computer coach can analyze and possibly criticize the construction of that diagram, as well as the process of transposing the tree-like diagram in Logo code.

The *debugging tool* enables pupils to view the Logo-code and the corresponding graphical output together on the screen; each command is highlighted as it is being executed (see Figure 4). Moreover, the execution speed can be slowed down, and procedures can be carried out step-by-step.

Computer coach

The main difference with the previous level is that a coach checks the pupil's activities both in the planning and in the coding-and-testing phases. In both phases this coach, which is much less sophisticated than in traditional intelligent tutoring systems, locates syntactic and semantic errors as well as errors with respect to the programming strategy.

In the *planning phase* the coach checks syntactic and semantic aspects of the pupil's input during the construction of a tree-like diagram. The coach also reacts when certain criteria for an expert tree-diagram are not fulfilled (e.g., a procedure has too many subprocedures at the same level; two identical procedures are defined twice).

In the *editor* the coach reacts when the syntax of the programming code is not correct, and when the semantics of the code are not in accordance with the plan. Furthermore, the coach examines if the pupil takes into account the two principles underlying the coding-and-testing phase, namely top-down programming and immediate testing and debugging.

Within this third level which is also not compulsary, two sublevels are distinguished. At the first or tolerant sublevel the coach only gives a warning when an error is made; the pupil can ignore this warning. At the second or coercive sublevel the pupil is not able to proceed if the error is not corrected, or if the prescribed step from the programming strategy is not followed.

It is obvious that this enriched Logo environment has substantial built-in support. In this respect, two remarks should be made. First of all, although the system involves all this help, the aim is not that it should function as a stand-alone environment; to the contrary, the underlying idea is that it will operate in a teacher-mediated, collaborative learning context. It is hoped that the added value through the enrichment will improve Logo's potential and attractiveness for teachers, and that, as already mentioned, children's Logo learning will benefit from the shared instruction between teacher and computer. For instance, individualized

and small-group activities guided by the computer at some stages of learning Logo should allow a more efficient and productive use of time as well as an overall time gain. Simultaneously the teacher would get more opportunity for mediating students' knowledge and skill construction using cognitive apprenticeship methods discussed above such as modeling, explaining, coaching and scaffolding, but also articulation by the learners of and reflection on their knowledge and cognitive skills as well as exploration of new problems, hypotheses, and solution strategies. Depending on the objectives aimed at, the present needs and difficulties of the learners, and the nature of the intervention, these methods can be applied either individually, in small groups, or for the whole class.

Second, it would be incorrect to derive from the preceding description of the system that it is intrinsically directive in nature and, as such, conflicts with the constructivist character of learning processes. On the contrary, this Logo environment is representative of a moderate or realistic conception of learning : it attempts - in line with the position put forward in an earlier section - to achieve a balance between learner-controlled activity and teacher- or computer-initiated guidance, always taking into account individual differences in abilities, needs, and motivation between the learners. This balance permeates the entire enriched Logo environment. It becomes especially obvious in the optional character of the levels 2 (tools) and 3 (coach) of the system, and in the alternation of activities in which the pupil is in control, and others that are guided by the teacher or the computer. For example, the strict guidance of the computer coach at the second or coercive sublevel should in our view be preceded by Logo lessons in which pupils discover and discuss the value and usefulness of efficient planning and debugging strategies. Moreover, lessons in which learners work at this sublevel should also be followed by sessions in which the pupils themselves can decide whether they still want to follow the suggestions of the coach (tolerant sublevel), or even whether they still would like to use the available tools at level 2.

It is obvious that the hypothesized benefits and strengths of this Logo environment need to be tested in the projected empirical studies.

Toward the embedding of Logo environments in curriculum domains

Although enriched Logo environments as the one described in the previous section can contribute to an increase of Logo's utility and attractiveness for educational practice, we still need to go one step beyond such enrichments in order to fully exploit Logo's potential for children's acquisition of knowledge and cognitive skills in classrooms. Indeed, it is claimed here that a conditio sine qua non to make Logo really successful for learning and instruction in our schools is its embedding and integration in the curriculum. As already stated in the beginning of this article, embedding and integration do not mean adding on Logo as another subject to the already existing list of curriculum domains, but using it as a medium and a resource for learning and teaching within those domains.

Attempts at applying Logo within subject-matter fields have, of course, already been undertaken in a number of studies. However, they are mostly restricted to mathematics, and within mathematics the focus is largely on geometry. As illustrated, for example, by the Proceedings of Eurologo 91 (Calabrese, 1991) and by the volume edited by Gurtner and Retschitzki (1991), applications in other domains are rare, one exception being work done by Kynigos and his colleagues (Kynigos, 1991; Kynigos et al, 1993) in primary schools in Greece.

An analysis of the research results with respect to Logo and geometry shows a remarkable parallelism with the findings relating to the cognitive effects hypothesis summarized in a previous section. In the initial studies no positive impact of Logo on the acquisition of geometry concepts was found, and also here this was largely attributed to the lack of appropriate instructional interventions aimed at linking explicitly children's Logo experiences to the intended mathematical notions (for a review see Verschaffel, De Corte, & Schrooten, 1992). More recent investigations that took into account this weakness of the early work reported better, albeit still somewhat mixed results. Recently Clements and Battista (1992; see also Hoyles & Noss, 1992) summarized the state-of-the-art as follows :
"In sum, studies that have shown the most positive effects involve carefully planned sequences of Logo activities and teacher mediation of students' work with those activities. It would appear that Logo's potential to develop geometric ideas will be fulfilled to the extent that teachers and instructional materials

properly guide students' Logo experiences. This should include encouraging students to reflect on and forge links between Logo-based procedural knowledge and more traditional conceptual knowledge." (p. 452)

Notwithstanding the obvious attempts toward a real integration as described in this quotation by Clements and Battista, the present state-of-the-art is nevertheless that Logo has until now not had substantial impact on mathematics education in general, and geometry teaching in particular (Noss & Hoyles, 1992; Kaput, 1992). According to Noss & Hoyles (1992) this can at least partly be attributed to the discrepancy between doing Logo and doing mathematics. This discrepancy has indeed largely been overlooked in research and in educational practice, in the sense that it was mostly taken for granted that interacting with Logo objects is identical to doing mathematics, or will at least spontaneously lead to mathematical activity. However, it has now become clear that working with Logo does not necessarily induce in children such activity which involves abstraction, more specifically searching for mathematical structures and relations.

While the breakthrough of Logo in subject-matter domains in general, and in mathematics in particular has not occurred so far, there are signs that other domain-embedded software tools are rather succesfully introduced in a fairly substantial number of classrooms. For instance, with respect to geometry this is the case for a series of related software packages such as The Geometry Supposers (Schwartz & Yerushalmy, 1985), Cabri Géométrie (Baulac, Bellemain, & Laborde, 1988), and The Geometer's Sketchpad (Jackiw, 1991). These software programs help create in the classroom a powerful learning environment in which students become active learners of Euclidian geometry guided by their teacher. They were purposively designed to stimulate students at formulating and testing conjectures, which are considered to be the main activities of doing mathematics. These programs elicit and facilitate such activities by offering tools for constructing, manipulating, measuring, and exploring geometric shapes. An important aspect of the Supposers is the "repeat" option : the program captures the constructions carried out on a shape as procedures, that can be repeated afterwards on similar figures. This ability makes it possible to explore easily the effects of one's constructions across a set of equivalent figures, and, thereby, to test conjectures and hypotheses about geometric shapes on the basis of large amounts of visual information, and to look for invariants in geometric constructions. Cabri Géométrie and The Geometer's Sketchpad offer the same possibilities, although implemented in a different, even more sophisticated way : the learner can make continuous transformations of geometric figures simply by moving actions on certain parts of the on-screen shape. For example, the shape of a triangle can be changed by dragging a vertex, and during the transformation all the other parts (the other vertexes, the sides, the medians, ...) will move simultaneously and be in the correct position in the new triangle. There is evidence that, when used as intended, these computer construction tools create a powerful learning environment : traditional geometry lessons in which students absorb passively definitions, propositions, and theorems developed by others are transformed into teacher-mediated active and collaborative explorations of geometrical shapes resulting in stating, testing, and proving one's own conjectures. In addition, promising learning outcomes were already reported (for detailed reports about The Supposers see Schwartz, Yerushalmy, & Wilson, 1993; for a summary see Kaput, 1992).

The relative success of these interactive construction tools may be due to a series of features, such as their immediately perceived relevance for the teaching of geometry by a substantial number of teachers, and the fact that they elicit more easily than the available Logo software real mathematical activity in students. Starting from such presumptions, it looks an interesting perspective to explore the possibility of embedding Logo more thoroughly in the geometry curriculum by enriching the language with tools and resources, that allow to perform similar functions and activities as the domain-specific software programs mentioned above. A first attempt in that direction has recently been started in the so-called Geo-Logo project initiated by Clements and his colleagues (Clements & Meredith, 1993a, 1993b). To round off this article a short characterization of this project will be given.

The project aims at the design of a new Logo environment tailored for elementary school children's learning of geometry; it is part of a large-scale curriculum development project, funded by the National Science Foundation (NSF) in the U.S.A., and focussing on understanding and meaningful mathematical problem solving. Taking into account the mixed results of previous research, Clements and Meredith (1993a) have put forward five principles for the design of the Geo-Logo environment. It is anticipated that implemention of this principles in the new system will help to overcome a major negative finding reported in a number of

studies, namely that even after a fairly substantial amount of Logo experience children still continue to work mainly on a visually-based, non-analytic level. These five principles are (Clements & Meredith, 1993a, p. 6-7):

1. Encourage construction of the abstract from the visual.
2. Maintain close ties between representations - Logo code, the action of the turtle, and the resultant figure.
3. Facilitate examination and modification of code - ease of editing and repeating constructions and operations.
4. Encourage procedural thinking.
5. Provide freedom within constraints.

Interestingly, the design of Geo-Logo on the basis of these principles has resulted in a learning environment that constitutes a kind of hybrid combination of a traditional Logo world with the dynamic capabilities of the interactive construction tools mentioned above. This is achieved by modifying Logo's overall structure, and enriching it with a series of tools that allow a variety of activities such as measuring of angles and line segments, turning and moving the turtle with the mouse, changing the shape of a figure directly on the screen. The overall structure consists of a command center, a teach window, and a graphics window (see Figure 5).

Commands can be entered in the immediate mode in the command center, and as procedures in the teach window; the turtle draws in the graphic window and can be dragged to a new location. A major characteristic of the system is the dynamic link between the components, in the sense that changes in the commands, either in the command center or in the teach window, are immediately and automatically reflected in the drawing. But in addition, the linkage between symbolic code and visual output is reciprocal. Indeed, when the child uses the "Draw commands" tool to turn and move the turtle by dragging, or the "Change shape" tool to modify the shape of a figure directly on the screen, the code in the command center is immediately adapted to the new drawing.

This brief description is probably sufficient to show that Geo-Logo enriches Logo in a way that allows to embed the language more substantially in the geometry curriculum. At the same time it holds the promise of inducing more easily real mathematical activity. Indeed, conditional on the appropriate mediation by the teacher, Geo-Logo's flexible and dynamic overall structure can facilitate the linking of different mathematical representations, and, consequently, the transition from an intuitive and visual level of operating to a more analytic and abstract approach, resulting in the discovery of mathematical relations and geometrical structures. Further research to test the anticipated merits of Geo-Logo is certainly necessary, but initial empirical data (Clements & Meredith, 1993a) have already yielded promising results. More generally, Geo-Logo indicates avenues for exploring and exploiting more thoroughly the intrinsic educational potential of Logo for mathematics learning and teaching in particular, and for school education in general.

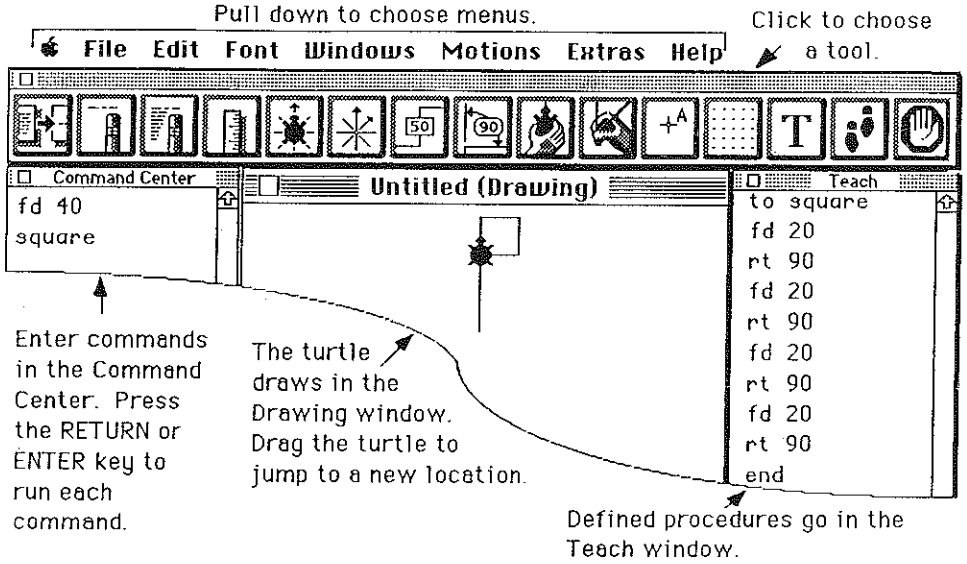


Figure 5: Example of the main Geo-Logo screen (reprinted with the permission of the authors)

REFERENCES

Au, W.K., & Leung, J.P. (1991). Problem solving, instructional methods and Logo programming. *Journal of Educational Computing Research, 7*, 455-467.

Baulac, Y., Bellemain, F, & Laborde, J.M. (1988). *Cabri Géométrie.* [Software.] Paris, France : Cedic-Nathan.

Burns, B., & Hagerman, A. (1989). Computer experience, self-concept and problem-solving : the effects of Logo on children's ideas of themselves as learners. *Journal of Educational Computing Research, 5*, 199-212.

Calabrese, E. (Ed.) (1991) *Proceedings. Third European Logo Conference. Parma, Italy, 27-30 August 1991.* Parma, Italy : ASI (Associazione Scuola e Informatica).

Campbell, P.F., Fein, G.G., & Schwartz, S.S. (1991). The effects of Logo experience on first-grade children's ability to estimate distance. *Journal of Educational Computing Research, 7*, 331-349.

Carver, S.M. (1988). Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 259-297). Hillsdale, NJ: Lawrence Erlbaum Associates.

Catchart, W.C. (1990). Effects of Logo instruction on cognitive style. *Journal of Educational Computing Research, 6*, 231-242.

Clements, D.H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research, 3*, 73-94.

Clements, D.H. (1990). Metacomponential development in a Logo programming environment. *Journal of Educational Psychology, 82*, 141-149.

Clements, D.H., & Battista, M.T. (1992). Geometry and spatial reasoning. In D.A. Grouws (Ed.), *Handbook of research on methematics teaching and learning* (pp. 420-464). New York, NY : Macmillan.

Clements, D.H., & Meredith, J.S. (1993a). *Design of a Logo environment for elementary geometry.* Buffalo, NY : State University of New York at Buffalo.

Clements, D.H., & Meredith, J.S. (1993b). *Geo-Logo documentation.* Buffalo, NY : State University of New York at Buffalo.

Clements, D.H., & Merriman, S. (1988). Componential developments in Logo programming environments. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 13-53). Hillsdale, NJ : Lawrence Erlbaum Associates.

Cobb, P. (in press). Constructivism. In T. Husen & T.N. Postlethwaite (Eds.), *International encyclopedia of education.* (Second edition) Oxford, UK : Pergamon Press.

Collins, A., Brown, J.S., & Newman, S.E. (1989). Cognitive apprenticeship : Teaching the craft of reading, writing and mathematics. In L.B. Resnick (Ed.), *Knowing, learning, and instruction. Essays in honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ : Lawrence Erlbaum Associates.

Dalton, D.W., & Goodrum, D.A. (1991). The effects of computer programming on problem-solving skills and attitudes. *Journal of Educational Computing Research, 7*, 483-506.

De Corte, E. (in press). Embedding computers in powerful learning environments. In S. Vosniadou, E. De Corte, H. Mandl, & R. Glaser (Eds.), *International perspectives on the psychological foundations of technology-based learning environments.* Hillsdale, NJ : Lawrence Erlbaum Associates.

De Corte, E., & Verschaffel, L. (1986). Effects of computer experience on children's thinking skills. *Journal of Structural Learning, 9*, 161-174.

De Corte, E., & Verschaffel, L. (1989). Logo, a vehicle for thinking. In G. Greer & G. Mulhern (Eds). *New directions in mathematics education* (pp. 63-81). London : Routledge.

De Corte, E., Verschaffel, L., & Schrooten, H. (1990, March). *Cognitive effects of computer-oriented learning.* Invited address presented at the Seventh International Conference on Technology and Education, Brussels, Belgium.

De Corte, E., Verschaffel, L., & Schrooten, H. (1992a). Cognitive effects of learning to program in Logo: A one-year study with sixth graders. In E. De Corte, M. Linn, H. Mandl, & L. Verschaffel (Eds.),

Computer-based learning environments and problem solving (pp. 207-228). (NATO ASI Series F : Computer and Systems Sciences, vol. 84). Berlin : Springer-Verlag.Kynigos C, Gyftodimos G. and Georgiadis P. (1993) Empowering a Society of Future Users of Infromation Technology: A Longitudinal Study of Application in Early Education, European Journal of Information Systems 2, (2), pp. 139-148.

De Corte, E., Verschaffel, L., & Schrooten, H. (1992b). Kognitive Effekte computergestützten Lernens : Zum Stand der Forschung. *Unterrichtswissenschaft. Zeitschrift für Lernforschung, 20*, 12-33.

De Corte, E., Verschaffel, L., Schrooten, H., Olivié, H., & Vansina, A. (1991). A Logo-based tool-kit and computer coach supporting the development of general thinking skills. In E. Calabrese (Ed.), *Proceedings. Third European Logo Conference. Parma, Italy, 27-30 August 1991* (pp. 433-450). Parma, Italy : A.S.I. (Associazione Scuola e Informatica).

Doyle, M.P. (1992). Logo : Little used in UK schools after ten years, why ? *Eurologos, 1*, 32-35.

Gallini, J.K. (1987). A comparison of the effects of Logo and a CAI learning environment on skills acquisition. *Journal of Educational Computing Research, 3*, 461-477.

Grandgenett, N., & Thompson, A. (1991). Effects of guided programming instruction on the transfer of analogical research. *Journal of Educational Computing Research, 7*, 293-308.

Gurtner, J.L., & Retschitzki, J. (Eds.). (1991). *Logo et apprentissages*. (Techniques et méthodes pédagogiques.) Neuchâtel, Suisse : Delachaux et Niestlé.

Harel, I., & Papert, S. (1991). *Constructionism*. Norwood, NJ : Ablex Publishing.

Howell, R.D., Scott, P.B., & Diamond, J. (1987). The effects of "instant" Logo computing language on the cognitive development of very young children. *Journal of Educational Computing Research, 3*, 249-260.

Hoyles, C., & Noss, R. (Eds.). (1992). *Learning Logo and mathematics*. Cambridge, MA : MIT Press.

Jackiw, N. (1991). *The Geometer's Sketchpad*. [Software.] Berkeley, CA: Key Curriculum Press.

Kaput, J. (1992). Technology and mathematics education. In D.A. Grouws (Ed.), *Handbook of research on mathematics teaching and learning* (pp. 515-556). New York, NY : Macmillan.

Kurland, D.M., Pea, R.D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research, 2*, 429-458.

Kynigos, C. (1991). Centration sur le processus avant le contenu : peut-on pratiquer Logo dans une école primaire en Grèce ? In J.L. Gurtner & J. Retschitzki (Eds.), *Logo et apprentissages* (pp. 93-104). (Techniques et méthodes pédagogiques.) Neuchâtel, Suisse : Delachaux et Niestlé.

Kynigos C, Gyftodimos G. and Georgiadis P. (1993). Empowering a Society of Future Users of Infromation Technology: A Longitudinal Study of Application in Early Education, *European Journal of Information Systems* 2, (2), pp. 139-148.

Lehrer, R., Guckenberg, T., & Sancilio, L. (1988). Influences of Logo on children's intellectual development. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 75-110). Hillsdale, NJ : Lawrence Erlbaum Associates.

Lehrer, R., & Randle, L. (1987). Problem solving, metacognition and composition : the effects of interactive software for first-grade children. *Journal of Educational Computing Research, 3*, 409-427.

Leron, U. (1985). Logo today : Vision and reality. *The Computer teacher*, 26-32.

Littlefield, J. (1992). Book review of *Constructionism*, by I. Harel and S. Papert, Norwood, NJ : Ablex Publishing, 1991. *Journal of Educational Computing Research, 8*, 395-399.

Littlefield, J., Delclos, V.R., Lever, S., Clayton, K.N., Bransford, J.D., & Franks, J.J. (1988). Learning Logo : Method of teaching, transfer of general skills, and attitudes toward school and computers. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 111-135). Hillsdale, NJ : Lawrence Erlbaum Associates.

Littlefield, J., Delclos, V.R., Bransford, J.D., Clayton, K.N., & Franks, J.J. (1989). Some prerequisites for teaching thinking : Methodological issues in the study of Logo programming. *Cognition and Instruction, 6*, 331-366.

Many, W.A., Lockard, J., Abrams, P.D., & Friker, W. (1988). The effect of learning to program in Logo on reasoning skills of junior high school students. *Journal of Educational Computing Research,*

4, 203-213.

Mayer, R.E. (1988). *Teaching and learning computer programming. Multiple research perspectives.* Hillsdale, NJ : Lawrence Erlbaum Associates.

Miller, G.E., & Emihovich, C. (1986). The effects of mediated programming instruction on preschool children's self-monitoring. *Journal of Educational Computing Research, 2*, 283-297.

Noss, R., & Hoyles, C. (1992). Looking back and looking forward. In C. Hoyles & R. Noss (Eds.), *Learning Logo and mathematics* (pp. 431-468). Cambridge, MA : MIT Press.

Ortiz, E., & MacGregor, S.K. (1991). Effects of Logo programming on understanding variables. *Journal of Educational Computing Research, 7*, 37-50.

Papert, S. (1980). *Mindstorms. Children, computers, and powerful ideas.* New York, NY : Basic Books.

Pea, R.D., & Sheingold, K. (1987). *Mirrors of minds : Patterns of experience in educational computing.* Norwood, NJ : Ablex Publishing.

Picciotto, H. (1993 March). *A tool-based approach to square roots : Geoboards, blocks, calculators.* Paper presented at the 71st Annual Meeting of the National Council of Teachers of Mathematics, Seattle, WA.

Salomon, G., & Perkins, D.N. (1987). Transfer of cognitive skills from programming : When and how ? *Journal of Educational Computing Research, 3*, 149-169.

Schwartz, J.L., & Yerushalmy, M. (1985). *The Geometric Supposers.* [Four software packages.] Pleasantville, NY : Sunburst Communications.

Schwartz, J.L., Yerushalmy, M., & Wilson, B. (1993). *The Geometric Supposer : What is it a case of?* Hillsdale, NJ : Lawrence Erlbaum Associates.

Swan, K. (1991). Programming objects to think with : Logo and the teaching and learning of problem solving. *Journal of Educational Computing Research, 7*, 89-112.

Thompson, A.D., & Chen Wang, H.M. (1988). Effects of a Logo microworld on student ability to transfer a concept. *Journal of Educational Computing Research, 4*, 335-347.

Turner, S.V., & Land, M.L. (1988). Cognitive effects of a Logo-enriched mathematics program for middle school students. *Journal of Educational Computing Research, 4*, 443-452.

Verschaffel, L., De Corte, E., & Schrooten, H. (1992). Transfer van Logo-kennis en programmeervaardigheden. (Transfer of Logo knowledge and programming skills). *Tijd-schrift voor Onderwijsresearch, 17*, 40-54.

Watson, J.A., Lange, G., & Brinkley, V.M. (1992). Logo mastery and spatial problem-solving by young children : effects of Logo language training, route-strategy training, and learning styles on immediate learning and transfer. *Journal of Educational Computing Research, 8*, 521-540.

Collaborating via Boxer

Andrea A. diSessa

School of Education
University of California, Berkeley

Abstract: In this paper we examine several case studies of the use of Boxer in collaborative modes. We identify the reasons Boxer seems especially good at supporting collaborative work.

Introduction

Boxer is a flexible, general-purpose computational system designed to serve educational purposes. Our commitment, like that of the Logo community, is toward giving individuals, especially students and teachers, the fullest use of computers. That means Boxer is a complete programming environment. Unlike the Logo community, we decided about 10 years ago to break from an incremental approach in order to redesign the computer medium we wanted to use from scratch. Now that Boxer is a reality and has been used in schools for several years, we have a good opportunity to compare different material means toward basically the same ends — Boxer compared to Logo and Logo derivatives like LogoWriter and Microworlds.

Rather than a side-by-side comparison, we will here simply explore some apparent successes of Boxer. We will try to trace their causes back to special characteristics of Boxer. Comparison with contemporary Logos and other languages will be left mostlyto others.

The particular dimension of Boxer we examine is its ability to support collaboration. Boxer was designed with an eye toward (1) learnability and comprehensibility and (2) general expressiveness and usefulness. In this light it may be surprising that some of our best successes seem to have come in fostering good collaboration. Indeed, we were not entirely prepared for what happened when we first used Boxer in substantial collaborative modes. This paper is our first chance to reflect systematically on collaborative successes. We will argue that the bases for these successes are the same as the bases for the improved learnability and expressiveness that we did expect.

In fairness, we did anticipate Boxer would operate in some important, fundamentally social modes. One of the best developed early images of Boxer use was of a community of tool builders and sharers (diSessa, 1990). There are important reflections of this idea in several examples, below. But in other ways, sharing and collaborating via Boxer proved a phenomenon to explain rather than an expectation validated. We proceed by a series of case studies.

First Course

The first organized Boxer course (a summer course for students age 13-16 on statistics) was, not unexpectedly, somewhat ill-prepared. The software implementation at the time was incomplete and buggy. Machines crashed frequently and their 16 MHz 68020 processors (close to that in current Macintosh Classics, half the power of low end LCIIIs) fairly crawled. The teacher we recruited was superb and experienced with Logo, but he had not taught a statistics course before with any technological help, and he had received almost no Boxer training. There was no Boxer documentation, and we had had almost no experience teaching Boxer to anyone. To make matters worse the course was only six weeks long (only half the time each day devoted to Boxer), so we had little time to recover from false steps; students in the course had an almost unmanageable range of expertise, from "hackers" to essentially computer-naive individuals.

Planning for the course initially was faulty as well. Buoyed by perhaps overoptimistic expectations with

little counteracting experience, the Boxer post-doc helping organize the course convinced the teacher to have the students implement every tool they used, from scratch.

Into the third week, the course bordered on collapse. Even the experienced programming students had failed to implement a workable version of the very first statistical tool assigned by the teacher. It was time to re-think.

Jumping ahead, by the end of the course each team (four total) of two or three students had produced a fine workable programming project that illustrated and elaborated statistical principles from the course. The "hackers," occupying 2 of the 4 project groups, produced fairly impressive products. For example, one group, produced a tool that generated "bar and whisker" charts showing confidence intervals on samples drawn from specified populations. The program did not rely on numerical charts or other "tricks," but used an internal simulation of sampling to generate reliable approximations to rather fancy statistical formulas.

Yet even the least sophisticated students also produced excellent working program-projects. One group of three, who, among them, had had only the barest prior experience programming Basic, developed an excellent pedagogical simulation. They developed a generalization of the standard science museum display of the normal curve emerging from balls dropped into a triangular array of pegs, collecting in bins at the bottom. Their simulation represented sequential samples of binary values (yes, no) from a selectable distribution (say, 55% of the population are "yes"). The simulation could be single stepped,allowing process explanations "in slow motion." It could also be run one batch (say, 16) samples at a time (say, producing one "ball" in the "7 yes" bin), or in multiple batches to show the growing histogram distributions. This program internally collected and sorted numerical data as well as showing it graphically.

Although a sample of four projects is scarcely definitive data, our prior experience with summer programs of this sort using Logo were rather different. Many groups failed to produce the quality of projects demonstrated by this class. Even worse, projects combining hackers with less computer sophisticated students were typically dominated by the hackers to the point that other students completely lost ownership of the computer aspects of theproject. Only hackers could run and explain their group's program.

Although we were not prepared with the kind of data collection one might have hoped for, we had both the expert observations of the teacher and unrehearsed, videotaped "exit interviews" of students concerning their projects. The teacher also believed these achievements were clearly beyond what he had experienced before. The video tapes, including one narrated by a near computer-naive student (beginning the course), showed excellent mastery of the project programs, including debugging-on-the-fly and perusal and explanation of the program's internal structure.

How had the course been turned around from near disaster to unusual success? In a word, code-sharing saved the day. After bailing out from "every programmer for him/herself" the teacher had begun bringing some central program-tools into the class to give to students. For example, he programmed a simple statistical calculator that had several essential statistical functions ready at hand. Graduate student observers dropped in other tools, e.g. a MEMBER? primitive (is X a MEMBER of set Y?) and a simple sorter of numerical data. Students too began contributing to this pool of tools that other students borrowed.

The transition to a tool-building and sharing culture was not as simple as project members and teacher priming the pump. Some students initially thought borrowing code was cheating. Others kept code they had received private, as if it were theirs to hoard, until this inconsistency was publicly pointed out. Still other students persisted in writing deliberately inscrutable code with great pride to the end of the course.

So, clearly, sharing was a cultural construction and achievement, at least in some measure. But was it completely cultural, or did Boxer have a role in fostering it? We believe Boxer did have an important role.

In the first instance, after four good years of further experience with Boxer, we have found code sharing consistently much more prominent than in any of our many years working with Logo. [N.B.: It is well worth noting that the structure and presentation of Logo programs, per se, have not changed essentially at all, even

while the programming environment has been enriched with text processing (LogoWriter), better file and other organization, and (especially with Microworlds) many more ready tools (paint tools, prefabricated buttons). It is the program structure and presentation, however, that we argue is the root of Boxer's collaborative spirit.] Continuing examples below help make this point. More profoundly, when we look to see why sharing in Boxer works well, it seems some of Boxer's innovations are at the root. Here we list four, which we elaborate and extend in other cases, below.

(1) *Visibility* — Boxer is designed to allow students to see as directly as possible all aspects of the system. A good example is variables, which are completely concrete boxes containing a value. When a variable changes, the result automatically appears on the screen. In contrast, no wide-spread language has a notation for the *fact* of a variable having a certain value. Instead, if variables are shown at all, they are shown in statements that command their existence (declarations) or change a value ("set" or "make"). Not only can you see more structure in Boxer (see also below), but you can dynamically see how programs change their environment. In terms of learnability, the idea of variable ceases being a barrier. In terms of sharing, seeing what another's code does and how it does it is much easier.

(2) *Pokability* — Variables can be set by hand simply by editing their contents, as well as by program control. More importantly, programs themselves can be poked in bits and pieces to see what they do. Concrete stepping is trivial and frequently used in Boxer to understand a piece of code. All a user has to do is point to each line in sequence and execute it.

(3) *Structure* — Boxer provides an extremely natural and powerful visual structure to organize code. One uses boxes inside boxes (inside boxes) to organize hierarchical code. Subprocedures may appear directly in place, or they may be defined locally in any box, just where they belong, at any level of the hierarchy. This provides a much more expressive presentation for program structure than the typical list of procedures. Programmers don't *have* to program so as to communicate with better organized structure, but Boxer provides resources that allow this. We have some reasonably well-studied cases of studentprogrammers achieving breakthroughs in programming complexity by mastering the expressive possibilities inherent in Boxer organizational resources (Ploger and Lay, 1992).

Boxer also provides a natural and intuitive way of *perusing* complex code — shifting focus, suppressing or displaying detail as needed. Boxes can be expanded or shrunk, or they can be "entered" by expanding to full screen. This makes for an easy inspection of program structure, allowing students to keep the screen as simple as possible, but exposing as much context as desired (within limits).

(4) *Visible, manipulable units* — Because of Boxer's structuring, it is fairly easy to arrange meaningful units that appear as complete, manipulable units on the screen. So a complex program, complete with many subprocedures, may appear simply as a small black box that can be cut out as a unit, transported and pasted in an appropriate place in another program. Well-designed units "travel" more easily in Boxer than in less concretely organized kindred languages.

We were not in a position to document in detail how sharing worked in this class. But the class's success first alerted us to the power of sharing in Boxer and began the analysis that pinpointed areas such as those listed directly above. We could see more directly how things worked in examples below.

Steve's Graphing Adventure Game

Three years ago we gave a full year-long class on physics for sixth graders, age 11 and 12, based on Boxer. Two of the students, Steve and Bill (pseudonyms) produced an impressive program as a final project. It was an elaborate graphing adventure game in which players were invited along on a fantasy journey in outer space. The hero of the fantasy was confronted by a series of challenges involving motion, and the player had to select from among a set of 5 position, velocity and acceleration graphs the one that indicated a motion that would extricate the hero from his current predicament. If you selected the correct graph, you advanced to the next level of the adventure. If not, you would be scolded for selecting the wrong graph (with an insulting but enlightening description of the motion represented by your selection), and you would be sent back to level

one to start again. Level five of the challenges was, itself, an independent game of "space invaders," shooting invading aliens with three levels of difficulty.

The program itself was stunning in size and complexity for sixth grade students. It consisted of approximately 500 boxes, with file size over 100 kilobytes. The program stored and displayed dozens of graphs, contained a textual introduction and help with reading graphs, produced interactive text and dynamically changing menus according to the current game context, and it had a cumulative scoring subsystem. The story-line and interactive text was marvelously inventive, filled with typical sixth grade humor.

How did Steve and Bill manage such an elaborate production? First, Steve was an intent and good "hacker." He was the most dedicated programmer in the physics course, but not by a very wide margin. (In the first half of the course, several other students, including one girl seemed as competent and interested.) He was certainly not the best Boxer programmer in the school. Steve did, in fact, do almost all the programming for the project, but it seems clear various modes of collaboration and code sharing helped tremendously.

To begin, Bill produced essentially all the interactive text for the game and, we believe, provided the basic form as well. Excellent synergistic collaborations between students with complimentary talents cannot be claimed as Boxer successes, except to this extent: Boxer's visibility and inspectability seem to help ordinarily less involved students keep a general understanding of, and feelings of ownership toward, Boxer programs. Indeed, there are many aspects of producing a typical Boxer program, including box layout (user interface design) and hypertext editing, that involve little more than the skills one learns in the first few days of Boxer experience. These can be manifestly important contributions while still not requiring exotic programming skill. We mentioned the surprisingly good collaboration between expert and less expert programmers in the above example, and we add others below.

There were other collaborative factors. On inspecting the game's code, I came across some telltale signs. I found, for example, some documentation and instructions for a graphing tool that was obviously unfinished (it contained notes to the author about features to be added and other unfinished and undesigned aspects of the tool). Evidently this had been brought into the classroom by a graduate student or teacher, and Steve saw its value and "borrowed" it.

What Steve took from the tool was quite interesting. First, he reused the basic spatial organization of having a separately named graphing "turtle" in each graph box so as easily to manage multiple graphs. Steve obviously added new graphs using the sameparadigm as the tool he borrowed. Second, Steve used the axis drawing part of the tool. Third, he took the idea of storing graphical data as lists of numbers that was the basis of the graphing tool. On the other hand, Steve left a lot of the tool behind. Much of the working structure was simply deleted from his world, including the basic routine to plot points. Instead, he wrote his own (rather inelegant) graphing routine. Either he felt he did not need to borrow that, or he felt he needed something slightly different, or both.

All in all, Steve's borrowed and new code and their articulation show that he rummaged selectively for ideas as well as code components and had little difficulty joining his own ideas with those of others. Visibility, pokability, structuring resources andclear unit boundaries, we believe, played essential roles in allowing the construction of this complex program.

There is one additional collaborative aspect of this construction that is easy to overlook. Almost all the features of Boxer that allow excellent collaboration and code sharing in general apply to individuals, as well. Good modularity and structure allow easy combining of past work with present. All the inspectability of Boxer helps programmers regenerate and extend an understanding of their "old" code. Thus, many of the good collaborative properties should show up in individual programmers' combining and extending their own work into very complex creations.

Steve and Bill's graphing adventure game had an excellent example of this. Level fiveof the game, as we mentioned, was an independent subgame that Steve had, in fact, finished as an earlier, independent project.

To first approximation, Steve had simply and literally cut this game out of its original context and pasted it in the middle ofthe graphing adventure game. All the working structure of the game — menus, graphical display and hierarchical code — came, intact, in a working visible unit. Steve and Bill only needed to find an appropriate place in the adventure game to add the oldgame. Contrast this with systems that require linking code by linking the thread of activation within the loops and eddies of the "main program."

Trivial to perform and easy to conceptualize joining of old and new code is a vital first step. But then customizing and refining are equally important. Given that the old game was played squarely in the midst of the program structure of the new game, interlinking is made easy. For example, Steve and Bill shrank the invaders game box and added a command in itto throw the cursor out and reshrink the box if someone tried to enter the game before successfully completing the first four levels. (Boxer contains easy hooks to activate code on a user's entering, leaving, or changing a box.) This relied on a hidden global variable that kept track of the player's progress. Indeed, on exiting the game, Steve and Bill had a small segment of code to neaten up the space invaders game for a new player and to pass on the score the current player had achieved, upon which some of his/her future fortunes would depend. Strongly associating code with place makes an excellent modularity principle in Boxer's overall structure. It is especially powerful on modifying and extending for new and more elaborate contexts.

The People Mover

Among the nice pieces of video data we have from our year in sixth grade with Boxer is one that shows, from start to finish, a two hour programming project undertaken collaboratively by three students. The project was to program a "people mover" — a moving walkway such as found at airports — so that one could experiment with different speeds and directions of walking and walkway motion. Our intent was to create an engaging and programming-mediated encounter with some of the basic ideas of relative motion. Students did learn a fair amount about relative motion from this exercise, in interesting ways. The point here is only to note some of the things we learned about Boxer-mediated collaboration from this tape.

One of the students in the group was anexceptional Boxer programmer who had been using Boxer for the better part of a year. He, naturally, contributed substantially to the success of the group. But so did the others. In fact, about a third of the time the other two collaborators worked without him, and it is notable that the work continued nearly seemlessly through his departure and return. This was excellent indication of joint ownership.

Of the two non-experts, one had had Boxer experience. The other was a complete computer neophyte. Once substantial point about collaboration is that even the neophyte contributed significantly. Some of his contributions, in fact, were critical to the project; they concerned how to computerize (in arithmetic) some of the group's intuitive ideas about relative motion. For these, the important point about Boxer is mainly that the student understood enough of what was being programmed, and how, that he could make productive suggestions without prior programming experience.

The more general point, which we could see in all of the students' interactions, is that Boxer provides an extraordinarily rich and direct visual presence. Students can not only see a lot on the screen in terms of program structure (e.g., where the pieces are that accomplish various tasks) but also they can see how things operate. Boxer is nearly unique in allowing one to see at the same time the effect of a program and the programming structure that causes those effects.

Visible variables and pokable code allow easy access. This is familiar from "long distance" collaborations described above. The novel thing we could see in this context was how much the screen served as a place to point and explain, like a super (interactive) blackboard for students to coordinate their ideas and actions, make visible and articulate their contributions. We have noticed in other videos of joint programming how much a contribution the Boxer screen makes to a wide and effective communicative channel. Talking, alone, is less effective. Boxer expressivenessthus manifests itself also in real-time interaction effectiveness as well as on "asynchronous" collaboration.

Vectors

For our sixth grade class and subsequent high school courses on physics, we developed an extension to basic Boxer. In effect, we added vectors as a basic data type. With the addition, one makes a vector by pressing a key, like ordinary boxes. Vectors appear as interactive graphics boxes containing an arrow that one can resize and reorient using the mouse. Vector operations were also added. One can add two vectors and see the resultant, one can tell a sprite (turtle) to move along a particular vector, and one can name and change a vector like other Boxer variables. In addition, all graphics boxes in Boxer can be "flipped" to reveal another non-graphical presentation, which we chose in this case to show the vector's numerical components. These could be manipulated as well as the graphical presentation.

Vectors provided extraordinary resources for our whole teaching and learning community. We developed many tutorials and exercise microworlds out of vectors. The flexibility they afforded us is illustrated by one occasion in our high school course. We had had students take stroboscopic pictures of balls tossed and dropped in the air, scanned them in, and were prepared for students to study these motions analytically by driving sprites around over the images. Unfortunately, the lessons on two dimensional kinematics were not going as well as we had expected. We decided to scaffold the students' analysis with a tool that suggested how one should think about the motion and what results one might get. Using vectors we spent about an hour as a group designing and implementing the tool, which was used successfully the next day in class. To emphasize the flexibility and ease of modification Boxer-plus-vectors provided, one student suggested a rather different type of analysis and with the teacher's help succeeded in modifying the supplied tool.

Vectors infused the course more generally. We usedsimple vector programs to define and illustrate basic kinematics terms, a role that would ordinarily have been taken by algebra. In addition to exercise and tutorial microworlds, vectors spread to students as a central part of personal projects. It is rare that intellectual tools, like abstract vectors, can be concretized to the point that they provide such obviously helpful support for student-initiated activities.

Along with the general properties of Boxer programming that made vectors successful contributions to this community, this example illustrates one key property that, as far as we know, is unique to Boxer. Namely, Boxer vectors were complete, self contained interactive objects, with their own mouse interface, and yet they were as well first-class programming data objects that could be used in student or teacher extensions in the same way other built-in Boxer objects could be used. Vectors could be named, changed by programming statements, supplied as inputs and returned as outputs of programs. This dual citizenship — as simple screen-interactive objects and also as full-fledged, extensible computational entities — meant easy learnability and near-complete adaptability for students' and teachers' particular needs. As such, they are powerful tools that can grow effectively in particular collaborative communities to serve local purposes.

A Class Box

The last example is a very simple, but potentially powerful collaborative product. It relies on Boxer's simple, perusable spatial structure, easy composition of complete working units, and the ease with which text and hypertext annotation may be added. The teacher of a high school class using Boxer to teach about infinity and fractals decided to produce a class box. It contained:

a) all the tutorial materials the teachers had developed, in Boxer, to teach both Boxer and the mathematics of the course, including the exercises he assigned;

b) teacher comments on how students did with various materials in the course, including successes and difficult points;

c) all of the students' final projects, each in a standard form, which had working programs, textual explanation of mathematical analyses, and demonstrations. The students worked in groups of two and three, and the final projects of this particular class were visually impressive, artistically presented and generally mathematically cogent.

In addition to providing a motivating and informative class product, such a hypertext product provides excellent "dissemination" of instructional ideas to other teachers. Seeing student work is an excellent source of ideas and calibration of expectations.

The Future — Community Boxer

Boxer was not designed with collaboration explicitly in mind. However, we have seen after the fact that many of Boxer's learnability and expressiveness properties contribute to excellent code sharing, and to asynchronous and synchronous collaborations. At present we are considering extending Boxer to better allow multiple machine collaborations. Our goals are straightforward. We want to provide generic multi-machine collaborative support in the same way Boxer now provides general facilities for programming, personal or small group work, microworld and tool building. We want users to design and modify their own collaborative structures in the way Boxer users now develop, share and modify traditional Boxer materials. The means toward this end are also relatively clear. We intend to expand Boxer's current spatial metaphor and its sharing and hypertext structures across network connections. Users may "inhabit" the same Boxer space or share substructures arranged to support many forms of collaboration.

REFERENCES

diSessa, A. A. (1990). Social niches for future software. In M. Gardner, J. Greeno, F. Reif, A. Schoenfeld, A. diSessa & E. Stage (Eds.), *Toward a Scientific Practice of Science Education,* Hillsdale, NJ: Lawrence Erlbaum, 301-322.

Ploger, D. and Lay, Ed. (1992). The structure of programs and molecules. *J. Educational Computing Research*, **8** (3), 347-364.

Symbolic Programming vs. Software Engineering —Fun vs. Professionalism— Are These the Same Question?

Brian Harvey

University of California, Berkeley

I was asked to talk about "the role of Logo-like learning of programming in computer science education." The trouble is that I'm not sure I know what "Logo-like" means. It's an interesting word; as soon as we hear it, we all feel as if we know more or less what the person who says it means. We don't have the same urge to interrupt that we would if someone said, for example, "recursion-like." But do we really understand each other? It gets a little fuzzy, for me, when I think about it in more detail.

What Is a Logo-Like Language?

Let me give a couple of examples. First, I notice that in the conference announcement, the programming language ISETL is considered to be Logo-like. But in 1976, Seymour Papert dismissed APL, which is in many ways similar in spirit to ISETL, by referring to "languages of the BASIC/APL/FORTRAN family" [Pap76]. Logo, APL, and ISETL are all interactive languages that support recursive functions and are designed around mathematical ideas rather than around computer hardware capabilities. Logo is different from the other two in that its mathematics is meant to appeal to young kids, while both APL and ISETL present a more conventional, adult mathematics. What has changed to make APL non-Logo-like in 1976 but ISETL Logo-like in 1993?

As a second example, I'm not sure that my own work, even when I use Logo, is really "Logo-like." In Seymour's new book he characterizes the spirit of Logo as a promotion of Concrete Operational thinking from an interim stage of development, one that should be discarded as soon as possible, to a mode of thinking that is at least as valuable as Formal or abstract reasoning, even in adulthood. [Pap93]

When I read that, I was reminded of the history of the Logo IF notation. The early dialects of Logo, up to the MIT Logo for the Apple and its Terrapin descendents, use a BASIC-like IF-THEN notation:

```
IF :X < 0 THEN PRINT "NEGATIVE
```

I was part of the design team for Apple Logo, the first LCSI product, and I was, as much as anyone, responsible for the shift to the notation in which IF is a procedure with two arguments, a true-or-false value and a list of instructions:

```
IF :X < 0 [PRINT "NEGATIVE]
```

Our argument was that Logo should have a single, uniform rule for evaluating expressions: Every expression consists of the name of a procedure, followed by some number of expressions that provide the arguments. Logo evaluates those subexpressions, then invokes the named procedure with their values as its argument values. The advantage of the uniform notation is that it paves the way to user-defined control structures like

```
WHILE [:X < 0] [PRINT :X MAKE "X :X+5]
```

But there's no question that the LCSI syntax requires some explanation for beginning Logo programmers. By making this change, have we put general rules—abstract thinking—above the concrete specificity of IF/

THEN? Was this a non-Logo-like change? LCSI is Seymour's company, and they still use the newer IF syntax.

What Is Logo-Like Learning?

Both of these examples raise the question of what constitutes a Logo-like language. I was asked to address a somewhat different question, about Logo-like learning. But I'm just as confused about what that means. Let me describe two of my own experiences as a teacher, and let's ask to what extent each of them is "Logo-like."

The experience I enjoyed most, and of which I'm proudest, was running a computer lab at the Lincoln-Sudbury Regional High School, in Massachusetts, from 1979 to 1982. It was never completely clear what I was supposed to teach. The principal, who came from a business background, talked about "data processing." Some of the other teachers had picked up on the recently invented phrase "computer literacy." Computer science as a high school subject didn't really exist yet; I was myself most interested in computer programming, especially systems programming. The students, too, had a broad range of interests and expectations when they signed up for a computer course.

At first I tried offering different courses for students with different levels of interest, but I soon learned that most students didn't really know what they wanted until they had some experience within the courses. (Remember that personal computers had just barely been invented, and most students had no idea what it meant to use a computer, or to program one.) I ended up using a self-paced structure, in which there was one course for everyone, but each student could design his or her own curriculum within broad limits. Logo programming was one of the options. Every beginner was required to learn a little Logo and a little word processing; beyond that, anything was allowed.

We had a Unix-based timesharing system. At that time, Unix provided programming languages and application software far in advance of anything available on personal computers, such as the Apple II and the Atari 800. Partly because of Unix, and partly because I was spending much of my own time on system software development, the kids soon got the idea that the adult programming language was C, and so that's what all of the really interested students used. C is about as far from a Logo-like language as you can get! The C programmer deals very directly with issues of data representation within a computer, such as word length and memory allocation.

On the other hand, what felt "Logo-like" about this situation was that the kids were engaged entirely in real work, using the computer for their own goals and not jumping through assigned hoops. (Some students took on the task of helping with the school's administrative computing; this work was somewhat less self-directed, but even more real, than the student-initiated projects.) A lot of effort went into video games, using the limited character-based graphics of the VT-100 terminals. One student wrote an EMACS-like text editor, called JOVE, that's still widely used at Unix sites. Not all students took to programming, but many more learned to use the word processing facility for their papers for other classes. (One year I had three or four students who were also taking a chemistry class, in which the teacher had said they could bring one page of notes into the final exam. These students collaborated on designing and printing a page, using the smallest available font, to cram in all their notes!)

One reason that Logo itself wasn't popular among these students was the technical limitations of the facility. Graphics terminals were very expensive in those days. We did have a few Atari 800 computers that we used as graphics terminals (Atari Logo hadn't been invented yet), but they could only handle 40-column text, so they weren't good for much other than graphics. The Logo interpreter we cobbled together for Unix was very slow. There was no hardcopy graphics capability. We did also have some Terrapin floor turtles, but they were unreliable.

Another Logo-like aspect, however, was the social interaction in the lab. There was no formal class time; people would come and go to use the machine or just to hang out. An organization of students took responsibility for running the lab, and its members were given keys to the room so that they could use the

machine without supervision evenings and weekends. (Nothing was ever stolen, and there was very little breakage.) Kids set up their own working partnerships; more experienced kids taught newcomers. Most of my effort went into this meta-curriculum rather than into teaching programming. It didn't always go smoothly; there were fights, violations of privacy, and the like. But we all saw these problems as community problems, not just as something for me, the teacher, to deal with.

(To make this focus on community sound a bit more serious, and perhaps more "Logo-like," let me discuss it in Piagetian terms for one paragraph. By high school age, kids are supposed to be at the highest stage of cognitive development, the Formal Operational stage. In a certain sense, there is nothing more for them to learn, cognitively. On the other hand, high school kids are still far from the top of the scale in moral development. So it makes sense that issues of community and responsibility should be central to their education.)

To sum up, my high school computer lab felt to me very much a Logo-like learning environment, although Logo itself was a minor component, and although many of its characteristics are ones that Seymour Papert lists as non-Logo-like: the equipment was concentrated in one room, the curricular focus was on computing rather than on math or other areas, and the style of work was primarily the formal, abstract, goal-directed style that Seymour criticizes as male-oriented.

I currently teach undergraduates at the University of California at Berkeley. Our introductory computer science course was developed at MIT [Abe85]; one of the developers is Hal Abelson, a Logo old-timer who led the first microcomputer Logo development projects for the TI 99/4 and the Apple II. The course uses Scheme, a dialect of Lisp that's similar to Logo in many ways. (At Berkeley we've even modified Scheme to add the Logo word and sentence primitives, so that our students can write Logo-like language processing programs.)

Is this course Logo-like? How I feel about that question depends on my standard of comparison. In a recent paper [Har91] I compare the Abelson and Sussman course with more traditional, Pascal-based, software engineering courses; by that standard, our course feels very Logo-like indeed. The traditional course is full of commandments about programming style, telling students what they should and shouldn't do. The traditional course must spend a lot of time on programming language syntax; our (dare I say) Logo-like language has a simple syntax that takes hardly any instructional time. Students work on programming projects that are designed to illuminate big ideas in computer science: ideas that help the students write programs, rather than abstract theoretical ones.

But compared to my high school experience, this course doesn't feel Logo-like at all. The style is very different, first of all. I give lectures to groups of 300 students. Then the students go to the computer lab to do their assigned projects, but I'm generally not in the lab with them. (We do have teaching assistants in the lab at scheduled hours.) The projects are assigned, rather than student-initiated.

In the six years I've been at Berkeley, I've taken some small steps toward a learning environment that feels slightly more Logo-like. For example, we used to assign weekly homework with exercises from the textbook. Now I assign the exercises, but don't collect them, so that it's clear to the students that the exercises are for their own benefit and not for evaluation purposes. I've instituted cooperative learning, redesigning the programming projects to be done by groups rather than by individual students. These are very tiny steps to be discussing at a Logo conference! But they go very much against the grain of the context within which I teach. The introductory course is part of a curriculum ladder, and later courses depend on "covering" the curriculum. The students, in these bad economic times, tend to be much more worried about competing for jobs than about learning.

The irony in all this, as we'd expect from our Logo experience, is that the students who end up doing best, even in the job market, are the ones who can find the courage to forget about jobs and grades and jumping through hoops, and treat my course as if it were a more Logo-like environment, getting excited about the new ideas and pushing beyond the required work to invent and explore new programming techniques.

When I was an undergraduate myself, I was lucky enough to fall in with the gang at the MIT Artificial Intelligence Lab, which was for me the perfect prototype of a Logo-like environment even before there was Logo. On the one hand, everyone was engaged in real work, pushing the state of the art in computing, but on the other hand, nobody felt any pressure about it. It was all for fun; you could try anything, and if it didn't work out, you just tried something else. (But don't forget that the hacker spirit at that lab is also the prototype of the style of work that Papert and Turkle call androcentric.)

How Are the Two Related?

After this long preamble, I think I'm ready to talk about the question in the title of this paper. There are two different ways of looking at a computer science lab that both seem connected somehow to the Logo-like scale. The more obvious one is about style of work; as I've been saying, from that perspective, my high school lab felt extremely Logo-like, and my university course doesn't. This is the question that I've inadequately represented as "Fun vs. Professionalism" in my title.

The other is the more technical question about curriculum and programming language. There is a very sharp distinction between the software engineering approach and what I think of as the artificial intelligence approach. In the software engineering approach, students are being trained primarily to be disciplined in their programming style, so that their code can easily fit in with other people's code. The curriculum is full of rules; the programming language used is always one with strong typing and similar restrictive rules. Software engineers talk about object oriented programming, for example, primarily as an advance in information hiding: An OOP language is one in which procedures as well as data have very tightly limited scopes. The AI approach replaces this moralistic tone with the cultivation of a sense of wonder at the ways in which thinking harder about programming can pay off in increased expressiveness. In this approach, there isn't one correct way to program, but rather a world full of programming styles and structures, each one appropriate for certain kinds of problems. On this view, for example, object oriented programming is about a metaphor, expressing a kind of parallelism in which each datum has its own intelligence, rather than about discipline.

This division, the one I've called "Symbolic Programming vs. Software Engineering," is fundamentally about the philosophy of computer science. It seems quite different from the question about style of work, but I think I can find some similarities.

One difference between a Logo microworld and an ordinary piece of educational software is that in the Logo version, the program itself is available for inspection and modification, not a black box. In the study of computer science, a programming language is itself a microworld. That's why the Lisp tradition of introducing even beginners to the workings of the Lisp interpreter is Logo-like: By examining and modifying the interpreter, we turn the Lisp microworld into a glass box instead of a black box.

A crucial part of a Logo classroom is that the learners have permission to explore using their own individual styles. There isn't just one right way to program. Similarly, in the symbolic programming approach to teaching computer science, there is no orthodoxy about top-down structured programming. Top-down is one fine approach, but so is tool-building without a specific goal in view.

Symbolic languages, and languages with relaxed attitudes toward data types, allow for programming projects about sentence generation, carrying on conversations, the automated psychotherapist, and so on. These projects are very much in the Logo tradition, and in contrast with the heavy emphasis on numeric computation that's typical of the software engineering languages.

Logo-Like Languages Revisited

Finding these examples in which the technical details of a programming language and a curriculum seem to be connected with working in the Logo spirit gives me the courage to return to the question of languages and hope that in so doing I'm still addressing the question that was given to me.

In my 1982 *Byte* article, I listed several characteristics that seemed to make Logo special. [Har82] Let me go through the list and comment on where we stand in programming languages today.

Procedural. We seem to have won this battle. Even BASIC, in its modern dialects, allows a modular program structure. In fact, procedural languages are now the right wing of language design, in contrast with newer approaches such as functional, object-oriented, and logic programming. Some recent versions of Logo address the issue of parallelism, although that newer Logo technology is not yet, I think, widely understood among the people actually using Logo in schools.

The issue today that corresponds to the one about procedural programming languages a decade ago is that many people don't want to deal with languages at all. Instead, modern software provides a graphical user interface, with mice, buttons, menus, and so on. Even Logo has not ignored this trend; LogoWriter on the Macintosh is full of immediate-mode graphical controls for things like moving and turning the turtle by hand. I think the battle we have to fight is to maintain the position that immediate-mode control isn't enough, because it doesn't allow for the abstraction of regularities and patterns of work. There must be a way to create control structures that can carry out the same operations that can be performed graphically.

Interactive. This is a crucial issue, and the terms of the debate have changed dramatically in recent years. Originally, there was a clear-cut distinction between interactive languages such as Logo, Lisp, and BASIC, and non-interactive languages like C and Pascal. The style of work in the non-interactive languages was awful. Programmers had to write their programs in an environment entirely separate from the language itself, then go through a compilation process, and finally run the program, often with no runtime support for debugging or modifying the program. Their excuse was that a compiled program runs much faster than an interpreted one, and indeed the Logo interpreters of that time were unpleasantly slow.

The characteristics of the underlying languages have not changed. But today's Pascal or C programmer works with an integrated development system. There is a text editor that understands the syntax of the particular language, so that it can check the form of a program as you write it. There is interactive debugging, generally using a window system effectively so that the debugging information is visually separate from the output of the program itself. You still can't walk up to a C compiler and ask it to print 2+2, and yet today's C programmers are likely to complain that a Logo programmer suffers from an insufficiently interactive working environment! For myself, I still find interaction with the actual programming language more powerful than these auxiliary interactive tools, but I'm not sure whether that's just because I'm accustomed to an old-fashioned style of work. It can certainly be argued that commercially available versions of Logo have moved backward in their provisions for debugging support; Apple Logo allowed the user to enable an automatic PAUSE in case of an error, but LogoWriter doesn't.

Recursive. In the early Logo work, learning about recursion was a crucial step in intellectual development. Ideas like the little person model of procedure invocation were invented mainly to help in understanding recursion. In one sense, this battle seems to be behind us; every modern programming language allows recursion. But the conventional wisdom in computer science education is that recursion is too costly for ordinary use; students are taught to prefer iterative language mechanisms. Meanwhile, except for drawing fractals, Logo users rarely use recursion in a way that couldn't be handled by conventional iterative language constructs. (Indeed, part of my technical work has been the introduction of iterative constructs to Logo.) I think it's still worth remembering that recursion is a single mechanism with remarkable expressive power. For example, these days it's easier to draw a circle in a canned drawing program than in Logo, because the canned program has a dedicated circle feature; but the drawing program can't use its circle feature to draw spirals or stars.

List processing. Logo is still out in front on this point. Even other dialects of Lisp don't make the manipulation of words and sentences as convenient as it is in Logo. If you want to conjugate a verb or take the plural of a noun, Logo is the language of choice. On the other hand, I think it's still true that most actual Logo users consider list processing to be arcane and difficult. LogoWriter fans often prefer to manipulate text by carrying out a sequence of text editing commands on the representation of the text on the screen, rather

than apply recursive functions to the text taken as data. My hope is that a good collection of higher-order functions like MAP and FILTER can make list processing seem more palatable.

Untyped. This is probably where you'll find the most heated arguments between symbolic programming enthusiasts and software engineers. The lack of type declarations in Logo, and in Lisp generally, makes programming enormously easier to learn; the software engineers, though, think it's sinful. The real argument for typed variables is that the compiler can produce more efficient programs, but the theoretical argument is that restricting a variable to a particular type makes for more disciplined programming. (It also makes for ludicrous user interfaces, in cases where the natural thing is to read numbers until the user says something like "DONE" to indicate the end of the numbers. Instead, the typed languages make the user type something like "9999" or "-1" to indicate the end.) Some computer scientists will say that the advent of object oriented programming has settled this question in favor of strong typing, but the example of Object Logo shows that that isn't so.

Extensible. We are beginning to win this battle when it comes to application software. Many large programs these days include a "scripting language," such as Hypertalk as the scripting language for Hypercard. But mainstream programming languages are not generally viewed as bases for building language extensions, except in the sense that you can write a full interpreter or compiler from scratch. This property is still very important in Logo practice, because it's the technical basis for microworlds. Any Logo programmer (not just the implementors at the software companies) can develop an extended Logo with extra primitives for some microworld.

Interesting applications. New applications are invented very fast these days. Logo is in the lead in some cases (robotics, with Lego-Logo), hustling to catch up in others (MIDI support recently added to Object Logo), and not really in the running for some (real-time animation, unless you have a massively parallel computer on which to run *Logo). The proliferation of special-purpose software tuned to each of these application areas raises the question of whether we can reasonably expect any general-purpose programming language to be a good match in every case. In the old days, the appeal of Logo programming wasn't so much the act of programming itself as the results you could tease out of a computer; today you can get more spectacular results without programming at all. This is why there has been a vast decrease in the amount of time spent on programming in schools in the past few years; students are using application programs instead. Is this good or bad? In a way the issue is similar to the one that we argued about when Sprites were introduced in TI Logo: "It's too easy to get pretty pictures on the screen in immediate mode, so kids won't learn to think procedurally."

User-friendly. In the old days we made a big fuss about things like the text of error messages and the names of primitives. This tradition is still alive in Logo; the switch from OUTPUT to REPORT in LogoWriter is an example. But a new generation of computer users accustomed to point-and-click interfaces would question the user-friendliness of any Logo environment. I think we have to admit that Logo syntax, although better than Pascal, is still far from transparent. Can we develop more compelling notations without sacrificing the essence of programming? I guess this is what Boxer is about, in part.

No threshold, no ceiling. We are starting to see a division of the Logo using community. Products like LogoWriter are deliberately aimed at the young kids who make up most of Logo's actual audience. Object Logo is a high-end version for more advanced programmers. I've found it necessary to implement my own version, Berkeley Logo, just so that there will be a more or less unchanging interpreter for the language used in my books. In principle I suppose this is a good thing, because it allows us to extend the limits of Logo usability in both directions. But I wonder whether Logo's rather small audience can support so many separate development teams.

To sum up, some of the technical issues that seemed important a decade ago in distinguishing Logo from other languages are no longer at issue, such as procedural organization and recursion. Others burn as brightly as ever: list processing and untyped variables. Most interestingly, others have changed into different issues, especially the ones about user interfaces, such as the definition of interactiveness.

REFERENCES

Abelson, Harold, and Gerald Jay Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs,* MIT Press/McGraw-Hill, 1985.

Harvey, Brian. "Why Logo?" *Byte,* vol. 7, no. 8, August, 1982.

Harvey, Brian. "Symbolic Programming vs. the A.P. Curriculum," *The Computing Teacher,* vol. 18, no. 5, February, 1991.

Papert, Seymour. "An Evaluative Study of Modern Technology in Education," MIT Logo Memo 26, June, 1976.

Papert, Seymour. *The Children's Machine,* Basic Books, 1993.

Exploiting Logo-like Learning Environments for Learning Mathematics

Celia Hoyles

Institute of Education,
University of London

Logo has not fulfilled its potential? Discuss.

One can imagine this question on examination papers around the world! How would we answer it from a perspective of over a decade of research and development of Logo in schools? From the point of view of LogoMathematics, we are all well aware of the wonderful work that students can do, the ingenuity they can display - but we also must face the inescapable fact that simply by interacting with the computer, students' acquisition of prespecified mathematical ideas is at best haphazard. Students have their own goals, agendas, and beliefs; they have differing experiences and expectations, all of which colour their reading of the Logo work, and the meanings they construct of the activities in which they are engaged. Thus computer microworlds structure and guide students' developing conceptions but simultaneously students mould the software tools to fulfil their own purposes. That is to say, students' activities are constrained by the tools they have available, by the syntax and semantics of the software *but* (hopefully) they can format the medium so that it can be made to express their own mathematical ideas.

Taking as a starting point the theoretical position that concept and representation are inseparable, it follows that the actions through which representations are constructed form the means through which mathematical ideas are expressed. We (Richard Noss and I) have termed such expressions, *situated abstractions.* A situated abstraction is a student articulation of a mathematical generalisation which is *constructed* by the student through an inductive process and is *expressed by the student in terms of the medium of construction.* A situated abstraction is thus a 'legitimate' but 'alternative' form of representing a mathematical concept. A Logo example would be a definition of a regular polygon by means of a Logo procedure. The problem is how can we 'help' students construct their own situated abstractions and subsequently build on these by the processes of recontextualisation and synthesis.

These issues are not, of course, unique to computational environments - they are at the heart of constructivism. However with Logo-like software, that is software which involves action plus representation (my definition!) students can use mathematical ideas *before* they have fully discriminated the relationships involved, and moreover can use them in ways which push in the direction of mathematical formalisation. Thus Logo-like software can play the role of an action-based means of communication and as such can offer a unique opportunity for the expression of mathematical ideas. As a bonus, the computer offers a window on to student conceptions and a mirror of our interventions (see Hoyles 1993 for a description of the background to these ideas). While working with Logo-like software, it is hard for students 'to fool us' by simply enacting 'what the teacher wants them to do!' and, at the same time, it is difficult for us to avoid noticing mismatches between our agenda and that of the student.

So how can we exploit Logo-like software for learning mathematics? Much can be done by personal exploration, but in the real world of schools and classrooms it is rare that computers are personal technology - tools owned by the student and used according to their whims and needs. In most schools it is rare to find more than one computer per class and, as a consequence, learners often work together whilst interacting with the computer. How can we exploit groupwork with computers for the purposes of mathematics learning? In the remainder of this paper I will point to three possible avenues to follow - each with its own advantages and disadvantages.

Collaboration and Learning

Research within the peer collaboration paradigm provides a cognitive analysis of group interactions using theoretical frameworks from developmental and social psychology to explain peer facilitation (e.g. Perret-Clermont, 1980; Forman and Cazden, 1985). Socio-cognitive conflict and negotiation of joint action are two mechanisms by which social interaction is said to lead to beneficial effects on an individual's cognitive functioning, and more particularly on conceptual understanding in mathematics. Recent research in the psychology paradigm has introduced the computer into the experimental setting to mediate the social interaction through control of the computer: for instance, using the computer to manipulate the presentation of the task or to restrict the available forms of response (Blaye, 1988; Davidov, Rubtsov and Kritsky, 1989). However, this branch of research has little to say for mathematics learning specifically as the software used does not provide the students with a means of expression - is not Logo-like!

Many investigations of students working collaboratively with Logo have been undertaken (see Hoyles and Sutherland 1989 for examples). There is evidence that *after a period of familiarisation with the software,* students:

- come up with projects of a mathematical nature,
- negotiate goals and processes and have to justify their choices to their partner,
- develop a shared language for communicating the actions to be taken,
- brainstorm solution strategies,
- debug from a range of perspectives.

We have found though that a *group etiquette* often develops over time where each student adopts a role in the collective enterprise according to their expertise and preference. This provides an efficient means of 'getting the answer' but may mitigate against any individual appropriation of more global mathematical ideas as illustrated in the following episode.

Sally and Janet worked happily and productively together for three years during the Logo mathematics project (Hoyles and Sutherland 1989). Despite interventions throughout the period of their computer work, we did not appreciate until we conducted an individual interview the extent of Janet's uncertainty of the meaning of turtle turn - how it was manifested on the screen, how it related to paper and pencil work with angles. The one-to-one discussion revealed some strange interpretations of what was going on. In order to try to understand how and why this had happened, we analysed the video transcripts[2] of the work of the pair and discovered - rather to our surprise - that *all* the decisions about turn had been taken by Sally while Janet's role had been one of 'programmer' and 'brainstormer'. The girls seemed to have agreed implicitly on a 'division of labour' which undoubtedly contributed to their differential understanding of particular concepts. Moreover, since the partnership worked so efficiently and harmoniously this division of responsibility was hidden from the observer.

We have noticed other instances of students developing a group *etiquette* which can work against individual learning. It is also the case that peer interaction is not always successful - there can be problems of communication, refusal to discuss and share or even to try to take another's point of view. How can we encourage students to explore together, to talk about their ideas? How can we overcome the legacy of school mathematical practice where competition and individual achievement is central and begin to create conditions where groupwork is valued? To attempt to address these questions I turn to another theoretical position, that of the French didacticians.

The French didacticians present a particular framework in which to consider the social aspects of learning mathematics (see for example Laborde (1989) for an overview of this work). In this paradigm, groupwork amongst students is classified as an 'adidactical' situation - one taking place in classrooms in order to promote learning but one without teacher input during the actual learning process. They argue that in such situations, the means of solution and the information as to its validity must be available to the student from the situation - that is, from feedback from peers or context.

The French group also distinguish 'formulation' situations as those where mathematics is used and expressed - situations which are inevitably social. Laborde (1993) distinguishes two modalities of formulation situations; the first modality takes the social dimension as a priori by means of a communication aspect built into the activity, that is an asymmetry between students with respect to some mathematical information which has to be 'sent' from one to another; the second modality takes mathematics as the starting point and the social aspect is 'added' by the stipulation that a common solution is required. I illustrate both of these modalities by reference to work we have done in Logo Mathematics. The examples also serve to illustrate how the computer-based work can be interweaved with reflection and discussion away from the computer.

Communication and Conflict

In the mathematics education literature it is well-known that students tend to solve 'difficult' proportional problems by adding rather than multiplying. We devised a group task to address this conceptual issue. It involved an element of :

- communication - so student strategies had to be made explicit,
- conflict - so students had to come together to reflect upon and discuss their different solution paths.

The format of this activity was a game for a group of four students split into pairs (each initially designated as pair A). These pairs were asked to produce a figure in proportion to an original house whose Logo program was provided and whose image is shown in Figure 1. They had to record their strategy and the lengths of their new figure. Pair A then passed the value of one length of their house to pair B and then challenged them to find out how the new house had been generated.

Thus, the first stage of the game was designed so that pupils (as pair A) would construct a similar shape to house and, if they used a scale factor, would not choose one which was obvious (i.e. not integral) in order that it be a challenge for pair B. It was important too that their method of construction was 'captured' in a Logo program so that it was available later. In the second stage, each pair (now all acting as pair B's) had to 'find the rule' which the opposite pair had used to construct a new house, given one of its lengths and the dimensions of the original. They then could use this rule to calculate all the lengths of the enlargement.

In the final (off-computer) stage of the game, the group of four students who had constituted 'opposing' pairs (and who had each played the double roles of Pairs A and B) compared the numerical sizes of their houses. If these were different, the pairs were forced to defend their decisions, and explain and justify the strategies adopted - strategies clearly exemplified in their Logo programs.

I make a few observations as to the functioning of this group activity. First, we found that the *construction* of a house (as pair A) often called up a different strategy than that adopted to *interpret* the opponents' house. For example, Toby and David constructed a larger house by *multiplying* each of the sides by 2.379. However, when their opposing team (Paula and Jane) presented them with a larger house whose vertical side was 53.95 (the original length was 20), Toby and David edited their HOUSE procedure changing the first FD 20 to FD 53.95 but added 8 to the next input to FD (the smaller house had a 'roof length' of 28, 8 more than 20), and fixed on this additive strategy for the remaining lengths. This left them with a problem for the base of the house, but despite their apparent understanding of proportional strategies in the construction part of the activity, they happily found the length (106) by a 'homing-in' strategy - just adding more FD's till it closed up!

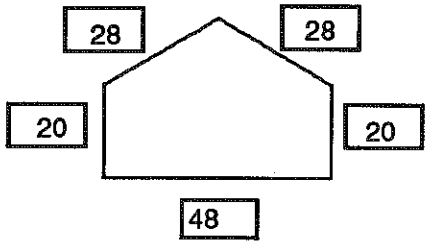
Second, the communication aspect did 'work' in most cases. Pair A had a responsibility to provide pair B with some specified information and pair B to provide them with feedback. In the case discussed, this communication worked as both pairs played by the rules. Additionally, conflict was provoked and this conflict was noticed and worked on by the group. When the group of four came together they were confronted with two different answers and strategies! The girls had constructed their house by multiplying by 2.6975 and the boys by adding 8 and adjusting the base. The girls at first thought they had made a mistake and

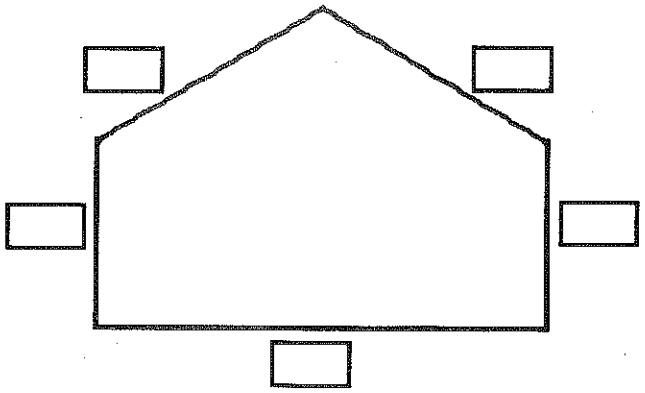
PAIR A NAMES_____
PAIR B NAMES_____

EDIT [HOUSE] to make a new house which is still in proportion (don't forget to change the name).

You will tell Pair B one length and they will have to use this to calculate all the lengths.

But first fill in all the lengths of your new house **and check it out on the computer.**





TEAR OFF .....

PAIR A: NAMES_____

PAIR B: NAMES _____

Calculate all the missing lengths for Pair A's house and fill in the gaps

How do you think they made the calculations?

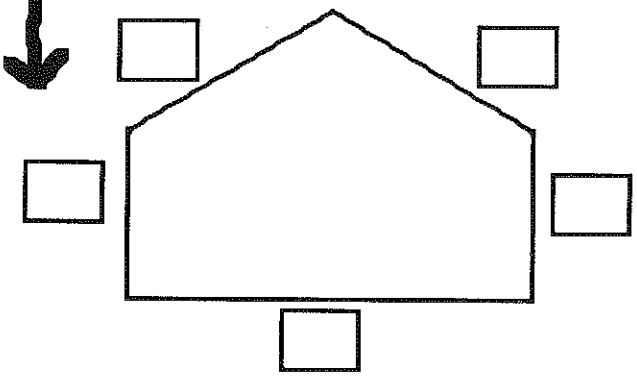
.....

.....

.....

.....

Pair A fill in this length and pass to Pair B



Compare with Pair A

Figure 1: A Group Activity: Find the Rule

checked all their multiplications. Then, convinced that they were not in error, they explained to the boys what they had done. David then changed sides and supported the girls - recalling that he and Toby had in fact constructed their own house in this way! Toby needed more convincing but some work on the computer helped him to see the others' point of view (for a detailed description of this work see Hoyles and Noss, 1992).

We have to be careful not to overgeneralise from these case studies. Conflict does not always occur. It is noteworthy that quite frequently students sometimes seem quite happy to be inconsistent - after all consistency is only part of a mathematical culture. Additionally, communication does not always work out as we plan. We can point to cases when the rules of the game are not apparently clear to the students - or indeed circumvented. A case in point is illustrated below:

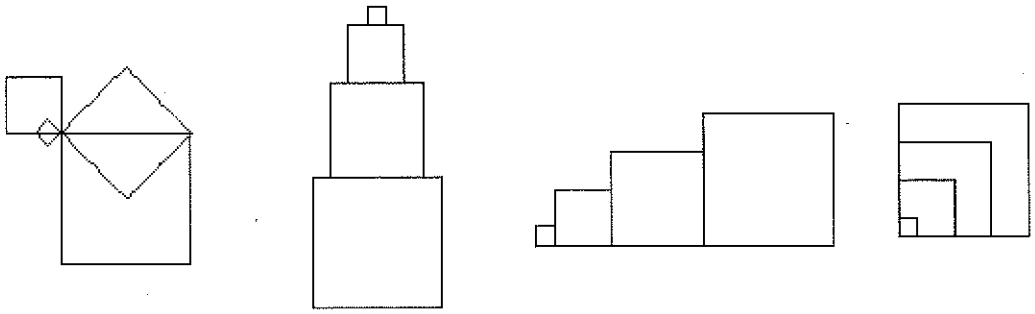


Figure 2: The Variable Squares Task

Students, working in pairs, were asked to construct programs for each of the patterns illustrated in Fig 2. Each pair was then required to communicate the program of *one* of the patterns to another pair in their 'team' in such a way that the latter would be able to identify which figure had in fact been drawn. The idea was that both the *construction* and the *communication* aspects of the activity would provoke the students to come up with a modular program - one which clearly differentiated the different-sized squares and the interfaces between them. I describe the work of two girls working on this task.

When Sally and Janet came to program the first drawing, they wrote a long string of sequential commands which they then split into arbitrary subsequences. As we expected this proved to be hard to build without errors, hard for the students who received it to follow and impossible for them to identify the figure which would be drawn. We asked the girls why they had constructed their program in this way and discovered that the reason could be traced to their alternative interpretation of the communication situation. Sally and Janet maintained, *in contrast to us*, that they thought that a list of sequential commands would be easier for the other pair to interpret than a modular design! We also wondered if this was the complete story and whether they had actually wanted to make things hard for the others in the spirit of winning a game - they would find the other pair's figure before their figure was identified!

This story shows how difficult it is to interpret student behaviour, to set up situations where there is a congruence between our aims and those of the students. In the next section, I turn to an approach which asks students to come up with a mutually acceptable solution to a mathematical task involving computers - the second modality in the French paradigm of a formulation situation.

Structured Group Tasks and Negotiated Consensus

Over the past four years, we have been researching groupwork with computers. We have set out to design tasks for groups - tasks which exploit the potential of both peer and computer interaction for learning mathematics. In addition to the theories mentioned earlier we drew on the work around cooperative learning methods, influential in the U.S., which involved restructuring the learning environment to emphasise peer interaction and within-group interdependencies rather than the individual or competitive goals of the traditional classroom (e.g. Slavin, 1983; Aronson, Bridgman, and Gellner, 1978; Johnson and Johnson,

1975). In the tasks we developed we set out to capitalise on computer work as a medium by which situated abstractions would be constructed. These would then serve as a means and language of communication amongst the group so that each individual contribution could be the subject of group reflection (and possibly debugging) in the pursuit of a common group goal.

Our task design had three basic features:

- a set of parallel components to share amongst the group members and to be constructed at the computer (called local targets);
- a network of mathematical ideas to be discussed across and away from the computers (called global targets).
- one group outcome

Since the local targets would be negotiated at the computer, each pair had the opportunity to express their mathematical ideas in their own way. No structure or form of representation was pre-specified. This meant that all students were able to contribute to the group goal. They were aware that their contributions were necessary for group success since their empirical 'findings' from the computer work provided the basis from which the group could come up with mathematical generalisations. Thus, the design optimised the chances that that everyone would contribute, everyone would feel committed.

Our tasks were designed with groups of six in mind, using up to three computers. Group size is not crucial but the the ratio of students to computers probably is - relaxed and easy access to the means of construction and debugging is important. We designed several tasks with the same infrastructure, using different software - but software which was Logo-like and provided the possibility of action-plus-representation at different levels of sophistication. I will provide just one example of a group task using Logo, called the Spokes task.

The Spokes task was designed to assist pupils in coming to comprehend the idea of 360o; it was about rotational symmetry and the geometry of complete turns. Each pair of students were expected to construct at the computer a radially divided geometric shape, a spoke design (see Fig 3), to identify some of its features and to complete the appropriate part of the table (shown in Fig 4).

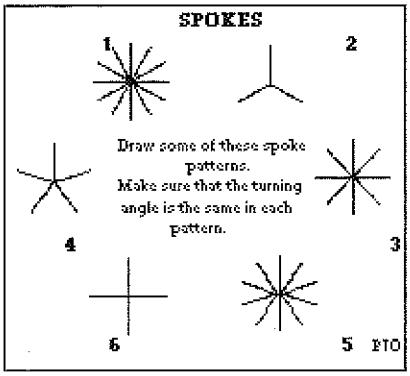


Figure 3: Constructing at the computer

	NUMBER OF SPOKES	TURNING ANGLE
1		
2		
3		
4		
5		
6		

Figure 4: Table of pupil results

When the table was completed, the group were asked to come together to discuss whether they could identify a general relationship between the two variables in the table (see Fig 5). They could use the computer to validate or debug their hypotheses. They then could split up again at the computer to make dandelions - using all the different spokes in more complex designs and again applying the idea of 360 (see Fig 6).

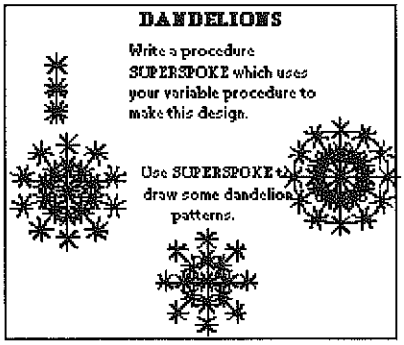


Figure 6: Dandelion designs

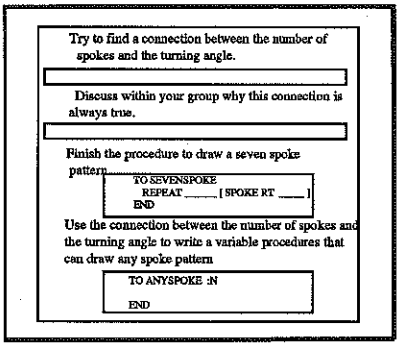


Figure 5: Making generalisations

This task proved most successful; students discussed and argued both at and away from the computers and showed evidence of commitment to achieving a collaborative outcome. However, there were distinct variations in approach which need to be explained. Students organised themselves, the task and the available resources – worksheets and computers – in very different ways. We identified three styles of organisation, *integrated, fragmented and connected*. Integrated styles most closely resembled the way the task had been planned, with local targets shared out and global targets negotiated in the quest to reach a one group goal; in fragmented styles, the group split up and rival subgroups strove to construct the most impressive computer products; and finally in connected styles, the group split up with subgroups again working towards separate goals but channels of communication were maintained through which help was given and task demands discussed. These three organisational styles are illustrated schematically in Figure 7.

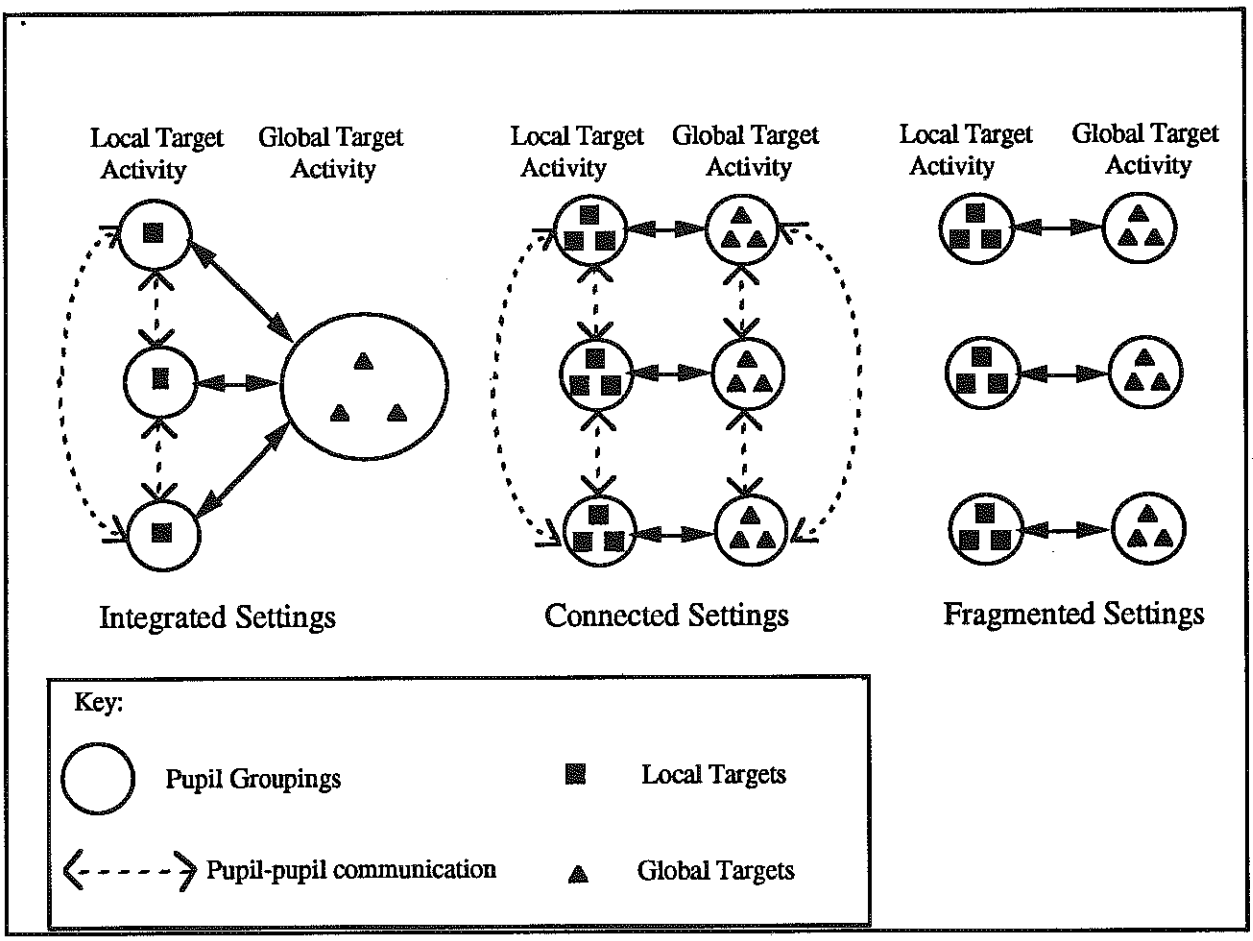


Figure 7: Styles of Organisation

Table 1 presents the configuration of process variables which characterised each style of organisation.

ORGANISATIONAL STYLE	PROCESS VARIABLES	
	Local Target Sharing	*Global Target Sharing*
Integrated	Yes	High
Connected	No	Middle or High
Fragmented	No	Low

Table 1: Process Variables characterising Styles of Organisation

So can we say anything about how to organise successful groupwork for mathematics? First we have be clear what we mean by successful - in terms of group outcome or individual learning. We termed the first productivity and the second effectiveness and found that success in one aspect was not necessarily associated with success in the other but high productivity and high effectiveness were likely to be associated with an integrated style of organisation. In contrast, fragmented styles were much less successful. We probed more deeply into the composition and functioning of the groups and we identified a number of features that distinguished successful and unsuccessful ways of working.

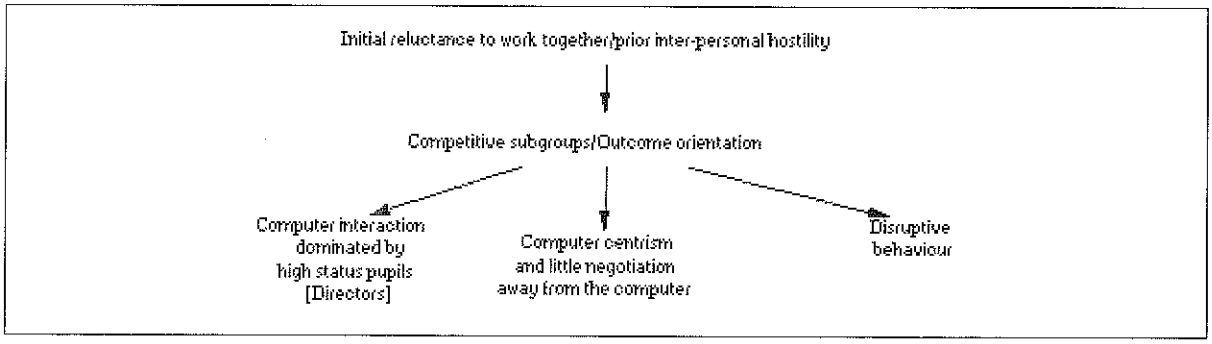


Figure 8: Unsuccessful group settings

Figure 8 characterises unsuccessful group settings - those which were both unproductive and ineffective. Antagonism between boys and girls tended to result in a fragmented style of organisation with single-sex rival subgroups working towards separate outcomes. The provision of several computers offered a means to avoid interdependencies through the atomisation of the group (the paradox of personal computers?). This fragmentation was associated with a centration on computer products and a curtailment of negotiation both at and away from the computer. The visibility of the computer screens undoubtedly fuelled these tendencies, and as pupils became aware of their relative progress (or lack of progress) disruptive activities became more frequent.

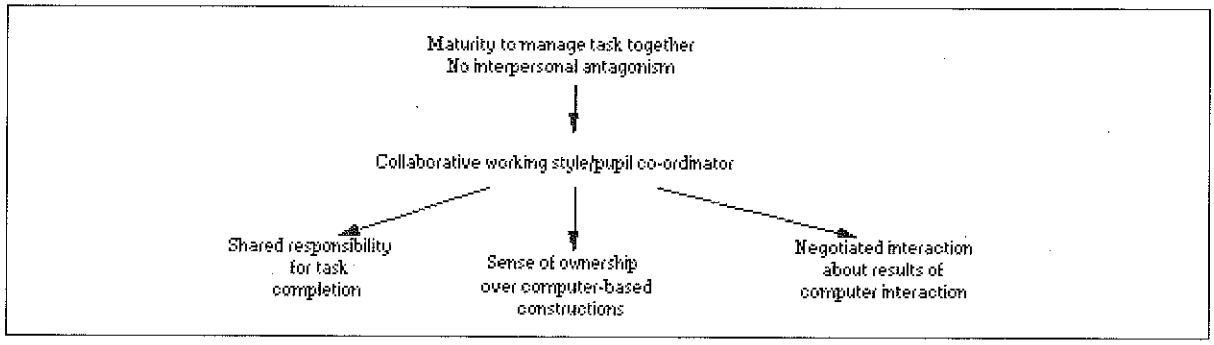


Figure 9: Successful group settings

Group settings with the characteristics shown in Figure 9 tended to be both productive and effective. Structured interdependencies set up through student acting as group co-ordinators ensured a shared responsibility for task completion: all students discussed the group outcome together after negotiating their individual contributions. The availability of several computers and the type of software used gave the pupils the opportunity to develop ownership over their ideas and express their ongoing mathematical activity in a formalised language. Thus, the group, the task design, the available resources and the type of software interrelated to form a successful balance between interdependence and autonomy.

Concluding remarks

If we take the position that the social dimension in mathematics learning is part and parcel of learning, and if we recognise that Logo and Logo-like software can serve as a medium for using, constructing and communicating mathematical ideas, we must draw on theoretical positions which help us to exploit these potentials in a more systematic way than is generally observed in the classroom. *In this paper I have suggested that successful groupwork with computers can be facilitated by Logo-like software which provides pupils with the opportunity to act upon and express their ideas formally - that is to articulate situated abstractions - and task design which provides a forum where these ideas can be shared away from the computer.* The extent to which any group is able to exploit these learning opportunities depends on circumstances outside the domain of mathematics education – on the social relationships in the group rather than the tasks, software or the mathematics.

Successful groupwork with computers stems from achieving a fine balance between pupil interdependence and pupil autonomy – a balance which can be affected by the social norms in the group, the individual personalities of the pupils. It is self evident that no simple solutions can be put forward to guarantee universal success - but this does not mean that we should not seek to optimise the chances by theoretical analysis of the influential factors.

REFERENCES

Aronson, E., Bridgman, D.L., and Gellner, R. (1978) The Effects of a Co-operative Classroom Structure on Student Behaviour and Attitude, In Bar-Tal, D., and Saze, A. (eds) *Social Psychology of Education: Theory and Practice,* John Wiley, New York.

Blaye, A. (1988) *Confrontation socio-cognitive et résolution de problème,* unpublished doctoral thesis, University of Provence.

Davidov, V.V., Rubtsov, V.V. and Kritsky, A. G. (1989) Psychological Foundations of the Learning Activity Organisation based on a Computer System Usage, *Children in the Information Age,* Sofia, Bulgaria.

Forman, E., and Cazden C. (1985) Exploring Vygotskian Perspectives in Education: the Cognitive Value of Peer Interaction, In Wertsch, J. (ed), *Culture, Communication and Cognition: Vygotskian Perspectives,* CUP, Cambridge.

Hoyles, C., and Sutherland, R. (1989) *Logo Mathematics in the Classroom,* London, Routledge.

Hoyles, C and Noss R (1992) A Pedagogy for Mathematical Microworlds, In *Educational Studies in Mathematics,* 23 pp. 31-57.

Hoyles, C, (1992) Mathematics Teaching and mathematics Teachers: A Meta-Case Study. In *For the Learning of Mathematics*, 12, 3 pp. 32-44, November.

Johnson, D., and Johnson, R. (1975) In *Joining Together: Group Theory and Group Skills,* Prentice Hall, New Jersey.

Laborde C (1989) 'The Audacity and Reason of French Research in Mathematics Education'. In *For the Learning of Mathematics*, 9,3, pp. 31-36

Laborde C (1993) 'Solving Mathematical Problems in Small Groups: interaction between cognitive and social aspects'. Paper to Seminar 'Collaborative Learning: what can children learn together, specific skills or general concepts', Oxford. February.

Perret-Clermont, A.N. (1980) *Social Interaction and Cognitive Development in Children,* Academic Press, London.
Slavin, R. (1983) *Co-operative Learning,* Longman, New York.

An Educational Perspective of the Socio-Cultural Pre-requisites for Logo-like Education in Greece

Gitsa Kontogiannopoulou - Polydorides[1], Chronis Kynigos[2]

[1]Division of Sociology of Education and Educational Policy,
Department of Education,
University of Patra

[2]Section of Education,
Department of Philosophy, Education and Psychology,
School of Philosophy, University of Athens

Introduction

It is now well accepted that Papert's more than twenty year - old thesis was but a strongly argumented vision (Papert, 1972, 1975, 1980, 1981, 1987). The use of computers in education has not spontaneously brought about self - created problem solvers, happily weaving between the concrete and the abstract, enjoying learning as a way of life and having discarded the educational heritage of passive reception and rote reproduction of chunks of disembedded information organised, chosen and quantitatively evaluated by others (Leron, 1985, Noss and Hoyles, 1992). The problem of generating learning environments related to Papert's paradigm has in itself been a task where, although important progress has been achieved by education researchers and/or informed, motivated and flexible teachers in (to a greater or lesser extent) ivory - tower situations, it has been matched by an increasing understanding of the complexities involved in bringing about such change in learning and teaching (Hoyles and Noss, 1992, Hoyles and Noss, 1992b, Hillel and Kieran, 1987, Harel and Papert, 1990, Hoyles and Sutherland, 1989).

Nevertheless, contemporary conceptions of the role of computers in education as that of a qualitatively powerful tool for children to express and explore ideas and ways to represent them facilitated by a teacher, do seem to have their roots in this vision. The ways in which computers have actually been perceived and practiced in educational systems around the world (not excluding the use of Logo - based software) seem to have gone through an evolution from a) a technical and isolated subject matter in itself, to b) the use of content - specific software designed to minimise users' need to learn about the technology itself, to c) the "integrated" perception of learning about the expressive power of computers within the learning of subject - matter (Kontogiannopoulou-Polydorides, 1991). What is interesting is how this 20 year - old vision seems to be ingrained in the "integrated" paradigm (Noss and Hoyles, 1992, diSessa and Abelson, 1986, Give'on, 1991), which is still a target for most countries and a recent development for some (Thompson and Noel, 1991, Plomp and Pelgrum, 1992, Solloway, 1990).

The very nature and structure of many educational systems constitutes contradictory environments for facilitating or even accepting such attempts. The wide discussion and concern about the, so-called, "crisis" in education and the expediency of the issues involved makes politicians include it in presentations about their priority structures. In fact, one could argue, education is of low priority as the respective spending at national levels indicates, of poor cost effectiveness and means of evaluation, and slow (far beyond any term of government) fruition of investment. The rhetoric on the "crisis" in education is quite convincing that, around the world, educational systems are characterised by a traditionally conservative nature, illiteracy is surprisingly widespread and the way in which society influences what constitutes educational progress is in many respects distanced from what educators have to say about it. The new educational technology and

its effective use for the "salvation" of education from the "crisis" in the form of exemplary applications in specific school settings will not automatically bring widespread change (Apple, 1991, Budin, 1991, Noss, 1992). Spreading educational change is a very different problem to that of technology bringing information to every home or computerising a firm to improve cost effectiveness. Educational priorities for the shaping (Noss, 1992), the development (diSessa and Abelson, 1986, Solloway, 1989) and the use (Solloway, 1990) of this technology are built on a growing theoretical and empirical problematic whose origins lie on interrelated educational and technological foundations (Papert, 1980, Sinclair and Moon, 1991) and the effects are very difficult to measure, especially in the short run (Papert, 1987). As Weisnbaum stated a few months ago in a critique on the use of computer technology in the U.S. educational system, cost effectiveness does not apply to education (Weisenbaum, in press).

Influences on how computers are used in Greek schools

Following the above, the case of computer use in education in Greece is not exclusive or particularly surprising. However, the way in which computers are being introduced in education are dependent on specificities which relate to the structure of the country's educational system and practices, the centralised nature of national policy and decision making, processes concerning decision making in the pedagogy of specific subject matter and perceptions related to technology and its use in education (Kontogiannopoulou-Polydorides, forthcoming).

It is important to analyse these factors in order to understand the complexity of the problem of introducing Logo-like pedagogy and pedagogical practices in any form recognised (if not encouraged) officially by educational authorities in this country, since the wider educational context seems to nurture contradictory practices. To do this it is necessary to examine:

a) the fundamental features of educational policy and practices (referred to as "E.P., i.e. "EDUCATIONAL PARADIGM") and how they relate to the characteristics of policy and practices regarding the use of new technologies in education (referred to as "E.T.P." i.e. "EDUCATIONAL TECHNOLOGY PARADIGM") and

b) how they contrast with certain aspects of educational practice deriving from the above outlined integrated approach in the use of educational technology in the international scene.

We argue that Logo - like pedagogical practices in Greece may emerge only under alternative E.P. and E.T.P. circumstances, whereby, top - down processes would have to somehow facilitate or encourage developments emerging directly from school practice, and in the Greek setting this seems contradictory. In doing this, we need to be able to demonstrate the kind of problems emerging from culturally situated school - based experience of using computer technology. We're thus going to use an a-typical example of a Logo-like experience in a specific Greek school setting and build our arguments by identifying the contrasts of the emerging educational issues with the wider educational context.

The Greek educational paradigm

We begin with a short outline of the fundamental features of E.P. in Greece. These features, as they relate to policy making, include the way subject matter is formed and teacher employment policies.

The pedagogical practice prevailing in practically all Greek schools is characterised by an emphasis on content, abstract knowledge, teacher - centred approaches and a lack of a systematic pragmatic orientation. Firstly, the teacher's role is to re-produce the knowledge domain which is strictly defined and can be found in the (until recently) unique curriculum book. Inevitably, the teaching paradigm is characterised by frontal teaching and information - transmitting (be that explanatory or descriptive). Secondly, "learning" is perceived as the (usually rote) memorisation of content and the ability to re-produce it upon demand. Thirdly, curriculum content is perceived and formed as a simplified and reduced version of (inevitably) abstract University curricula (why this has come about is explained in the next section). Finally, learning is seen as an individual, rather than a cooperative or a group experience[1].

The prevailing education paradigm in Greece has the characteristics of what M. McLean has termed as "humanism" with a few elements of "encyclopaedism" (McLean, 1990). Classical Greece has traditionally dominated language teaching up until the 70's, when demotike language was established and prevailed ever since. Greek Orthodox Christianity has penetrated with a moral - relevatory approach and has influenced the epistemological view on which knowledge production and education have been based (McLean, 1990). This orientation has been traditionally coupled by too much factual rote learning and too much formal instruction with very limited opportunities for class discussion or creative work.

With respect to teacher employment policy, there is a pattern which has been witnessed in Greece over and over throughout the history of the evolution of the education system; the integration of discipline educated scientists and/or professionals in the education system as subject "teachers", and the perpetuation of a single emphasis on what has been traditionally considered useful knowledge and the way to appropriate that knowledge, which is memorising mainly the abstract content of the subject matter. In this way, educational policy in introducing computers in schools is in effect "creating" teachers by Law, where teachers do not exist, defining, therefore accordingly what the content character and structure of computer education in schools will be (Kontogiannopoulou-Polydorides, forthcoming). This is duly intensified by a number of factors:

a) the relatively limited opportunities of scientific expertise to be meaningfully employed ("scientific" unemployment) forcing them to fill the ranks of "applicant teachers"

b) the domination of traditional University disciplines over what happens to primary and secondary education (e.g. the domination of the School of Philosophy of the University of Athens, or the Mathematics Department of the University of Patra) over the structure of the educational system and curriculum structure and content, and

c) whereas other areas of knowledge require that University training takes into account the development of science, technology, the economy and/or society at large, "education" and "teaching" at the secondary level does not need to do it since the forces which shape the content of University studies control and determine at the same time what "education" and "teaching" in schools is.

How computer use began and developed in Greek schools

Let us now outline a brief account of the E.T.P. in Greek schools. From the beginning of the 80's there has been a general tendency in decision centres at the level of national strategy to accelerate the integration of information technology in the public and private sector. The two main reasons for this were the optimism created at the beginning of the decade due to a change in government and the general climate of urgency to narrow the gap between Greece and the northern E. C. countries. In order to achieve speed, top - down procedures were adopted which in any case matched the centralised decision -making structure of the public sector. From the outset, the decision centres regarding new technology integration strategies consisted mainly of engineers and computer specialists. In each respective case, when the target population's needs to use computer technology was studied, the analysis was carried out under the generalised methodology of systems analysts, with the theoretical and empirical backing of the effects of computerisation on company needs. Unfortunately, but not surprisingly, this process was transferred to education.

It is not surprising for a number of reasons. The decisions of national strategy committees are crucial for each application domain and these of course depend on the specific expertise of their members. Until now, such committees consisted mainly of Computer Science University Professors and Engineers (Kontogiannopoulou-Polydorides, et. al, 1992). As it has been presented in the previous section, the exercise of power in Greek education has resulted in educational decisions on subject matter being made by the respective subject matter specialists who define content as an academic and abstract body of knowledge taught in the University. This has been applied to the field of informatics as well. Subject matter is taught, by Law, by subject matter specialists with minimal, if any, teacher education, a practice reinforced by labour unionism and graduate unemployment. It is within this practice that Informatics could be taught by

Computer Science graduates with no knowledge whatsoever about education. Unemployment due to low absorption of scientifically trained personel is "creating" a potential teaching staff consisting of such informatics graduates and an artificial need to add the topic of informatics at school level instead of integrating the technology in the teaching of curriculum subjects[2]. Finally, the relatively recent upgrading of pre-school and primary education to University level still means that educators in the field are often left away from decision making.

Thus, at the level of national strategy, the origins of the conceptualisation and philosophy of introducing computers in education came a) as an extension of the attempt to computerise professional functions, b) as the intention to train students who would soon board the working community, to fit the new scheme, i.e. to use the technology for more effective production and c) as a means to employ unemployed University graduates.

In this, there was a passive consensus amongst political, European Community and technical decision centres, so that internationally promoted goals, at the time, were interpreted in the above specific scheme. The position of education authorities to accept the use of computers as long as the curriculum and the teaching method was unchanged, was aimed at "protecting" existing educational practices, meaning in fact that pedagogy was altogether left out of the picture. Money was one of the few limiting factors in equipping a large percentage of schools with computers and proceeding with the technical training of the pupils.

Within the educational system itself, the initial strategies of computer use targeted vocational and technical education, where the argumentation for the above processes found some common - sense ground. Recently, however, there is an attempt to transfer such policy to lower - level general education. The default tendency seems to be the preservation of existing educational practices and the attempt to computerise them in the expectation that the production of the future workforce will become more effective. It is evident that regarding educational goals and priorities at the level of general education, these are totally irrelevant, not only on the basis of what is the internationally accepted trend in the educational use of computers, but on the basis of what is considered a sound approach to general education in Greece.

Within this context, computer education and computer use, promoted as "informatics" is "taught" as a simplified reproduction of the classes initiated a few years earlier at the University. Informatics has thus been introduced as an independent subject following the goal of training for higher productivity in the workplace, as it has been interpreted by the Greek education authorities. Indeed the educational goals set forth by the Greek Governement, as they evolve from statements included in the Presidential Decree 108/1987 are to:

— Introduce a beginner's familiarisation of students with the computer

— Provide secondary school graduates with sufficient training, so that they can be integrated in the production process, which is constantly transformed throughout the use of new technologies

— Encourage specialisation of the future workforce in the technical use of the computers.

The introduction of "informatics" in schools is thus clearly part of the continuous process whereby education and teaching is reduced to nothing more that the simplification of lower - order content offered in the course of University studies. A recent Survey research (I.E.A.[3] carried out to establish the way computers are actually used in Greek schools in comparison to other countries around the world (Kontogiannopoulou-Polydorides et al, 1992, Pelgrum and Plomp, 1991) reveals a picture of isolated theoretical and technical teaching which does not interact with the educational activities of the school. For instance, the percentages of the context in which pupils receive most of their instruction in learning about computers reveal that this context is a separate course in100% (lower secondary) and 96% (higher secondary) of Greek schools (now embedded in the local curriculum), while the average in five E.C. countries is 63 and 65 respectively, and in Portugal (often compared to Greece because of its parallel developmental status within the E.C.) the percentages are 35 and 68 (!). Furthermore, the percentage of computer use in particular subjects for lower secondary shown in the table below reveals its isolated nature with respect to other E.C. countries.

Table: Percentage of Computer Use in Particular Subjets[4]

Topic	*Greece*	*Some E.C. Countries*	*Portugal*
Informatics	97	73	15
Mathematics	12	56	42
Science	0	27	19
Mother Tongue	0	28	14
Foreign Language	2	23	9
Creative Arts	0	8	9
Social Studies	0	18	21
Commercial Studies	2	25	19
Technology specific	0	18	10

Finally, with respect to the kinds of applications used, there appears to be a strong emphasis on programming languages followed by word processors and data bases, which are used in their programming potential rather than as tools for creative educational projects and/or activities.

It is clear that, within this context it is practically impossible to expect that computer education in schools will develop into something more than the content of introductory programming and application packages. Subject matter domination, meaning in fact traditional University discipline domination, will be one of the major forces which will shape computer education is secondary (general) schools, at least in the immediate future. Furthermore, while there is evidence that teachers in charge of computer education before the incoming computer specialists (mostly mathematics, but also physics teachers) were aware and, in some cases, used the computer in line with a pragmatic and even an integrated model, the presently adopted law reduces considerably their already minimal impact. Such a development will jeopardise efforts of innovation and modernisation which have been promoted in the last fifteen years, as well as a few, but existing, isolated attempts for the integrated use of the computer in the classroom, such as the one we elaborate below.

A Logo-like experience within a school setting

So far, we have only argued that computers in Greek schools are used in ways which are irrelevant to the problematics of general education, while international experience and research is showing that not only is this needless, but that specific computer applications have properties which allow for a qualitative upgrading of educational process and practice in a new and powerful way. What needs to be discussed is the kinds of issues which would emerge from such a use of computers within the Greek cultural context. What kind of issues would be relevant to the development of Greek educational practice? How can computer technology be used to surface these issues? To do this in a concrete way, we will situate our argument within an example of a Logo-like educational project in a Greek primary school.

The example we describe is an on-going seven-year long project in Psychico College - a Private Greek Primary school setting[5] - aiming to try out alternative pedagogy by means of the children's learning of programming with conventional Logo; all pupils and their own teachers use this technology to try out child - centred investigative group work in the classroom for one hour a week throughout the final four years of primary school (Kynigos, 1992). Two disclaimers are immediately appropriate, the first mainly due to lack of consistent financing throughout all these years and the second due to automatic restrictions posed on the application of alternative pedagogy to the conventional curriculum; the initial technology of Apple IIc's and conventional Apple Logo II have remained unchanged and the developing learning environment has not, at least explicitly, been integrated into the curriculum in a formal and systematic way. Nevertheless, the school project involved continuous but not intense teacher education, a Logo curriculum development and informal research into pupils' and teachers' strategies. More details of the project's outline, educational objectives, working structure, classroom setup and "taught" content can be found in the proceedings of the last two

Eurologo Conferences (Kynigos, 1989, 1991). Studies involving children's learning process can be found in Kynigos, (1991, 1992, 1992b, in press). A study into children's use of programming ideas is described in (Kynigos et al,1993). Finally, a two year research project based at the Department of Informatics has very recently been accepted by the General Secretariat for Research and Technology. The aims of the project are to study certain crucial aspects of a pedagogically - centred use of computer technology in a school setting after the emerging learning environment has had time to mature; namely, pupils' small group social dynamics in a collaborative setting; how they construct or reformulate meanings; which are those meanings and how they (may or may not) relate to specific curriculum content; the teachers' developing pedagogy; and certain emerging implications for the design of exploratory software.

In this paper, we outline certain features of the Psychico College project which are quite crucial to its development in the past seven years, yet either absent from the top - down directives of national strategy (at best), or contradictory to them. In our argumentation, we will draw from data collected from the above studies in order to provide insight into characteristic teaching and learning episodes. Episode analysis has been one of the main methods with which the data has been analysed within a framework of ethnographic progressive focusing methodology (Atkinson, 1979, Goetz and Lecompte, 1984, Cobb et al 1992, see also Kynigos et al, 1993).

The data used here is collected by means of a) transcriptions from recent video recordings made by non - participant observers of teachers' interventions in the learning process in the classroom, b) the children's extensive written accounts of their six teaching period - long investigations and their teachers' comments on those (note 1) and c) transcriptions of interviews with the teachers in order to look into their pedagogical strategies and agendas, their attitudes towards this way of using the computer in the classroom, towards mathematics and the extend to which they use methods developed in the Logo classroom in teaching other topics.

Two episodes from the Logo classroom will be analysed here. The first is a simple interpretation of all the comments a teacher in the third year of primary made during one teaching hour, complemented by data fom the interview we took of the same teacher. All the comments have been interpreted and numbered for reference in this paper (in brackets). The episode is mainly used in a discussion of the attempt to try out an alternative pedagogy and the long lasting relevance of the process - oriented pedagogical objectives. The second is highlighted in Figure 1 and shows the result of a project carried out by a group of children who are now in the final year of primary. We use it to discuss both children's inductive way of making generalisations and the issues emerging in the teachers' content - related interventions.

Trying out an alternative pedagogy

From the outset, the technology was used to set up a very different classroom organisation; as implied earlier, the pedadogy prescribed in Greek schools tends to involve the teacher addressing the whole class at once, transmitting disembedded information, coaching the solution of exercises and testing the reproduction of that knowledge with little means to determine wether it was achieved by rote or not. The curriculum is to be followed accurately and until very recently it came in the form of a (unique) book approved by the Pedagogical Institute. In the Psychico College project, the technology was used to qualitatively reform and develop learning and teaching processes, rather than to quantitatively optimise existing ones.

Pupils worked in groups of two or three, were encouraged to discuss ideas, negotiate how they would cooperate, persist on a problem and deal with it in depth, develop autonomy and responsibility towards themselves and their group peers, present and discuss the results of their projects (studies of related issues in school settings can be found in Hoyles et al 1992, Hoyles and Sutherland, 1989). Evaluation came in the form of informal encouragement to do better, rather than a final judgement on past and irreparable performance (e.g. see comment 11 in the analysis below). The computers were used as tools for the pupils to express and explore ideas in a process of alternating between qualitatively different methods of representation (Bruner 1974, Mason, 1980, e.g. see comment 15). Computer feedback was used by the teachers to inject pupils' perceptions with a different meaning to error; in the early years of the project, it

was frequent that children would hide the screen with their hands (or repeatedly press return) when the teacher came close, so that they wouldn't see their mistake! (see, e.g. how this contrasts with comments 9,10,13,15,20).

The pedagogical paradigm developing within the project can perhaps be better described by the story emerging from the following informal analysis of a teacher's interventions during one teaching period in a third year classroom. This particular hour was the third in a four - hour investigation, so learning situations had already had time to emerge. It is interesting to observe the balance in the teachers' comments between cooperation and management within a group, Logo mathematics content, classroom management, technical and factual assistance, process - oriented skills such as perseverance analytical explanations, awareness of feedback.

1) Teacher requests responsibility regarding the writing of commands in the writing book and not forgetting it by one group member passing the responsibility on to the next.
2) Teacher comments (disapproval) on the easy abandoning of effort to two groups.
3) On a group's request for permission to de-goal - in the sense of Noss, 1985 - (reason is that they forgot the book and can't remember quantities to make their figure) teacher insists on reconstructing the same goal and adjusting the size later.
4) Teacher makes a factual comment: type instead of switching the computer off.
5) Teacher suggests noise reduction five times.
6) Comments on writing book and abandoning of effort.
7) Comments on content: asks a group to conjecture on a quantity instead of homing in one by one turtle step.
8) Throws the control back to the pupils who asked for help in this.
9) Raises pupils' awareness of feedback and the commands they typed in. They thought the computer did not do what they instructed it to.
10) Encourages analytical thinking (e.g. difference between rt 90 and rt 89 as example of why line is dented on screen).
11) Well done, she says to a group showing patience and perseverance.
12) Intervenes on cooperation: stops one in a group who was typing things without negotiating with the others - asks for discussion amongst them.
13) On content: regarding the circle (clearly not first time issue arises - look back) nudges pupils to look for a pattern to repeated approximations of turns and fd's.
14) Intervenes in an argument amongst a group in favour of effective work (if you cant remember do it approximately and you'll check it in the writing book next time)
15) Encourages pupils to try an idea on paper - then to try it by playing turtle, on their request for a turning quantity (checks direction awareness).
16) Teacher asks for a clear explanation from the pupils and insists when its not clear enough.
17) Teacher spends time circulating to check who needs her
18) Teacher intervenes. "What are you looking for?" the pupils could not make out the quantity of a turn. The teacher identifies homing - in strategy, suggests they get more bold about it and conjecture how much it is from the start.
19) Teacher asks one in a group to join in.
20) Back to the circle group... asks where they're at and discovers they didn't do what she wanted, i.e. constant turns and moves, they change turns. She accepts and asks whether they can predict what shape will come out. They say it wont be very much like a circle, there will be straight bits. Then she says, you better think again about turning it. Showing on screen that with one group of commands it looks more like a circle than with another she concludes that they should rethink. So that it would come out more like a circle. She asks them to compare. The, asks them not to erase old commands so that they can notice afterwards - reflect - and then she says: see if in this group where the turns were the same if it was more like a circle.
21) A technical comment - switched to Greek. She says to the children that the computer does not do things by itself. Encourages them to try again.
22) Addresses the whole class: children we have finished - bell has rung. Be careful to write all the commands - don't miss anything out. One child says: the next week it will be... she says: next week it will be the last period in this investigation, ok? you'll have to complete and have time to print.

The use of a specific technology (note that it is not so new anymore - many children have much better machines at home) and the changing role of the teacher in order to bring about and encourage pupil - controlled problem - solving activity, by no means came automatically; the above episode, for instance, is recent and this means that the teacher has had almost seven years' experience of developing this pedagogy. This, of course, was not due to some specific conceptual difficulty teachers or children had with understanding how they could use the Logo environment to practice an alternative way of working. What took time to develop was the process of making the important issues socially explicit within the classroom (this agenda is clearly embedded in comments 2,3,8,11,14,19). For instance, the teachers gradually developed a meaningful way to communicate to the children why they were not readily giving them factual answers and why they would often throw the responsibility of a situation back to the children themselves (as in comment 8). It also took time for the children to absorb that this is socially acceptable and legitimate

(after all school experience says that the teacher is there either to provide answers or to test whether the pupil can provide them). What has changed over the years is the level at which this paradigm has integrated in the school's culture, i.e. the working method, classroom organisation and teacher - pupil relationship seem as though they are autonomously there - unlike in the early days, in the episode we don't see evidence of the children's discomfort with the way the teacher behaves. However, this does not mean that the process - oriented educational objectives are not relevant any more - quite the contrary.

The long term relevance of process - oriented objectives

In contrast to the E.T.P. perspective of using a drug and then "inventing" an illness for it (these are Weisenbaum's words in criticising similar tendencies of computer use in the U.S. in a recent I.F.I.P. conference), the use of a new technology in this case originally aimed at educational objectives already there and poorly met by current methods. This was initially based on evidence from the pilot year (Kynigos, 1989) when trying to setup a Logo-like classroom revealed the children's dependency on the teacher for learning, their reluctance to explore an idea voluntarily or persist beyond the trivial to solve a problem and their overwhelming inexperience in constructively cooperating to achieve a negotiated goal.

As it has turned out, these process-oriented objectives are still in effect. For instance, by the end of primary education, every pupil has had four years' experience of negotiating ideas, working method and the delegation of responsibility, within a group of three. There is a consensus amongst the teachers that this is not only valuable education for the children, but that they don't get enough of it. In some cases, teachers have developed ways for the children to work in groups within their normal curriculum (as stated by the above teacher during the interview). So the teachers' developing role in the Logo classroom towards that of child - centred learning facilitator, has been a crucial feature of the project.

The working structure of carrying out "investigations" and then presenting them on paper was not immediately employed to encourage investigational work. Indeed, the main reformulation of the program after the first year, was for the teachers to exemplify the rules of the game to the children and to raise their awareness of the structure within which they were working by means of taking more time to reflect, even within the process of one investigation (note how the teacher encourages this in comment 22). For example, in the third and fourth year, teachers now actually state during each Logo session, that this is, say, the second period of the fourth investigation and are particularly firm on receiving a "complete" presentation of each investigation, i.e. consisting of all three main parts, essay, commands and graphics. This is why, for instance, in the third year almost double time is provided for one investigation; investigatonal work still lasts for four teaching periods (this is one of the few un-negotiable rules of the game), but time is given for reflection in between and after completion. After the first two years of the program, the role of the working structure as an end in itself began to recede, gradually giving place to that of being used more as a means to encourage the children's learning of content and the enrichment of their use of powerful ideas in an investigational and personally meaningful spirit (note how in comments 1,2,3,6 the teacher uses the fact that the writing book has been forgotten to teach responsibility and persistence with a problem and not to impose an arbitrary rule on a structure for working).

Focusing on the learning of content

In a sense, focusing on the learning of content was and still is, a complicated affair. To begin with, the context within which the program took place did not favour intensive a-priori training of the teachers in Logo programming and the mathematical and programming content embedded in the language; very little time for teacher training was made available (a far from rare situation in any school) before and during the program; all the teachers were taking part, with varying attitudes to change, to computers and technology in general, and naturally to taking-on additional work; the teachers had little means of being socially reassured, either by tangible examples of similar work, or by official educational authorities; within the Greek educational system, there is anyway very little provision for the teacher to take initiatives in deciding on teaching method or content. It was therefore far from reasonable to expect the teachers to change their teaching method and role in the classroom and to simultaneously teach substantial content within the new state of affairs.

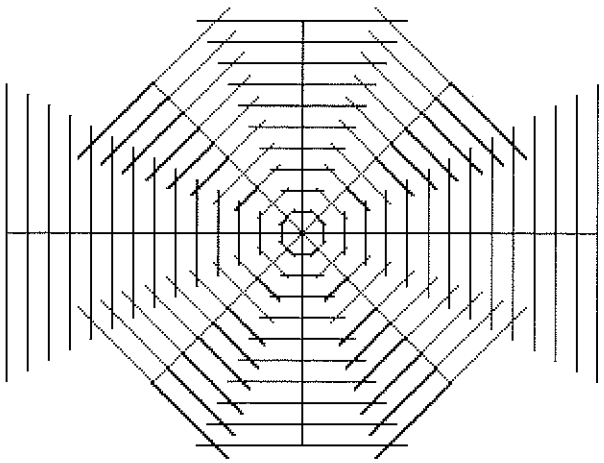
The second difficulty with focusing on content is related to Logo's strength as an educational tool. Situations where the children use ideas such as turning relations first, and then discriminate and generalise them, are common amongst the children's work (Hoyles and Noss, 1987). However, the problem lies in trying to influence the learning environment, on the one hand in order to discourage an unreflective use of Logo (Leron, 1985), as the teacher is clearly trying to do in comments 7,9,10,16,18, and on the other, in order to help the children to focus on and become aware of the interesting and powerful ideas which they use in amongst their projects (see for example comments 13, 20). The need to intervene in the children's work regarding both of the above issues is as real in this school as in any environment where children do Logo (Hoyles and Noss, 1992b). The teachers have two means to make interventions in the children's learning within the program's working structure; interventions in the classroom during the learning process; using their presentations to provide feedback. This teacher's comments reflect a general trend, whereby the teachers have begun to detect opportunities for such content related interventions, and are now in the never-ending process of enhancing their recognition of ideas and situations such as the ones described above and of refining and enriching the ways in which they intervene (for a related discussion see Hoyles and Noss, 1992b). Within this context of the teacher's struggle between enforcing a didactic agenda and allowing the pupils to persevere with an alternative but interesting set of ideas on which they stumbled, the following episode highlights the ad - hoc and localised way in which they make generalisations by repeatedly building superprocedures.

Pupils' inductive generalisations

The following example (Fig. 1) firstly shows how children may use specific programming ideas to express notional objects of an increasingly generalised nature. Secondly, it shows how these generalisations are derived inductively and are characterised by their direct applicability to the local problem at hand. In an overview of research into children's mathematical learning with the use of this programming environment, the term "situated abstractions" has been used to emphasise both that children may be capable of abstract thought more than they were given credit for and that their abstractions are bound and closely related to the learning situations they find themselves [Noss and Hoyles, 1992].

```
TO TAT :A
FD :A
LT 90
FD :A / 2
BK :A
FD :A / 2
LT 90
FD :A
RT 180
END

TO KARDIA
TAT 10
TAT 20
TAT 30
TAT 40
TAT 50
TAT 60
TAT 70
TAT 80
TAT 90
TAT 100
END

TO BBB
RT 90
KARDIA
TAT 110
TAT 120
TAT 130
TAT 140
END

TO PAPAFLESSAS
RT 180
KARDIA
TAT 110
TAT 120
TAT 130
TAT 140
END

TO FATANGEL
LT 90
KARDIA
END

TO BTFSCCG
RT 135
KARDIA
END

TO KKK
RT 90 KARDIA
END

TO LLL
RT 90
KARDIA
END

TO MMM
LLL
END
```

children's presenteation :



the schema results by accumulation of :
KARDIA BBB PAPAFLESSAS FATANGEL
BTFSCCG KKK LLL MMM

Figure 1: Generalizing in an inductive way

The procedure written by the children (aged 12 with three and a half years' experience) for the letter T (TAT

:A) was the result of a specific attempt by their teacher to provide them with a scaling microworld by means of showing them how to half the variable input in order to form the top part of the letter. What is in focus here, however, is their ad-hoc subsequent building of superprocedures. Firstly, procedure KARDIA , i.e. ten executions of TAT with an increment of 10 for the input, is used as a notional object (subprocedure, independent of the turtle's current state) for the writing of all subsequent procedures.

However, the children did not in the same way perceive as an independent object a repetitive (14 times) execution of TAT which at least in their figure was of equivalent "status" to the output of KARDIA. Instead, they thought of the 14 times iteration as a synthesis of KARDIA plus four more direct (i.e. not within a subprocedure) executions of TAT . This is evident in both procedures BBB and PAPAFLESSAS, which are exactly the same apart from the interfacing commands (the second procedure is the same as RT 90 BBB, a pattern which they used in procedures containing KARDIA). In fact, their use of interfaces from one shape to the other within the procedures which form the shapes themselves may be one factor leading them to write different procedures for the same figure, since this strategy is apparent in all their procedures.

However, the contents of procedures KKK, LLL, and MMM indicate that they were not at ease with perceiving the abstraction property of procedure writing. The first two have identical contents, the only difference between them being the situation in which they were used and the third one is a trivialisation of superprocedure building, stressing even further their resistance to use a procedure independently of the situation for which it was firstly written. The antithesis of their use of these last procedures to that of procedure KARDIA which is perceived as an object abstracted from specific situations is characteristic of the notion of "theorems in action" (Vergnaud, 1982).

Finally, it is worth noting that the teacher's didactic agenda for the children to employ the notion of scaling in order to write procedures was not used at all by the children after writing the initial T procedure; their self - imposed task of constructing this figure did not involve situations where scaling may have been meaningful as far as they were concerned. The children therefore did engage in mathematical thinking, but mostly on a content which was subsidiary to the teacher's agenda. The point is, that the teacher seems to have allowed and encouraged this, respecting the specificities of the learning situation and perhaps postponing his/her agenda for scaling problems for the next project.

The emerging educational paradigm

It is evident from the above that the educational paradigm emerging from the Psychico College experience is in contrast to both the E.P and the E.T.P. as outlined earlier. Instead of the frontal setup, the teacher assumes the role of a collaborative facilitator in the pupils' construction of knowledge from their own experiences and provides child - centred support, sensitive to the mixed abilities within the classroom. There is minimal "knowledge domain" to be transmitted in the form of information, instead the children are encouraged to form theorems and refine them as they engage in self - initiated and chosen projects. The content is formed by combining actual educational needs arising from school practice and learning how to use computer technology as an expressive medium. They are further encouraged to acquire autonomy from the teacher and to work collaboratively in small groups.

Another kind of emerging characteristics of this educational venture is that it was not quantitatively evaluated, there was no hurry and no intensity, especially from the teachers' point of view. All the teachers and children were engaged (and not just the "clever boys" from both populations) and perceived it as a self - development exercise rather than a means to pass a test. The educational objectives were evolving and continually being refined instead of serving as directives. Finally, the teachers took the role of research collaborators, graciously discussing their experience with the researchers from "outside".

Conclusion

The first question which is raised from this dual presentation of the fragmented nature of the wider E.P. / E.T.P. and the specific school - based pedagogical paradigm integrating the two is that of relevance; how relevant is the experience of the Psychico College primary school for the Greek E.T.P.? No doubt this specific

primary school operates, in principle, within the wider Greek E.P. The use of computer technology, however, has not only operated "outside" the Greek E.T.P., but also brought an emerging alternative E.P., at least within the computer hour. What is more, certain features of this new technology which enable developments in educational practice surfaced and were put to use accordingly.

In order to think about how computer use of this nature could develop in the wider E.P. context, the circumstances under which this happened in this case must be specified. The teachers were receptive to new ideas, encouraged to try them out for themselves, being provided only with a general framework within which to develop their own pedagogy; they were systematically supported concurrently with practising in the classroom; seminars were related to classroom practice, they were short-term and wide-between and there was no intensity or formal quantitative evaluation of performance. The crucial issue for discussion was on which features of this tool to use for developing educational practice and how this could be done. Also, the teaching of content has emerged under this new E.P. and is being continually refined. But more importantly, the Psychico College project has been carried out first at a time when no National curriculum regarding educational technology or computers existed, and second, within an educational culture which prides itself of being innovative, pedagogy oriented and managing to generate within the system but on pedagogical principles against those of the system.

The questions relevant to educational policy making in order to draw some conclusions for the potential use of the school's experience to restructure the Greek E.T.P. are the following:

— what is the specific educational and decision - making process which has prevailed in Psychico College?
— how different are the wider educational practices of the teachers in this specific school from other schools in the country?
— how well will the pedagogical paradigm in the use of Computers in this school be sustained after a directive national curriculum on the use of computers is enforced?
— under what circumstances can teacher education support and be concretely related to teaching practice?
— what is a workable balance for the teachers be between learning about features of computer technology and learning about exploiting those features to develop pedagogy?

It seems that to enable the development of an integrated approach in using computers in general education, prerequisites which make sense are more to do with developments in the Greek Educational Paradigm and practices than an immediate derivative of E.T.P. and thus, by definition, more complex and difficult to apply. An essential feature, for instance, is the decentralisation of decision making concerning educational practice; how else can teachers be encouraged to develop their pedagogy and relate their teaching to pupils' needs? What does it mean (for teacher education) to be able to adapt a systematic and pragmatic pedagogy? Which brings us to a second prerequisite, which is for teacher education in computer use to be closely related to the integrated approach. This would imply a major reorientation of E.T.P. where the priority now would be on which features of computer use can be employed to develop E.P. As a subsidiary to this issue (note how the priority changes again), the pedagogical capabilities of informatics teachers should be an integral part of their eligibility to enter the classroom (as indeed applies to any subject), if informatics is going to be added on to the general education curriculum (as seems to be the case for secondary education).

Notes:

[1] An unfortunate (for the pupils) paradox, is that recent developments in the contents of curricula books have been "imported" from contexts discouraging rote learning, so that the pupils now need to memorise information which is not designed for this purpose.

[2] See "classified" letter to the Ministry of Education in which the Head of the corresponding Department in one of the major Universities of the country analyses to the Minister the reasons for which informatics graduates versus existing teachers ought to be given priority in the employment policies of the Ministry.

[3] International Association for the Evaluation of Educational Achievement

[4] Source: Pelgrum, W. J. and Plomp T.J. (1991) The Use of Computers in Education Worldwide, Oxford: Pergamon

[5] The school project referred to, is directed by Dr. C. Kynigos and is taking place at the Psychico College, Athens. An important part of the application of the project was funded by the M. and C. Maliotis Foundation. We would like to thank the school's children, teachers and Director for their cooperation in the research. The research project currently in progress at the school is funded by the General Secretariat for Research and Technology and based at the Dept. of Informatics, University of Athens. Project leader is Prof. P. Georgiadis and research director, Dr. C. Kynigos.

REFERENCES

Apple M W (1991) The New Technology: Is It Part Of the Solution Or Part of The Problem In Education? Computers in the Schools, vol 8 (1/2/3), pp. 59-81, Haworth Press.

Atkinson, A.: (1979) Research Design in Ethnography Part 5, in Social Sciences: A Third Level Course Research Methods in Education and the Social Sciences, Block 3B, Research Design, The Open University Press, U.K..

Bruner, J.: (1974), Beyond the Information Given, Allen and Unwin, London.

Budin H R (1991) Technology and the Teacher's Role. Computers in the Schools, vol. 8 (1/2/3), pp 15-26, Haworth Press.

Christodoulakis D, Hatzilakos Th. Kynigos C. and Koutlis M (1992): Astrolavos: A Program to Utilize Computing Technologies to Support Education in the Hellenic Secondary Schools, Proposal prepared for the Ministry of Education under the supervision of Prof. D. Maritsas, Director of the Computer Technology Institute.

Cobb P, Yackel E and Wood T (1992) Interaction and Learning in Mathematics Classroom Situations, Educational Studies inMathematics, 23, pp. 99-122.

A. DiSessa, (1989) A Child's Science of Motion: Overview and First Results. Proceedings of the Fourth International Conference for Logo and Mathematics Education (Leron, U and Krumholtz, N. Eds), The Israeli Logo Centre, Department of Education in Science and Technology, Technion - Israel Institute of Technology. pp. 211 - 231 (1989).

A. DiSessa and H. Abelson, (1986) BOXER: A Reconstructible Computational Medium. Communications of the ACM vol 29 no 9, pp. 859 - 868.

Give'on Y.S. (1991) The Role and Positionning of Logo in Educational Informatics, Proceedings of the Third Eurologo Conference, pp 331-342, Parma, E. Calabrese (Ed).

J. P. Goetz , and M. D. Lecompte, Ethnography and Qualitative Design in Educational Research, Academic Press, London (1984).

Harel I and Papert S (1990) Software Design as a Learning Environment, Interactive Learning Environments, 1, 1 - 32.

Hillel J and Kieran C (1987) Schemas Used by 12 year-olds in Solving Selected Turtle Geometry Tasks. Recherches en Didactique des Mathematiques. p. vol. 8, no 12, pp. 61-103.

Hoyles and Noss, (1987) Children Working in a Structured Logo Environment: From Doing to Understanding. Recherches en Didactiques de Mathematiques, vol 8, no 12, pp 131-174.

Hoyles C and Noss R (Eds) (1992) Learning Mathematics and Logo, Cambridge: MIT press.

Hoyles C and Noss R (1992b) A Pedagogy for Mathematical Microworlds, Educational Studies in Mathematics, 23, pp 31-57

Hoyles C and Sutherland R (1989) Logo Mathematics in the Classroom. Routledge.

Hoyles C, Healy L and Pozzi S (1992) Interdependence and Autonomy: Aspects of Groupwork With Computers.

Kynigos, C.: 1992, 'The Turtle Metaphor as a Tool for Children Doing Geometry', in Learning Logo and Mathematics, C. Hoyles and R. Noss Eds, Cambridge MA: M.I.T. press, 97-126.

Kynigos, C.: 1992b, 'Insights into Pupils' and Teachers' Activities in Pupil - Controlled Problem - Solving Situations: A Longitudinally Developing Use for Programming by All in a Primary School', in Information Technology and Mathematics Problem Solving: Research in Contexts of Practice, Springer Verlag, NATO ASI Series, 219-238, Berlin.

Kynigos C (1989) Integrating Technology into the Culture of a Primary School. Proceedings of Eurologo '89 (Schuyten G and Valcke M, Eds), pp. 457-468. Gent: EDIF.

Kynigos C (1991) Spending Time on Process Before Content: Can Logo Work in a Primary School in Greece? Proceedings of the third European Logo Conference, pp. 139-164, editor: Calabrese E., University of Parma.

Kynigos C, Gyftodimos G. and Georgiadis P. (1993) Empowering a Society of Future Users of Infromation Technology: A Longitudinal Study of Application in Early Education, European Journal of Information Systems 2, (2), pp. 139-148.

Kynigos C (in press) Children's Inductive Thinking During Intrinsic and Euclidean Geometrical Activities in Computer Environments, Educational Studies in Mathematics.

Kontogiannopoulou - Polydorides G (1991) The Educational and Social Dimensions of the Use of the New Technologies in the School, paper written in Greek in Contemporary Issues vol. 46-47, Dec. 1991, pp. 77-93.

Kontogiannopoulou - Polydorides G et al (1992) Computers in Education, Research Report, Greek National Centr of I.E.A., University of Patra

Kontogiannopoulou - Polydorides G (forthcoming) Computer Education Policies and the Cultural Context.

Leron, U.: 1985, 'Logo Today: Vision and Reality', The Computing Teacher, 12, 26-32.

Mason, J.: 1980, 'When is a Symbol Symbolic?', For the Learning of Mathematics, 2, 8-12.

McLean M. (1990) Britain and a Single Market Europe: Prospects for a Common School Curriculum, The Bedford Way Series, Institute of Education Univesrity of London, Kogan Page Ch. 5.

R. Noss, and C. Hoyles, Looking Back and Looking Forward. In Learning Logo and Mathematics. (C. Hoyles and R. Noss Eds), Cambridge MA: M.I.T. press (1992).

Noss R. (1992) The Social Shaping of Computing in Mathematics Education. In: Pimm D. & Love E. (eds) The Teaching and Learning of School Mathematics. Hodder & Stoughton.

Papert S (1980) Mindstorms. Children, Computers and Powerful Ideas. Harvester Press.

Papert S (1981) Computers and Computer Cultures. Creative Computing, 7, pp. 82-92.

Papert S (1972) Teaching Children to be Mathematicians Versus Teaching About Mathematics, Journal of Mathematics in Science and Technology, 31, pp. 249-262.

Papert S (1975) Teaching Children Thinking. Journal of Structural Learning, 4, pp. 219-229

Papert S (1987) Computer Criticism versus Technocentric Thinking, Educational Researcher, 16 (1), pp. 22-30.

Pelgrum W.J. and Plomp T.J. (1991) The Use of Computers in Education Worldwide, Oxford: Pergamon.

Plomp T.J. and Pelgrum W. J. (1992) Introduction of Computers in Education: state of the art in eight Countries, Computers and Education, Vol. 17, No 3, pp. 249-258.

K. Sinclair, and D. Moon , The Philosophy of LISP. Communications of the ACM, vol. 34, n. 9, pp. 41-47 (1991).

E. Solloway, (1990) Quick, Where do the Computers Go?, Communications of the ACM, vol 34, n. 2, pp. 29-33.

E. Solloway, (1989) Highly Interactive Computer Environments. GOLEM Newsletter of Technology and Education, vol 1 n.12, pp. 9-10.

Thompson and Noel (1991) Computers, Curriculum and the Learning Environment: Computers and Education, Vol 16, No 1, pp. 1-5.

G. Vergnaud, (1982) 'Cognitive and Developmental Psychology and Research in Mathematics Education: Some Theoretical and Methodological Issues', For the Learning of Mathematics, 3, (2), 31-41.

Weisenbaum J. (in press) Plenary Lecture at the IFIP Conference on Informatics and Changes in Learning, Gmunden, Austria.

Learning Processes in an Exploratory Data Handling Environment

Vasilios Makrakis, Evagelos Papavassiliou
Department of Education
University of Crete

1. Introduction

Learning processes can be characterized in a number of ways. In general, learning with data handling is characterized as learning in an exploratory environment in which the learner is actively involved in learning processes, such as:

- problem solving;
- reflective thinking;
- discovery learning.

An understanding of these processes in learning with computer data handling is fundamental for a better comprehension of the pedagogical use of this type of computer use.

Problem Solving

One of the fundamental goals of education is to teach students to become successful problem solvers. Problem solving may be defined as "the combination of existing ideas to form a new combination of ideas" (Sutcliffe, 1988, p. 34), as well as a directed cognitive learning process that makes use of previously learned knowledge and cognitive strategies (Duffield, 1991; Duffield, 1990).

Much of a computer's potential for enhancing problem solving lies in its ability to deal with data. Computers can perform calculations that would be tedious, time-consuming, and may free students to concentrate on the problem solving task. A computer is also capable of generating data. This allows students to solve more problems, enables extraneous or confounding variables to be controlled, and allows students to operate under controlled circumnstances (River and Vockel, 1987).

Reflective Thinking

The skills required in an information society are those that are related to acquiring, analyzing, synthesizing, and evaluating data. The development of a sound base of reflective thinking skills is a prerequisite to successfully function in a society where such skills are highly valuable.

Discovery Learning

Discovery learning is actually a very ambiguous term which refers to the ways of classifying, relating and synthezing known information, besides the potential discovery of new knowledge.

According to Ausubel, Novak and Hanesian (1978) the processes of discovery include:

- hypothesis formulation;
- hypothesis testing;
- determining the strategy of application;
- identifying fruitful approaches;
- using systematic and economic methods of inquiry.

It is likely that the discovery process is a mixture of inductive and deductive inquiry. Discovery learning is potentially one of the best approaches for helping learners gain metacognitive skills, especially through the opportunities provided to learners by data handling technologies. This approach has tremendous potential provided that it satisfies the need for a combination of structured and experiential learning.

2. An instructional design framework for data handling

We are presenting a framework that purports to serve as a basis to instructional design theory in the field of computer data handling process. In advancing this cognitive-based framework, we have taken into consideration four major guidelines.

First, the framework would have to address both the acquisition and the construction of knowledge.
Second, the framework would have to deal with the process rather than the outcome of learning.
Third, the framework would have to deal with the building of higher order level thinking skills.

Fourth, the framework would have to include a holistic perspective to learning (taking into account both the cognitive, affective and psychomotor domain). The basic components or phases of the proposed data handling learning process include the following:

- data handling executive environment,
- learner's knowledge base (or long-term memory), and
- the enquiry-oriented components of planning, implementing, and communicating.

Each of these components are further subdivided into a set of related processes.

Data handling executive environment includes the program's characteristics, and its function with other existing media. In other words, this includes the various ways in which external information can be stored, processed and presented. It also incorporates, the data handing assignment, including the learning subject, the audience, the rationales, and the guiding learning goals. The latter may be summarized in the following:

(1) identify and generate enquiry questions, find, select, code and use information derived from a wide range of primary and secondary sources;
(2) identify systematic patterns or relationships in the data, and how to display data in a graphical form;
(3) analyze, interpret, synthesize and compare new information with existing knowledge; and
(4) make learners aware of what they have learnt and to reflect on how they went about a task and how they might plan their next learning task.

The second component refers to the learner's knowledge-base or long-term memory which may be viewed as the repository for all previously acquired information. This information is stored, in terms of linguistically based concepts linked together in a highly developed network (Sutcliffe, 1988, p. 28). Prior knowledge plays an important role in an enquiry-based computer data handling process. In contrast to the limited capacity of working memory, long-term memory focuses on the nature of knowledge representation in terms of content, strategies, and skills (Tennyson, 1992).

Content knowledge is categorized in three types (Tennyson and Rasch, 1989):

(a) declarative (knowing what);
(b) procedural (knowing how); and
(c) contextual (knowing why, when, and where).

As discussed earlier, learning with a computer data handling package gives emphasis on higher order skills, such as collecting, decsribing, analyzing, interpreting and synthezing data. The method which can be applied to is that of discovery or enquiry-oriented learning, a mixture of inductive and deductive reasoning. Inductive reasoning is similar to classification; by observation of facts we conclude a new fact which desrcibes the

initial assertions. Deductive reasoning starts with assertions and discovers new facts by logically examining the relationships or properties which the assertions describe (Sutcliffe, 1988, p.36-37).

The learner's knowledge base directly affects the three phases in the process of learning with data handling identified as: planning, implementing, and communicating (third component). Planning contains processes like orientation, hypothesis generation, and operationalization. The orientation is intended as a process for identifying testable questions. These questions are structured in the form of hypotheses, that is, a descriptive generalization or assumption about variable relations, which reveal anticipation about certain consequences. This process has a heuristic character and is strongly related to the operationalyzing process. This process specifies what is to be changed (independent variable), what is to be kept constant (controlled variable), and what to be measured (dependent variable) under the effect of the variable which is changed (independent variable). In other words it turns hypotheses into measurable assumptions.

Being aware of hypothetical but measurable assumptions children are aware of the need to make measurements and testing ideas or hypotheses by constructing simple instruments for data collection. The data stored in the data handling program could be collected either from standard sources or it could be gathered by the pupils themselves from their field studies.

Collection and encoding of local data from field studies, may considerably enhance practical work, experiential learning, and learners' active participation in the learning process. The correctness of the hypotheses are verified or rejected by the analysis and the interpretation of the data: a process that leads towards the formulation of more precise predictive hypotheses. Indeed, through analysis, interpretation, prediction and synthesis children can form generalizations and move towards increasing levels of abstraction. These processes may help children develop an awareness of pattern and sequence and become more aware of the relationship between structure and function. Through these cognitive processes pupils are learning much more than doing simple calculations. They are making connections, something which is lacking in the present day school work (Niess, 1992).

Communication of data may take place via verbal, textual and/or iconic presentation. As the very act of creating a presentation can in itself reinforce knowledge and understanding (Ross, 1991). The comparative advantage of computer data handling lies not only on the manipulation of large sets of data, but also on the representation of information in graphical forms. It has been documented that the iconic representation of model structures encourages discussion about qualitative conceptual structures (Niedderer et al., 1991), and may generate a much clearer and deeper personal understanding of the identified patterns and trends. The graph becomes a 'working medium' rather than an end product or object of study (Barton and Rogers, 1991), for exploratory thinking in which analyzing, interpreting, predicting, synthesising and communicating become the important activities.

3. Monitoring

A genereal conclusion drawn from research studies in the 1980s shows that students, especially low-achievers, "lack the knowledge and/or motivation to make the right choices in a learner-controlled setting" (Hollis, 1991, p. 51). However, as learners become more aware of how they learn and the cognitive processes involved in problem solving, they are more capable of gaining control of their learning process, which in turn enhances their achievement (Haynes and Armour-Thomas, 1988). Although, there is lack of evidence concerning the use of executive control skills for switching between the various subprocesses of data handling, we may hypothesize that only from the sixth grade pupils can cope with abstract planning processes. As it is argued by MESU (1988), only by the end of the primary school children are able to select relevant information from observation and can form generalizations, and moving towards increasing levels of abstraction. However, monitoring of cognitive processes involved in data handling is a critical component of the data handling instructional system developed here. In this constructivistic approach to learning, the teacher's role should be to enhance learners' ability to describe what they know after being engaged in a data handling activity (declarative knowledge), how they know it (procedural or operational knowledge), and justify the reasons in engaging in these activities (contextual knowledge). These three metacognitive processes are necessary components for learning to learn.

The most critical component in monitoring concerns teacher's effective intervention geared towards the promotion of learner's involvement in ganerating and organizing their own ideas. It also concerns, teacher's intervention at helping children to manage their anxiety and lack of confidence and enhance their critical thinking.

In a process-oriented approach, teachers encourage students to discuss with others the reasons and the ways by which they made certain solutions, and the processes by which they arrived at their solutions (metacognitive skills). They also have to employ data handling strategies that are appropriate to the individual child, and guiding the data handling activities in ways that keep the learner in control. Cooperative learning seems to be a teaching strategy that promotes the positive interaction of learners in small groups. Eraut and Hoyles (1989, p. 14) argue that "neither individual work nor whole-class teaching gives much scope for pupils to talk about what they are doing. Only in groups is it possible to envisage pupils talking about their work to any significant extent". Cooperative learning while doing group problem solving can improve students' team work, mutual respect and better patterns of communication (Mittlefehldt, 1991). Research has found that students using learner-controlled software were more motivated than students using program-controlled software (Kinzie & Sullivan, 1989). Added to that, students using either problem-solving or simulation software scored significantly higher on measures of knowledge and domain-general problem-solving strategies than students using drill and practice and tutorial software (Norton & Resta, 1986). Successful implementation of cooperative learning in a certain subject depends, to a great extent, on the climate of the classroom, which should be considered before longer cooperative activities are used (Lyman and Foyle, 1991).

Recent classroom-based studies of well-planned cooperative learning have indicated short and long-term benefits for both academic achievement and social goals (Slavin, 1986; 1983). This environment of learning, besides being socially interactive, it should also be reflective. This implies that students should be encouraged to reflect upon and monitor their own reasoning.

4. Concluding remarks

It is worth pointing out that the design and use of an effective data handling program obviously depends on the learning result being aimed at. In view of this, it is important to have a plausible classification of learning goals for data handling software.

Most of the ideas behind the software to be developed reflects a view of teaching and learning which aims to increase the factual and theoretical knowledge of pupils, develop their critical thinking skills, and provide practice in the designing of discovery-oriented learning activities.

Unfortunately, solid empirical findings are lacking today, but some promising results have already been revealed. Most of the studies published so far, evaluating data handling instruction report mainly anecdotal descriptions and case studies without control comparisons. It is our belief that many researchers have failed to take into account of:

(1) the type of human-computer interaction;
(2) the type of teacher intervention;
(3) the sort of learning tasks provided;
(4) the appropriate cognitive strategies; and
(5) the social context of learning.

The roles that data handling technologies are to play, will largely depend on both technical, pedagogical and social factors. There are, in fact, two main issues that must be considered if we are to maximize the potential benefits of data handling technologies. The first issue refers to the data haldling system-design, and the second to the contextual environment in which this system will be placed and utilized.

With respect to the first issue, human-computer interaction and associated human psychological and

learning factors are of primary concern in building a data handling computing system. The user interface must be interactive, user-friendly, and didactic in terms of learning how to learn rather than of what to learn. It should also be responsive to a user's data handling needs and entry level. An interactive computer data handling program should enable pupils to apply some of their own theories to the learning process and to take initiative in the discovery learning process.

The second related issue is the importance of attitudes and values towards learning with computer data handling. In particular, there is the purpose and motivation of the learner and teacher that will fuel the enquiry-oriented process. This implies that, it is one thing to know how to develop data handling computer systems, and it is quite another thing to be able to successfully accomodate this system in the school context. If we want to see data handling computing as an empowering tool, then society and the school should promote the idea of data handling not only as a descriptive and informatory, but as analytic, and reflective medium.

REFERENCES

Ausubel, D.P., Novak J.D., & Hanesian, H. (1978). Educational Psycjology: A Cognitive View. New York: holt, Rinehart & Winston.

Barton, R. & Rogers, L. (1991).The Computer as an Aid to Practical Science-Studying Motion with a Computer. Journal of Computer Assisted Learning. Vol. 7, No. 2, pp. 104-113.

Duffield, J. A. (1991). Designing Computer Software for Problem Solving Instruction. Educational Technology Research & Development, Vol. 39, No. 1, pp. 50-62.

Duffield, J. A. 1990. Problem Solving Software: What Does it Teach? (Doctoral Dissertation, The Florida State University, 1989, Disertation Abstracts International, 50, 3925A).

Eraut, M. and Hoyles, C. (1989). Groupwork with Computers. Journal of Computer Assisted Learning, Vol. 5, No. 1, pp. 12-24.

Haynes, N.M and Armour-Thomas, E. (1988). Assessment of Metacognition in Problem Solving. Journal of Instructional Psychology, Vol. 15, No. 3, pp. 87-93.

Hollis, W.H. (1991). Humanistic Learning Theory and Instructional Technology: Is Reconcilation Possible? Educational Technology, Vol. 31, No. 11, pp. 49-53.

Kinzie, M. B. & Sullivan, H. J. (1989). Continuing Motivation, Learner Control, and CAI. Educational Technology Research and Development. Vol. 37, No. 2, pp. 5-14.

Lyman, L. & Foyle, H. (1991). Teaching Geography Using Cooperative Learning. Journal of Geography. Vol. 90, No. 5, pp. 223-226.

MESU (1988). Data Handling in Primary Schools. London: NCET

Mittlefehldt, B. (1991). Social Studies Problem Solving with Databases. The Computing Teacher. Vol. 18, No. February, pp.54-56.

Niedderer, H. et al., (1991). The Role of Computer-aided Modelling in Kearning Physics. Juornal of Computer Assisted Instruction, Vol. 7, No. 2, pp. 84-95.

Niess, M. L. (1992). Winds of change, The Computing Teacher, Vol. 19, No. 6, pp. 32-35.

Norton, P, & Resta, V. (1986). Investigating the Impact of Computer Instruction on Elementary Students' Reading Achievement. Educational Technology, Vol. 26, No. 3, pp. 35-41.

River, R. W. & Vockell, E. (1987). Computer Simulations to Stimulate Scientific Problem Solving. Journal of Research in Science Teaching, Vol. 24, pp. 403-415

Ross, S.M. (1991). Interactive Videodiscs for Science Education. Journal of Computer Assisted Learning, Vol. 7, No. 2, pp. 96-103.

Slavin, R. (1983). Cooperative Learning. New York: Longman.

Slavin, R. (1986). Cooperative Learning: Engineering Social Psychology in the Classroom. In: Feldman (Ed.), The social Psychology of Education: Current Research and Theory. Cambridge: Cambridge University Press.

Sutcliffe, A. (1988). Human-Computer Interface Design. London: Macmillan Education.

Tennyson, R.D. (1992). An Educational Theory for Instructional Design. Educational Technology, Vol. 32, No. 1, pp. 36-41.

The Integration of the Use of Computer Technology in Greek General Education

Dimitris Maritsas

Computer Technology Institute
Computer Engineering Department, University of Patra

The Greek experience

Computer technology began to appear in Greek schools in 1984. Until a few months ago, 23.2% of high schools were equipped with a lab of eight 8088 computers. The topic of informatics was taught in the third year of high school, usually by mathematics or physics teachers. 17.2% of Technical and all Vocational schools were equipped[1]. The use of computers in education has been perceived as serving to teach Informatics as a discipline and at the same teaching method and perspective has been retained as in the other curriculum topics. In relation to international models of computer use in schools, Greece has been adopting the "technical" and "isolated" model (Kontogianopoulou - Polydorides, 1991), i.e. computers have been taught as a topic added on to the curriculum. Recently, after severe criticism from the E.C., a policy document has been issued from the ministry regarding the national strategy for 1992-1997 (see footnote 1). In it there is an expressed tendency to move away from the technical and isolated model by including the use of computers in the curriculum topics and inserting computer use in the national teacher education centres for all the topics. Also, there is a plan for software development (not clearly stated) which seems to take this issue a bit more seriously than before. However, at the same time we are faced with situations and actions which are not clearly related to what is being planned.

Computer technology in compulsory education has been seen as a lower order and simplified version of what it is for tertiary, technical and vocational education. This is compatible with the structure of the Greek Educational system and the way curriculum content is formed (Kontogiannopoulou - Polydorides, forthcoming). It is also compatible with widely held perceptions of technology as a subject matter and not as a tool. It is not clear that the complicated issues requiring attention in order to encourage such deep changes are being met.

The problem so far has been addressed only in a top - down way. This is in itself a contradiction to allowing teachers to develop their teaching strategy with the help of computer technology. Decision making on teaching method has to come down to school level (Budin, 1991).

The problem has been addressed in a fragmented way - there is no synergy amongst the establishments involved especially at the bottom level.

"Big" and "one - instance" solutions have been the modus operande. (purchase of 22 labs in 1986, 180 in 1990, 208 in 1991, and 600 in 1993 (!) for general education high schools). Teacher training has been exclusively perceived as showing Informatics and Subject matter teachers alike "how to use the computers", i.e. information and practice on the operating system (Ms Dos and/or Unix), the network (Novel), Windows and s/w which commercially goes with the package, i.e. Cobol, Pascal a Spreadsheet. The only "development" in teacher training has been from the trainers being the technical staff of the companies providing the hardware or software, to engineers not necessarily belonging to those companies. No provision has been made on education on teaching methodology, either for the topic of Informatics or for other curriculum topics.

The plan (only a plan for the time being, when simple arithmetic shows how many computers are already

positioned in schools) for designing software does mention that tutorials, drill and practice programs and naive simulations are a thing of the past. However, the software is perceived as content specific and in general "protecting" the student and teacher from learning anything about the use of computer technology as an expressive medium. Furthermore, although it is stated that teachers will take part in designing the software, there is no plan as to how these teachers will be knowledgeable on the use of this technology to develop teaching and not just to reproduce it. Perceptions on how the computer will be used include those of an electronic blackboard, a simulation device, a data processing tool and a database. The use of the computer for the construction of knowledge it is argued will make that process faster. As for the types of software envisaged for general education, those are: Modelling and simulation, databases, graphics-animation and natural language understanding systems under advanced interfaces, multimedia-hypemedia, intelligent tutoring systems, animation and telecomunication. There is no need to explain to a Eurologo Conference why most of these applications and especially the way in which they are presented have nothing to do with the development of educational practice.

The international experience

Even though changes in educational practice are a very complex issue depending on a much wider grid of parametes than those related to technology itself, it is useful to use the descriptive model of developments in computer use in education forwarded by Polydorides - Kontogiannopoulou within the context of the COMPED research program; i.e. from a technical and isolated approach, to a pragmatic approach where the expressive power of computers is hidden from the user in favor of uncritical developments in interface, to the integrated approach, where, at least the software, allows users to construct meanings from both subject matter and the use of computer technology. It is interesting that the Ministry's proposal mentioned above, acknowledges that the current Greek paradigm is the technical and isolated one. But, what is more interesting, the other two paradigms are either used interchangeably, as if there is not much difference between the two, or as a developmental sequence where after the first step, one must strive for the second and only plan for the third!

The position of C.T.I.

C.T.I. put together a formal proposal after a request from the Ministry (Christodoulakis, Hatzilakos, Kynigos, Koutlis, 1992). Some points which are crucial to the position of C.T.I. are outlined here. The proposal is based on two premises; a) integrating the use of technology in compulsory general education is a serious task b) the task is of a different nature to that of integrating IT in technical, vocational and tertiary education and needs autonomous attention (its not a simplified version of it).

C.T.I. proposes an intergated developmental approach

integrated:
technical support,
software development,
teacher education and followup in the classroom
application in real schools and
academic (based at the University) supervision and concurrent research

happens **concurrently** and in close **synergy**

developmental:
objectives, methodology and practice are **continually refined** and **self - evolving**

Argumentation the integrated and developmental approach (see also "Astrolavos").

We can only say that a technology may have any long term educational use, when it is developed to suit the culture in which it is inserted, when teachers have been educated as to its use and educational "exploitation",

and they have been subsequently supported in the initial phases of their practice and their pupils have used it for a considerable period with tangible results in redesigned curricula.

An educational setting will always resist and finally reject a technology which cannot be used for self - enhancement at all levels and one which is imposed from the outside, whether that imposition is directed by policy or by society (Apple, 1991).

A continual link between practice and academic research is essential for the development of relevant educational goals. In this respect, in our country we have a serious problem: the absence of infrastructure which may enable an integrated and interdisciplinary approach to the problem. Specifically, Greece is missing an organised link amongst:

a) school practice,
b) school - based research and teacher education at University level,
c) institutions providing know - how for a culture - dependent production of the appropriate technology and
d) decision and policy - making authorities.

As will be elaborated below, we are in the phase of simultaneously "discovering" two things: the inadequacy - and its social and educational consequences - of our hitherto experience of a hurried purchase of too much technology in one instance (Kontogiannopoulou - Polydoridi, 1991); the potential for qualitative development in our educational practice through small scale case - study research in specific school settings (Kynigos, et al1993).

Serious and well organised piloting instead of "holistic" solutions

We therefore suggest that there is urgent need to setup the infrastructure for an integrated and interdisciplinary approach to the problem at a small scale. That is, to setup a small number of "poles" around the country, where local technological development, university based in - service teacher education and subsequent practice in suitably situated small groups of public schools are made possible and closely evaluated with a view to their subsequent direct generalisation.

Emphasis on the integrated paradigm for computer use in education

We see the computer in compulsory education as:

a medium for the symbolic and/or iconic expression of ideas
a means for communication
a tool for accessing, prosessing information
a tool for construction and control of models
a means for drawing and artwork
a simulation tool for experimentation

all these uses should be learned from within the curriculum topics. The curriculum, the teaching method and the evaluation method should be re-designed and qualitatively upgraded by the interdisciplinary consortia necessary. These uses may also eventually help to develop a less fragmented curriculum, i.e. where pupils carry out project work and learn about several topics within a project.

Prerequisites for success

Our policy and proposed strategy for implementing a progressive integration of the educationally aware use of computer technology in our educational system consists of the following points:

It is essential to adopt an interdisciplinary perspective of synergistic development of:

a) culture - dependent technological know how,
b) educational methodology
c) teacher education
d) school practice

In the initial phases of this process, detailed and systematic data collection concerning the pedagogical outcomes of this implementation within the specificity of our culture and society.

Within the above framework, we advocate a stepwise spread of the use of computer technology in schools, in three main phases: the "pilot phase" consisting of small scale carefully designed research implementations, the development phase where technology and teacher education is concurrently provided in a growing number of schools and the phase of establishement. We are well aware that, in England, e.g., the former two phases have lasted for about 12 years and not entirely convinced that the second phase is over yet, despite the level of the integration of technology in the U.K. National Curriculum.

Interaction and cooperation amongst academic research, school practice and technological development is preserved through all three of the above phases and the infrastructure enabling this begins from the pilot phase.

As far as the Greek system will allow in its present structure, a degree of choice as to the technology and the pedagogical method applied for its use should be exercised locally, perhaps at the level of the integrated educational settings mentionned above.

We are faced with the inevitability of an adjunct hour of "Informatics" and "The New Technologies" as new curricula topics for the first, second and third year of secondary. Under certain circumstances, this may be put to good use. According to how far the structure of an educational system allows development of pedagogical methods and practices, the use of computer technology may begin by direct integration with the already existing curriculum (we have examples of this in the U.K.) or go through an initial phase where pedagogical method becomes the object of educational practice in an adjunct topic of "informatics" to the curriculum. A hurried implementation of the former may bring about the use of new technology for old educational objectives and practice, something we would like to avoid at all cost, since international experience and history shows that it may well lead to the rejection of the technology itself and at worst, to a lost opportunity for educational development. On the other hand, a delay in the spreading of educational method and use of technology to a span of curriculum topics may well result in technology becoming an isolated object of study, and the other curriculum topics being deprived of its use.

Thus, the process of a pilot research phase has two further prerequisites: the timely development of educational objectives and technological know how and the use of the technology which education requires, independently of current cost or technological performance (see for example, the developmental process of the design of Logo and Boxer, diSessa and Abelson, 1986).

A few years ago there was not enough successful experience of mass introduction of technology in educational systems and in this sense the Greek experience was not atypical. The countries where there was a serious effort to achieve this during the 80's met with enormous problems mainly due to the lack of educational policy, i.e. a clear understanding of what exactly the technology would be used for (Solloway, 1990). Today there is a growing collective experience at least enabling a country to avoid major mistakes. However, it is not really possible to short - circuit the three - phase process and simply follow the naive equipping of a third of the schools with a computer lab by doing the same thing for the rest of the schools some years later.

What we would like to avoid and why

One of the most important characteristics of the computer is its flexibility as a tool. It goes without saying that technology may provide us with unsuitable tools for education, ineffective ones or even ones posing some danger with respect to learning or social development. Decisions may not therefore rest with informatics

scientists or specific subject specialists alone. On the other hand we may have a wonderful technological tool for education, well researched in the classroom and with important results regarding pupils' learning which is subsequently misused by the teacher who has not learned how to exploit its educational merits, nor how to improve old pedagogical methods. For instance, it is possible for a powerful computational expressive medium to be used as an object for rote learning. What is needed is cooperation amongst agents of technology, teaching and also learning, a central argument of this proposal. International experience shows that failures in attempts to use technological tools have their roots in the understandable initial lack of communication amongst the above agents. Some such phenomena we have observed and will attempt to avoid are the following:

Perceiving a computational tool as an object of knowledge and study (i.e. providing lengthy "lessons" on what is the computer, either in teacher or pupil education).

The use of computational applications or tools which are difficult to use or pose practical or conceptual obstacles irrelevant to the respective educational objective for their use (i.e. the teaching of technical code which is difficult to learn or does not have a sound conceptual base, in order to express simple meaning from everyday experience, see, for example Solloway, 1989).

The use of a large variety of poor quality software (there is a large volume of such software made by companies on the belief that they hit on a new market). It is often the case that schoools buy such software with the misguided objective to provide children with a breadth of related knowledge. The mistake is of course that not much breadth is covered at all vis - a - vis what the children learn, which in the end is not very useful in itself.

The development of poor quality software. This may seem self - evident but the large amount of such software shows that the issue is not trivial.

The restriction to one kind of technology and know how, with the consequence of a dependency on that technology and the extent to which it is developed in the future.

The use of immature or old technology. In the former case we have the danger of rapid development which will render purchased equipment obsolete while in the latter, what is learned has little relevance and use for the pupils' futures (e.g. command - driven operating systems).

Computational uses in the general education sector which place the computer in a directive and evaluatory role regarding children's learning attempt ro substitute the teacher in a way which is not acceptable educationally. The implementations have their roots in an era when computer science was detached from educational theory and mainly interest the world of informatics because of the challenge to model human behaviour and thinking.

Computational applications for rote learning. These applications do have a specific minor role in education, but not in this phase where we are attempting to draw out and implement general principles and specifications. However they certainly do not have a central role toplay at least in the nine year compulsory general education.

We know from the experience of such endeavours, especially in the European Community, that technology and pedagogical method for its use cannot be copied from one country to another. There are too many "idiosynchratic" characteristics regarding culture, society, educational practices and language for this to be possible. On the other hand there are lessons to be taught from each experience and this particular section is an example.

The above arguments corroborate the general argument for the avoiding of a mass purchase and installation of computer technology in schools, without the stepwise, careful and growing development of technology and method for its educational use.

The potential role of C.T.I.

Serious Technological Institutions are necessary in order to form an integral part of the chain of:

Technological support - S/W development - Curriculum and courseware design - Teacher Education - School Application

together with the Pedagogical Institute, Pilot schools, University Departments, under the Ministry of Education. C.T.I. being a leading R&D Organisation in Computer Science and Engineering in Greece (see profile), has recognised the historical importance of the venture of introducing the new technologies in the National Educational System. Having established its philosophy, C.T.I. is at the point of organising its effort, being fully aware that the issue deserves serious and responsible attention, avoiding "technological fix" approaches.

Conclusion

Integrating I.T. in general education is a serious and autonomous problem. It needs a synergistic serious addressing by pedagogical, technology, university and ministry establishments. C.T.I. is an example of such an institute with the added property of the status for the above job.

Profile of C.T.I.

CTI is predominantly an R & D Organization. Its whole structure, its targets and motivations put the R & D activities in first priority. CTI is open to research areas of the Science and Technology of Computing, and in a sense, it contributes to the shaping of the future of these areas. Basic Research conducted in CTI has long term visions and goals. Applied Research and Development play an important role in the support of the Greek Informatics Industry in the framework of the European Community. CTI main requirements in Applied Research and Development are high quality of the resulting products and reliability as well as effectiveness of the implemented prototypes.

CTI is well recognized and internationally competitive in research areas as, Algorithms Design, Distributed Computing and Telematics, High Performance Computing, Software Technology, Advanced Architectures, Digital Signal and Image Processing etc.

The area of High Performance (and Networking), which includes issues of Parallel and Distributed Computing, is of particular emphasis. CTI is an major share-holder of the "High Performance Computing Centre of Athens" together with other major Greek Academic & Research Institutions such as, the National Technical University of Athens, and the University of Athens. The choosing of High Performance Computing and Networking as an area of emphasis is one of the strategic decision of CTI for its future position in Research.

CTI is also well established in areas as Medical Computing, Artificial Intelligence, Multimedia, Graphics and Image Processing, Geographic Information Systems etc.

The Applied and Basic Research effort is strongly supported by fundamental research in Algorithms Analysis, Computational Geometry, Data Structures, Stochastic Systems, Signal Processing and other disciplines.

To efficiently conduct its research activities CTI cooperates with other prominent Institutions either local (University of Patras, Scientific Park of Patras) or at a national or international level.

The Applied R & D activities of CTI are also strongly supported by the existing modern computing infrastructure. This infrastructure was possible to build through the strategic Mediterranean Integrated

Programme (MIP) for Informatics.

This programme supported CTI with 5.5 MECU for a period of 5 years. The CTI itself as an integrated and competitive Institution (able to play a catalytic role in the development of Informatics in Greece) was a deliverable of the MIP Programme. In this sense, the European Communities (and especially the Peripheral Development) have essentially contributed to CTI's growth. Most of the funded research today comes also from the CEC's competitive Research Programmes. Thus, CTI mostly gains its support (in a competitive way) from the European Community and its R & D framework.

Overall, during its few years of existence, CTI has put its mark and did a measurable contribution to the Research, Development and Technology Competitiveness of Greece. Its well - recognized activity in European and world-wide R & D justifies the efforts and funds invested in it.

Note:

[1]Program for the Extension of the Introduction of Informatics in Primary and Secondary Education, Directorate of Secondary Education, Office of Informatics, Ministry of Education.

REFERENCES

Apple M W (1991): The New Technology: Is It Part Of the Solution Or Part of The Problem In Education? Computers in the Schools, vol 8 (1/2/3), pp. 59-81, Haworth Press.

Budin H R (1991): Technology and the Teacher's Role. Computers in the Schools, vol. 8 (1/2/3), pp 15-26, Haworth Press.

Christodoulakis D, Hatzilakos Th. Kynigos C. and Koutlis M (1992): Astrolavos: A Program to Utilize Computing Technologies to Support Education in the Hellenic Secondary Schools, Proposal prepared for the Ministry of Education under the supervision of Prof. D. Maritsas, Director of the Computer Technology Institute.

A. DiSessa and H. Abelson, BOXER (1986): A Reconstructible Computational Medium. Communications of the ACM vol 29 no 9, pp. 859 - 868.

Kynigos C, Gyftodimos G. and Georgiadis P. (1993): Empowering a Society of Future Users of Infromation Technology: A Longitudinal Study of Application in Early Education, European Journal of Information Systems 2, (2), pp. 139-148.

Kontogiannopoulou - Polydorides G (1991): The Educational and Social Dimensions of the Use of the New Technologies in the School, paper written in Greek in Contemporary Issues, vol. 46-47, Dec. 1991, pp. 77-93.

Kontogiannopoulou - Polydorides G (forthcoming): Computer Education Policies and the Cultural Context.

Papert S (1987): Computer Criticism versus Technocentric Thinking, Educational Researcher, 16 (1), pp. 22-30.

E. Solloway, Quick (1990): Where do the Computers Go?, Communications of the ACM, vol 34, n. 2, pp. 29-33.

E. Solloway (1989): Highly Interactive Computer Environments. GOLEM Newsletter of Technology and Education, vol 1 n.12, pp. 9-10.

The politics of Logo

Richard Noss

Dept of Mathematics, Statistics and Computing
Institute of Education, University of London

The education system demands of everyone alike that they have what it does not give.
Pierre Bourdieu

The problem

Seymour Papert provided us with an educational vision in which the 'artificial' and 'inefficient' learning environment of the school is seen as a result of the impoverishment of informal (extra-school) environments. The alternative he offered, in which much which is currently learned by formal instruction could be accomplished 'painlessly', and 'without organised instruction' through interaction with suitably-designed computer environments, offered not just an alternative scenario for education, but the hope and promise of its realisation.

It is only thirteen years since the publication of *Mindstorms*, a period which is much too short to determine whether Papert's revolutionary prognoses are likely to be realised. On the other hand, we have not, of course, seen schools 'wither away'. This is hardly surprising, and it seems to me that there is nothing much that needs explaining in this respect. Papert's claim was not that Logo's creation would magically dissolve the traditions of over a century of state-funded schooling[1] or that the act of programming a computer would automatically lead children to the promised land of mathematical (or any other) competence.

My reading of Papert's idea that schools would 'have no place', is basically a psychological one. And at this point, I think there *is* something to explain. I would like to understand why, to put it bluntly, the opposite of Papert's vision is taking place. Far from withering away, schools are called upon to play a greater and greater part in the education of young people. Far from being used as an instrument of intellectual emancipation, computers are used — both inside and outside school — as a mechanism for controlling people rather than liberating them. Almost everywhere, Logo has become — in its school instantiation — a shadow of itself, a means to draw pictures, learn about angles, develop 'problem solving skills'. From a mathematical perspective, there are few indications of any serious reappraisal of the content, rationale and methodologies involved in mathematical education as a result of the advent of Logo. In fact, there are signs — for example, in France and the US — that Logo is seen as having run its course, 'been done' and found wanting. In order to understand these processes, we need to adopt a political, not just a psychological, response.

How is Logo used?

It is worth beginning by stating that, whatever the state of Logo in schools, it is not helpful to claim — as is currently fashionable — that it is all 'the fault' of teachers. Suppose, for a moment, that this were true — what would it mean? It would mean that a 'pure' innovation — 'Logo', 'the computer' — had been 'perverted' (either intentionally or unintentionally) by the practitioners who used it. Even if this view could be sustained, we would still need to know how and why this perversion had taken place, unless we were content simply to blame teachers' general incompetence (even then there would be something to explain). The point is simply that the 'teachers' fault' view is totally lacking in explanatory power.

In UK primary schools, my impression is that Logo is most often seen as an elementary geometrical program, or simply as an exercise in enjoyable computer interaction. In secondary schools, the situation is complicated by the tendency to ghettoise machines into laboratories where they are used for 'computer studies'. If Logo is used at all in mathematical settings, it is often to tackle worksheets 'investigating' triangles, squares and the internal angles of regular polygons; or as a means to provide light relief within the mathematics classroom

— equally as a reward for those who have succeeded in completing their quota of 'real' maths activities, or as a device to neutralise those who have failed.

There are, of course, some notable departures from this depressing state of affairs (as evidenced, for example, in the pages of *Micromath*). Nevertheless, even this spirit of creativity and innovation has been dampened by the advent of the Mathematics 'National Curriculum', in which Logo appears precisely twice:

Target	Statement of Attainment	Example
AT1/4 Using-applying Maths	Identify and obtain information necessary to solve problems	When trying to draw repeating patterns of different sizes using LOGO, realise the need for a procedure to incorporate a variable, and request and interpret instructions for doing this.
AT3/3 Algebra	Use inverse operation in a simple context	Use doubling and halving, adding and subtracting, and FORWARD and BACKWARD (in LOGO) etc., as inverse operations.

The remaining three 'targets' — 'Number', 'Handling Data', and (incredibly) 'Shape and Space' contain no explicit reference to Logo.

From *Mathematics in the National Curriculum, 1991*

Thus a decade of research, curriculum development and in-service education based on Logo, is reduced to an ability to 'request' and 'interpret' the need to use an input (incorporate a variable — *sic*), and the (mathematically questionable) recognition that FD and BK are 'inverse' operations.

Perhaps we should not make too much of such trivialisation. After all, the UK National Curriculum was thrown together by a heterogeneous collection of people, at very short notice, without any meaningful consultation — and, as it happens, I am unable to locate any person who confesses to having participated in the secret discussions which led up to it, who will simultaneously admit to knowing anything substantial about Logo. Nevertheless, the existence of Logo in the mathematics National Curriculum, provides some (albeit small) insight into the way it is conceived in educational circles, the prevailing wisdom concerning what Logo is, and what it might be for.

Of course, it would be wrong to suggest that the 'statements' outlined above reveal the extent of Logo in mathematics classrooms or more generally. Moreover, it is important to clarify that it's use depends strongly on a variety of factors, not least the social class of the children who use it. I am unaware of research which has focused on this issue with respect to Logo, but Michael Apple (1987) has demonstrated a clear link (in the US) between social class, gender and race and how computers are used in schools: he shows that the number of computers is substantially *greater* in working-class schools than in middle-class schools. Similarly, Paul Olson (1987) points to a marked tendency for the children of the middle-class to use open-ended applications and to learn programming (language not specified), while schools in working-class areas tend to rely on mechanical drill programs and games. As I have stated, there is no corresponding work in relation to Logo, but I think it is not too far-fetched to postulate that a similar pattern would be found. Certainly, my impression is that the further down the social scale one travels, the more likely one is to encounter Logo as a rather inefficient tool for drawing houses.

Clearly we are dealing here with much more than the whims of teachers, or the preferences of individual children. The issue is how Logo is *constructed* as a social/educational phenomenon; what niche it is seen to occupy among the panoply of existing social and educational relations. One tiny anecdote which provides insight into the issue, is provided by Doug Clements[2], who recalls noticing suddenly that Logo had been removed from the computer network at his University. On asking the network manager for an explanation, he was told that 'DAZZLEDRAW' had arrived, and so he had assumed that Logo was no longer needed!

For Clements' network manager, Logo was a drawing tool which could be dispensed with as soon as something else became available; he had *constructed* Logo as a rather clumsy paint program whose educational, philosophical and mathematical characteristics were irrelevant. The point is not whether he was right or wrong, but to try to understand how such a 'misunderstanding' could occur. In order to do so, I am drawn to think about the social relations of technologies and technological change.

Ways of viewing technology

It is commonplace in both popular and academic literature to attribute to technological artefacts some degree of autonomy in the process of societal change. For the optimists, technology is seen as solving all problems, and the evidence is there, residing on the disk, or in the resolution of the screen. For the pessimists, there is simply *not enough* technology, it is too expensive, it is too primitive. This technological determinist view is routinely deployed to explain processes which range far beyond those in education to theorise changes in society, and forms the basis for many 'popular' accounts of history. For example, when I was training to be a teacher, a nameless tutor asserted that the collapse of the Roman Empire was due to the Romans' failure to develop a number system based on place value! By suggesting that a technology itself may have the power to revolutionise a society, technological determinism proposes that the technology is constituted independently of those who use it.

In mathematical education, it is routine to encounter findings which claim to have observed 'effects' of 'the computer' on children's learning; such accounts almost always ignore the settings in which the computer is employed, the social interactions among children, computer and teacher, and broader questions of the nature and purpose of the computer's presence in the educational arena (see, for example, Pea & Kurland, 1984). Papert has referred to this way of thinking as 'technocentrism', a specifically educational corollary of the general phenomenon of technical determinism. So rather than list further examples of its most obvious manifestations, I prefer to do the opposite: to give an example of its (apparent) converse.

A recent series of workshops organised centrally throughout the UK aimed to 'introduce' teacher-trainers to the role of the computer in mathematics classrooms, in the hope that they would pass this knowledge on to their students (prospective teachers). Those involved circulated an outline of their course. In the UK at least, it is now virtually obligatory to incorporate into such material a statement such as 'The focus of this course is on the educational issues, not on the computer'. Such a disclaimer is intended to absolve the writer from any hint of technocentrism, to display impeccable credentials as someone who cares for learners, for teachers, for education — not technology. And so, one course provider set about such a course, advertising that most of the work would be done without a computer, as *'all the interesting ideas are nothing to do with the machine'*.

Alas, things are not so simple. For such an approach, however 'pupil-centred', takes for granted that the computer cannot bring anything new to the learning situation: it guarantees that the computer is cast in a conservative role, unable to catalyse change of any significant kind. Moreover, it is, in fact, a technological determinist position, for it precisely views the computer as an asocial object — in this case, an irrelevant object rather than an omnipotent one — rather than as a new medium in which mathematical ideas can be expressed.

An alternative perspective starts from the view that the very notion of 'the technology' is problematic. In an educational setting, the challenge is to examine how the notion of, say, the *computer* is constructed by teachers and learners, to focus on the meanings evoked by it and the ways in which it functions as a set of *cultural symbols*. Viewed from this perspective the individual 'user' of a technology is seen as socially constructed: actual use is governed by the ways in which the user 'reads' the set of cultural signs which constitute the technology.

This perspective recognises that in terms of the way in which they are employed, technologies enter as active agents into the social arena — not, as the technological determinist view would imply — as passive artefacts. But what of the production of technologies? The general point is that technologies do not develop according to some inner logic of their own. Rather they develop as the function of a complex set of social, technical, economic and political factors. Each technical artefact — or piece of software — contains the congealed outcome of a set of negotiations, compromises conflicts etc. Technological artefacts represent the expression of a complex array of social and political choices and decisions which are frozen beneath their surface (or onto their chips).

The social shaping of technology

I begin with an example. Some time ago, I discovered to my cost (literally) that monochrome film now costs almost twice as much to be developed commercially as colour. When I asked for an explanation, I was told that this was because 'the machine has to be set up specially'. This is despite the considerably simpler

process and cheaper chemicals involved in monochrome processing. Presumably this technological change signals the end of mass black-and-white photography. Moral: the ways in which technologies develop are comparatively little to do with the inner logic of that technology and much more related to the social forces underpinning the production (and distribution) process.

But that is not all: such changes reinforce the view that photography is 'about' reproducing reality as 'faithfully' as possible (I do not discuss the truth of this assertion: only that it is one among many possible views of photography). From this perspective, colour is better because it is closer to reality, a more faithful representation of what is photographed. And so there is a dialectical relationship between the way in which the technology evolves and the ideas (or ideologies) which it gives rise to (and, recursively, which shape the technology). The social and economic spheres shape the development of the technology, and *simultaneously* legitimise particular views of what the technology is about.

This perspective is a variant of what has become known as the 'social shaping of technology' perspective. This approach focuses on the construction, design and development of technologies from the perspective of the network of social forces and interactions on which design is based, and which crucially shape the invention/production process (for an interesting introduction to the subject, see the collection by Mackenzie and Wajcman, 1985, and the helpful review by Mackay and Gillespie, 1989).

The development of programming languages is an interesting instance of social shaping. There are several hundred programming languages in existence, each formally equivalent (in the sense that each can be reduced to the native machine code of the computer on which it runs). But this formal equivalence masks considerable disparities in the ideologies surrounding each language (or its 'culture') and these are in no small part related to the purposes for which they were designed. For example, ADA is a computer programming language developed and used for missile guidance. There are, as the reader can imagine, some rather stringent requirements for such a language, but not ones which would necessarily converge with educational priorities!

I recently came across an article which argued that BASIC represented an important tool in the mathematics teachers' toolkit (MacKernan, 1991). The argument was summed up as follows:

> 'I happen to believe that Basic (sic) is an important educational (and for many of us an important mathematical) programming language.'

The author went on to argue that printing out the first ten square numbers was a simple task to accomplish in BASIC, and that this provided an example of a rationale for teaching children the language.

I could not resist rising to the bait. I began by trying to refute the argument from a psychological point of view, but it soon became plain to me that I was involved in a political discussion, not a psychological one. I reproduce here part of my reply (see Noss, 1991):

> "BASIC is neither important educationally or mathematically. I take this position not *only* because BASIC is less intellectually powerful than, say, Logo or because it appeals to a much narrower style of learning and thinking. These factors are important but they miss the point. BASIC is unimportant because it looks to the past: it is not evolving, it does not open new areas of mathematics to explore, it does not significantly widen access to the power of mathematics.
>
> I have a vision of what mathematics teaching could be like using computers. I think it is relevant that whereas an educated person has little difficulty having some feeling about what, say, a researcher in Literature, Music, or even Physics does, the same is not true of Mathematics. 'But what is there to find out?' is a common question. Worse still, it is hard to respond. Research mathematics (or even degree mathematics) is not just harder than school mathematics, it appears as a completely different subject from that which is presented at school. We do not even provide children with a language which enables us to communicate the power of mathematical ideas within our culture.
>
> And yet, a book on the history of the development of non-linear dynamical systems (James Gleik's book on Chaos) can become a best-seller. A television programme on mathematical puzzles can unseat Terry Wogan. And that damned cube can catch the imagination of every other child. Mathematical ideas — in so far as they can be made accessible to novices — undeniably hold a power and a fascination.
>
> This fascination takes place despite what we teach children in school. Even the most interesting examples of mathematics teaching which appear in the teachers' journals are only scratching the surface, pointing to little pieces of mathematics but rarely aiding in the process of helping children see what mathematics

is about. This is nobody's fault. What else can any of us do? Schools do not teach mathematics because they want children to harness the power of mathematical thinking (alongside the thinking that comes from studying literature, science or music). We teach children maths because children have to be taught maths (for social, cultural and political reasons). And maths, as I have said, has little in common with mathematics.

As I said, I have a vision. I believe that educated adults *could* have at least the same degree of acquaintance with mathematics as they have, say, with music, art or literature. I think that the power of mathematics for constructing explanations of how the physical and social world works is unparalleled. And I think that the computer can help in this task.

It is important to realise that the history of educational computing in schools is little more than one decade old. In just that time, we have come a long way, but we are still only in the very earliest stages of understanding how computers might be used for (mathematical) learning. It is now fashionable (and correct) to dismiss the ubiquitous drill programs which filled the catalogues until recently. A few years ago, this debate would have been about their educational value. Just over twenty years ago, Papert's colleagues at MIT dismissed as ludicrous the idea that any school would ever have access to a computer powerful enough to run Logo-graphics. Now at last, there is beginning to be a small but significant crop of mathematically serious programs being produced [...]. Some of the best are trying to incorporate programming power into their structure, so that users will be able to do more than simply 'customize' their program — they will actually program in it. What is more, there are efforts being made both to evolve the Logo family of programming languages as well as revolutionize it by the introduction of new computational power.

Like it or not, not a single one of these initiatives is taking account of any of the ideas underpinning BASIC (I take it for granted that they are not using BASIC — but I am saying more than this. The point is that none of the structural characteristics of BASIC are considered of any interest for this new generation of software). And there is, of course, good reason for this. BASIC belongs to a philosophy and practice of computing which no longer resonates with any of the technical and intellectual constraints which gave birth to it.

At root, the debate is about whether a programming language is really a tool for solving problems or a means of expression. I can order a meal in about four languages, but I can only (lamentably) express myself broadly in one. Similarly, there is no difficulty whatsoever in teaching children to print the squares of the first ten numbers in a wide range of programming languages. But if we want children to express themselves — and more significantly, to express themselves mathematically — we had better carefully choose the language we offer them. If we want a suite of short programs to solve a suite of soluble problems, the programming language is not crucial (it is still important, but I let that pass). But if we want open the doors of mathematics wide to children and adults, to present them with a means for genuinely exploring half-understood ideas and expressing half-coherent thoughts, then the choice of programming medium is absolutely critical.

The ideal language for such activity does not exist. I would be the first to admit that Logo is only an indication of what might be offered in the future. But the fact that it is an indication is significant. Seymour Papert is fond of quoting Warren McCulloch: 'Don't look at my finger, look where it's pointing'. If teaching mathematics is going to remain at the level of printing the squares of the first ten integers then it is really immaterial what language is used. To be frank, it might be better to have a numerical package — complete with icons, pointers, musical accompaniment and colour-effects — rather than a programming language at all. If on the other hand, we are trying to change the way children and adults think about mathematics, to incorporate far larger sections of society into mathematical culture, we had better look to the future, at pointers towards the computer as a medium for expression rather than simply a convenient tool to 'deliver' whatever happens currently to be in the National Curriculum."

We cannot easily separate the ways in which Logo (or BASIC) are seen, the way their signs are read, from the uses to which they are put. The actual role of Logo in the mathematics curriculum presupposes a (social) construction of the curriculum as much as a construction of Logo: it is not easy to see how Logo and the mathematics curriculum can peacefully coexist if the construction of one is widely at variance with the other. What are the elements of the ways in which the mathematics curriculum is 'constructed' in the developed world? What are the political imperatives which coalesce around it, and which lend it its particular role within the educational and social system?

The politics of school mathematics

I will start with an example from the UK's National Curriculum (see Noss, 1990):

> *This problem is appropriate across the age range 7 to 16.*
>
> If British Airways runs flights between each of 8 major airports in Europe, how many routes is that? What happens for a different number of airports?
>
> [Two 'lines of solution' are then provided]
>
> The mathematics of the above example is exactly the same type (sic) as that needed in a variety of other contexts.
>
> For example:
>
> a) The offices of 15 cabinet ministers are linked by direct telephone lines. How many lines are needed?
>
> b) If 30 people at a party shake hands with each other, how many handshakes will there be?
>
> *This use of the same mathematical content in widely different situations illustrates the power and beauty of mathematics .*

Mathematics 5-16. Department of Education and Science, 1988, § 7.23; emphasis added.

There is a clear view of mathematics here, as 'content' which can be 'exploited': it presupposes a similarly clear view of the way mathematics enters into people's lives. Elsewhere in the same report, there is a statement that having selected the 'appropriate mathematics', pupils should 'apply it sensibly and efficiently, try alternative strategies if needed, check on progress at appropriate stages and analyse the final results to ensure that the initial requirements have been met' (ibid, § 4.12). Reading this description, it is not difficult to believe that its authors had in mind a clear shop-floor image; a problem posed (by someone else), some maths needed, a result obtained with maximum speed and efficiency.

Of course, this kind of political perspective on mathematical education is not new, and resonates nicely with other perspectives on the rationale for teaching mathematics which have echoed throughout the last century. Howson, Keitel and Kilpatrick (1981) point out that there has been a view stretching back to the mid nineteenth century that arithmetic could foster 'the habit of promptitude...exactness and order' and that mathematics instils the 'acquisition of necessary truths' (p. 24) — a nineteenth-century statement of aims not so much at variance with that found in the D.E.S. report above, which describes its 'programmes of study' as seeking 'to draw out each child's full potential through the development of sound work habits, self-discipline and industry together with good personal qualities' (ibid § 8.4).

It is within this framework that we can, for example, make sense of the recent decision to reintroduce — as a compulsory element of the UK mathematics curriculum — long division and multiplication, at precisely that point in the development of human knowledge which makes such 'skills' totally anachronistic. In terms of mathematical learning, there is no perceived rationale on the part of governments — and neither has there ever been — for educating the mass of children to seek pattern and structure within their social and economic surroundings, or to gain a critical awareness that will allow them to interpret and change their increasingly mathematised reality. To put it crudely, long division has not been imposed because anybody thinks it is useful; it has been imposed because it is acts as an efficient and workable mechanism of social control. It may not help on the production line, and it certainly will not be of assistance in the dole queue: but it might just help the pupil to know his or her place. Divide and rule indeed.

Perhaps that is the image which we would like children to have of mathematics, but it is not one I share. But if it at least partly represents the reality of the situation, it is reasonably obvious why Logo as envisaged by Papert would have to be re-constructed to fit into the scenario I have outlined.

From the UK perspective, I think the process of reconstruction has been reasonably straightforward to describe. The first reaction of the educational body politic was to ignore the possibilities of Logo altogether,

and dismiss it as merely a program for infants. But the problematic nature of BASIC, the slow but sure emergence of a range of creative teacher-driven applications, and the rather clear relationship between mathematics and Logo which was maintained in the UK — all these converged in the late nineteen-eighties to establish Logo's right to be discussed as a serious part of the mathematics curriculum. A 'think tank' on the mathematics curriculum in 1985 lauded BASIC, and nodded condescendingly at Logo for elementary uses. A similar exercise earlier this year did not mention BASIC — in so far as programming was conceived as a legitimate activity at all (and this is as far from being accepted as it ever was), Logo was the only contender.

And so the tactics changed. I do not mean to speak of an educational system as if it maintains a conspiracy to oust Logo (or, for that matter, anything else). It is simply that the system — like any organism — organises itself against attack from within or without. And in this case, as we have seen from the National Curriculum snippets, the 'strategy' adopted was not simply to ignore Logo but to rebuild it in the image of the existing curriculum, the existing culture.

I recently heard a curious story that illustrates the case. A member of the working party responsible for the UK National Curriculum, and hitherto implacably hostile to all things Logo, is rumoured to be commissioning Logo materials for A-level mathematics (University Entry), on condition that it employs WHILE..DO loops! A bizarre story which, by itself, tells us only about the peculiarities of an individual, not of a system. But taken as part of a broader picture, we see a way in which Logo is marginalised, remoulded as harmless to the educational organism, redrawn in the image of PASCAL rather than run the risk of allowing it a chance to act as any kind of catalyst for rethinking the content of what is taught — or, more importantly — what needs to be taught at all.

Before I conclude, I want to return to the British Airways problem to point to the extreme artificiality, not only of the examples a) and b) but of the original problem itself: just what is the role of the inclusion of British Airways in the problem? It can hardly be that this is a 'real problem' of the kind which people need to solve in their future work. And it is far from clear how the embedding of the 'content' in the setting of linking cabinet ministers' telephones is supposed, in some way, to expose the power and beauty of mathematics in making sense of situations that are real for pupils. This question of artificiality is, of course, not a new problem for mathematics education, and the authors of the report are not alone in their view that problem-solving is essentially concerned with the solution of artificial mini-stories full of spurious contextual clutter but devoid of meaning.

The point here is that the explicit reinforcement of this view, that mathematics is essentially meaningless, is an important way in which mathematics functions as a political element in the curriculum. That mathematical problems are essentially without meaning seems not only to be deeply ingrained in the popular culture, but essential to its character in the curriculum. And much the same might be said of Logo. After all, the meaning (for a child) of a Logo program is dependent on the breadth of meanings which underpin the program — its rationale, its motivation, its status as a meaningful activity for the child. Without that meaning, Logo does just become another programming language which might just as well be taught as an example of WHILE...DO loops as any other.

I reproduce here a story which I published some time ago, but which I think still retains much of its impact (Noss, 1988) in terms of the meaninglessness which characterises much of school mathematics:

> "I recently gave a lecture in which I presented a class of 40 pre-service students (not mathematics majors) with a number of mathematics examination questions which contained rather clear political, moral or other value judgements. The students agreed that one of these questions was both violent and racist (it involved the well-known 'puzzle' concerning Christians throwing Turks overboard in a storm-hit ship, with a view to drowning only Turks), and I suggested to them that they might like to recast the question in 'value-free' terms, to see if such an exercise was feasible or desirable. The results overwhelmingly conformed to 'text-book' mathematics problems, with reference to human beings expunged altogether and meaningless references to beads and other artefacts of the mathematics classroom replacing human beings. Just over half the replies simply replaced 'Christians' and 'Turks' with various kinds of confectionery (sweets and toffees predominating) and fruit (mostly apples and oranges). It seems that the price that is paid for removing value judgements from mathematical problems is to literally de-humanise them. In comparison, a recent examination

question in the UK which invited examinees to compare the total amount spent on armaments (by the US and the USSR together) with the amount needed to feed the world's population, received a prolonged and unanimous attack in the national press. Small wonder then that mathematics lessons are so often 'not about anything' (D.E.S. 1982). "

This lack of meaning is perhaps the most critical challenge facing mathematics education — perhaps education in general. It has lead thinkers like Ivan Illich to the conclusion that meaning is not simply lost, it is irrelevant — perhaps even obscured deliberately:

> **It does not matter what the teacher teaches so long as the pupil has to attend hundreds of hours of age-specific assemblies to engage in a routine decreed by the curriculum and is graded according to his ability to submit to it.**
>
> Illich (1973) p.61-2

I do not propose to debate the pros and cons of Illich's contention, except to state that there is no need to accept Illich's pessimism uncritically: schools and educational systems are arenas of struggle in which teachers, pupils and parents are not merely passive recipients of policy decisions, they are actors on a political stage and, as we have just seen in the UK, they *are* able to effect change, decide what is in children's best interests, and choose between educational priorities[3]. Nevertheless, in the case of Logo, it is fairly clear that it is precisely the potential for *meanings* that Logo has offered, and it is precisely its *meanings* which have been stripped away in order that it become adapted to its educational environment.

Conclusion

Where do the meanings of Logo come from? How does Logo afford the child the chance to inject meaning into activities, and to discover meaning within them? I think one answer must be its *informality;* in ways which are too obvious to outline at a Eurologo Conference, I think we can agree that Logo offers playful and goal-less activity; it allows a medium for articulating half-formed fragments of ideas which can be cohered into concepts; it provides access to different representations (like the visual and symbolic representations in turtle-graphics); and it affords entrance to a range of cognitive styles. It builds on the informal.

If Logo's main claim[4] is to *informalise the formal*, it stands in essential opposition to the social objectives of schooling, which are precisely to privilege *formal* discourses *at the expense of the informal.* I began with a statement of Pierre Bourdieu's. One way of reading Bourdieu's statement is precisely this: that the education system demands a level of informal knowledge on which its (exclusively) formal pedagogy can be built. Schools do not offer these informal networks of ideas, these 'conceptual frameworks' — they offer formal, compartmentalised, hierarchical and testable knowledge. Those who arrive at school without such informal networks (and, of course, working class children are much more likely to be in this position than middle class children), do not, in general, find such networks of informal knowledge readily on hand in school or outside.

Those practices which fit most readily into the school setting are formal, hierarchical and subject bound. In contrast, the educational power of Logo lies in its potential for environments which are informal, heterarchical and blur subject boundaries. It is not surprising therefore, that a resolution of the difficulty is precisely to (re)construct the computer as a mechanism which is little more than a delivery system for existing pedagogic practices; we should not be surprised to discover that the computer is routinely used to formalise rather than informalise children's learning.

Bourdieu is talking about the education system; and so, I think, is Papert. In this respect, environments like Logo *do* make the educational system redundant, they do 'give' what — until now — they demanded that all already 'have'. But making an education system redundant in terms of what it offers to intellectual development, is not the same as making it redundant in practice. Educational systems are part of a much broader list of political priorities, and individual children's development is low on that list.

I do not feel it is especially encumbent on me to end on a positive note, as I do not believe the situation calls for great optimism. But I will end with one recent anecdote which I think raises an interesting possibility. On a recent visit to Mexico, I received an invitation to spend an afternoon with the Latin American Educational Center of a well-known computer multinational, to talk about Logo. With some trepidation I

arrived at the expensive building, my prejudices intact, confident that I would be cast in the role of a hopeless romantic while I could smugly dismiss them as faceless businessmen who wouldn't know Logo from a logo. I was wrong. I found more than a sympathetic hearing, I found they knew as much — if not more — than me; not just about Logo the language, but about Logo the educational philosophy, Logo the ideal, Logo the (hoped-for) catalyst for radical educational change. As my hosts outlined their truly radical vision for education within the whole of Latin America, and recounted their daily communications with Presidents and Ministers who, they said, shared their vision, I wondered what could account for their optimism and their sense of mission.

I have a possible explanation. In the 'developed' world, there is a crisis of educational overproduction. People are simply too educated for the unemployment and recession most western economies find themselves in. In the words of an official in the Department of Education and Science, 'People must be educated once more to know their place'. To be sure, there is still a need for a very small number of very highly educated persons. But for the rest, a primary aim of the education system is to ensure the reproduction of the existing social order. Education (in the positive sense) is an encumbrance.

In the third world — at least those parts of it who are not still in the grasp of feudalism or military dictatorship — the situation is radically altered. Here education is not an encumbrance, it is — potentially at least — a means to escape economic decline and to outlive centuries of colonial domination. In many third world countries, the crisis is partly caused by a massive *under*-production of educated people, insufficient even to fulfill the function demanded of consumers and workers within a modern economy. And so, a crucial step on the road to 'modernisation' is to build an education system which gives both formal *and informal* knowledge to its pupils (*pace* Paulo Freire).

I am not naive enough to believe that such a strategy is entirely altruistic on the part of governments. But I do think that it might be possible for third world countries to skip the growing pains of computers in education which have been reproduced in western country after western country. I do not know enough about the third world to know if the problems of teacher education, resources, and the design of suitable pedagogies can be overcome. In the first world, there are few signs that school systems are exploiting the computer to offer the majority of children who fail in school a chance of educational success. But at least Logo and its successors have the potential to give what they do not have.

Notes:

[1] I am thinking of the so-called 'developed' world here. I consider other parts of the world (briefly) at the end of this essay.

[2] Personal Communication. Also reported in Hoyles, 1992. Plenary address to the Sixteenth International Conference for the Psychology of Mathematics Education, Durham, NH, USA.

[3] English and Welsh teachers have just successfully boycotted the first round of National Tests on the grounds (among others) that they are educationally invalid.

[4] It is interesting how hard it is to avoid technical determinism in language.

REFERENCES

Apple M. (1987) Mandating Computers: The Impact of the New technology on the labour Process, Students and Teachers. In: Walker S. & Barton L. (Ed.) *Changing policies, changing teachers.* Milton Keynes: Open University Press.

Bourdieu P. 1973 *Cultural Reproduction and Social Reproduction.* In: Brown R. (Ed) Knowledge, Education and Cultural Change. London: Tavistock 1973, p. 80

Hoyles C. & Noss R. 1992 (Eds.)*Learning Mathematics and Logo.* Cambridge: MIT Press.

Illich I. (1973) *Tools for conviviality.* London: M. Boyars.

Mackay H. and Gillespie G. (1989) Extending the social shaping of technology approach: ideology and

appropriation. Dept. of Behavioural and Communication studies, Polytechnic of Wales.

Mackenzie D. and Wajcman J. (eds) (1985) *The social shaping of technology.* Milton Keynes: Open University Press.

MacKernan J . (1991) In defence of Basic. *Micromath.* 8,1,15-17.

Noss R. (1988) The Computer as a Cultural Influence in Mathematical Learning. *Educational Studies in Mathematics*, 19, 2, 251-268. 1988.

Noss R. (1989) Just testing: a critical view of recent change in the UK Mathematics Curriculum. In: Clements K. and Ellerton N. (Eds.) *School Mathematics: the challenge to change.* Deakin University Press.

Noss R. (1990) The National Curriculum and Mathematics: a case of divide and rule? In: Dowling P. & Noss R. Mathematics versus the National Curriculum. Falmer Press. pp. 13-32.

Noss R. (1991) In defense of Mathematics. *Micromath.* 8, 1, 18-20.

Noss R. (1992) The Social Shaping of Computing in Mathematics Education. In: Pimm D. & Love E. (eds) *The Teaching and Learning of School Mathematics.* Hodder & Stoughton.

Olson P. (1987) Who Computes? In: Livingstone D. et al. (eds) *Critical Pedagogy and Cultural Power.* London: Macmillan

Papert S. (1980) *Mindstorms.* New York: Basic Books.

Pea R. and Kurland M. (1984) On the cognitive effects of learning computer programming. *New Ideas in Psychology.* **2**, 1.

SPONSORS

The Organizing Committee of the Fourth European Logo Conference gratefully acknowledges the contribution to the organisation of the Conference of the following:

HELLENIC MINISTRY OF CULTURE

MINISTRY OF INDUSTRY TECHNOLOGY AND RESEARCH

OLYMPIC AIRWAYS

ANTINOOS

APPLIED SCIENCE

CONTROL DATA GREECE INC.

DOUKAS SCHOOL SA

ETVA BANK

HEWLETT PACKARD HELLAS SA

INTRASOFT SA

MOTOR OIL

PAPASOTIRIOU BOOKSTORE

SYSCOMP

UNISYS HELLAS

AUTHOR LIST

Abad Ferreira H. (A)	Brazil	s21
Bariamis G.	Greece	163
Batagelj V.	Slovenia	213
Beguin C.	France	77
Bellemain F.	France	223
Blaho A.	Slovakia	s1
Bornholt L.	Australia	3
Bustamante B. V. S.	Brazil	s13
Bustamante B. V. S.	Brazil	s21
Calabrese E.	Italy	85
Calani Baranauskas M.	Brazil	259
Carneiro de Souza M.	Brazil	s13
Chaimantas S.	Greece	163
Chiavarini G.	Italy	85
Close S.	Ireland	11
Coutinho de Souza J. C.	Brazil	s13
Crawford K.	Australia	3
Dagdilelis V.	France	223
Dapontes N.	Greece	97
De Corte E.	Belgium	335
De Marcellus O.	France	77
DiSessa A.	USA	351
Dicheva D.	Ireland	11
Doyle M.	UK	21
Elliott A.	Australia	33
Enkenberg J.	Finland	319
Farkas K.	Hungary	175
Futschek G.	Austria	105
Gow J.	USA	269
Grobelnic M.	Slovenia	233
Gyftodimos G.	Greece	s55, s63
Harvey B.	USA	359
Hilel J.	Canada	s45
Hoyles C.	UK	367
Ide A.	Japan	239
Kalas I.	Slovakia	s1
Kapnia Arapostathi F.	Greece	s27
Kato H.	Japan	239, 297
Kontogiannopoulou G.	Greece	377
Koros-Mikis M.	Hungary	175
Kotsanis Y.	Greece	163, 329
Koukouvinou S.	Greece	41
Koulouris C.	Greece	329
Koutsogiorga I.	Greece	47
Krivy I.	Czech	275
Kynigos C.	Greece	377, s55
Lakotieva A.	Finland	319
Lapidot T.	Israel	113, 181
Levy D.	Israel	181
Makrakis V.	Greece	391
Maritsas D.	Greece	397
Marouka E.	Greece	41, 329
Martins M. C.	Brazil	119
Martins de Pina C. S.	Brazil	s21
Michaelides T.	Greece	127
Minopoulos C.	Greece	127
Mladenic D.	Slovenia	53
Morato P.	Portugal	321
Noss R.	UK	405
Oestermeier U.	Germany	247
Papadopoulou D	Greece	325
Papathomaidi L.	Greece	163
Papavassiliou E.	Greece	391
Patarakin E.	USSR	s29
Patrineli A.	Greece	127
Rachman R.	Israel	113
Ripper A.	Brazil	63
Robinson C.	UK	189
Rossi A.	Italy	85
Rousseau-Gerard M.	Belgium	195
Schauer H.	Austria	105
Sendov B.	Bulgaria	281, 327
Sendova E.	Bulgaria	281
Sereno F.	Portugal	s35
Shafee Give'on Y.	Israel	137
Sklenar J.	Malta	289
Sklenarova K.	Malta	289
Stergiopoulou L.	Greece	s55
Summers F.	Australia	3
Suomala J.	Finland	69
Suzuki H.	Japan	297
Tergan S.	Germany	247
Tomcsanyi P.	Slovakia	s1
Travina L. (A)	USSR	s29
Triantafillou S.	Greece	41, 305, 325, 329
Tryphon A.	France	77
Turcsanyi-Szabo M.	Hungary	145, 201
Vieira da Rocha H.	Brazil	153
Vitale B.	France	77
Weinberger J.	Czech	275
Zupanic D.	Slovenia	311

A. Learning Process

3 • *Bornholt L., Crawford K., Summers F.*
Learning in LogoWriter Environment: a Hierarchy of Students' Conceptions of Tasks

11 • *Dicheva D., Close S.*
Misconceptions and Mental Models of Recursion

21 • *Doyle M.*
Logo: Windmills, Writing, and Right Triangles

33 • *Elliott A.*
Preschoolers' Logo Play: Characteristics Associated with Logo Competence

41 • *Koukouvinou S., Triantafillou S., Marouka E.*
Logo as an Assessment Tool

47 • *Koutsogiorga I.*
Logo, as a Studying Procedure

53 • *Mladenic D.*
Inducing Knowledge from Examples

63 • *Ripper A.*
Logo Environment in Pre-school: a Mediation Tool

69 • *Suomala J.*
Natural Learning in a Lego-Logo Learning Environment

B. Domain Specific Learning

77 • *Beguin C., Vitale B., De Marcellus O., Tryphon A.*
Experimentation et Programmation pour la Construction de l' Espace Sonore

85 • *Calabrese E., Chiavarini G., Rossi A.*
An Introduction to Database with Logo

97 • *Dapontes N.*
La Tortue, le Formalisme Mathematique, et les Connaissances Intuitives ...

105 • *Futschek G., Schauer H.*
Exploring Recursion: from Random Walks to the Design of Mazes

113 • *Lapidot T., Rachman R.*
MicroBridge: a Combinatorical Microworld

119 • *Martins M. C.*
Analysis of the Musical Domain in the Logo Environment

127 • *Minopoulos C., Michaelides T., Patrineli A.*
Discovering Angles with Logo

137 • *Shafee Give'on Y.*
Catch Infinity by its Tail: Infinite Sequences in Logo, Revisited

145 • *Turcsanyi-Szabo M.*
From Square to a Thousand Cranes

153 • *Vieira da Rocha H.*
The Use of Computational Representations in the Teaching and Learning ...

C. The School Interface

163 • *Bariamis G., Chaimantas S., Kotsanis Y., Papathomaidi L.*
A Microworld Oriented Approach in a Multi-Functional Logo-Based Curriculum

175 • *Koros-Mikis M., Farkas K.*
Informatics in Hungarian Public Education - Logo Environments in Primary Schools

181 • *Lapidot T., Levy D.*
From Programming to Computer Science: Opportunities and Pitfalls

189 • *Robinson C.*
Conversations with a Computer: Third Generation of Computer Assisted Learning
195 • *Rousseau-Gerard M.*
Pre-lecture par Logo en 3eme Maternelle
201 • *Turcsanyi-Szabo M.*
Where to Place Logo in Teacher Training

D. Software Design

213 • *Batagelj V.*
Prfkos Authoring System Based on Logo
223 • *Bellemain F., Dagdilelis V.*
La Manipulation Directe Comme Invariant des Micromondes Educatifs ...
233 • *Grobelnic M.*
Design of Logo Command Names
239 • *Kato H., Ide A.*
AlgoArena: a System for Learning Programming through Social Interactions
247 • *Tergan S., Oestermeier U.*
Cognitive Design Principles of a Generative Logo-like Learning Environment

E. Programming Environments

259 • *Calani Baranauskas M. C.*
A Constructionist Approach to the Design of a Logic Programming Environment ...
269 • *Gow J.*
Novice Knowledge Engineering with a Logo-Based Expert System
275 • *Krivy I., Weinberger J.*
Project Management Analyser in Logos
281 • *Sendova E., Sendov B.*
Learning to Speak Mathematically and Speaking Mathematically to Learn ...
289 • *Sklenar J., Sklenarova K.*
Network Paradigm Modelling in the Logos Language
297 • *Suzuki H., Kato H.*
AlgoBlock: a Tangible Programming Language - a Tool for Collaborative Learning
305 • *Triantafillou S.*
Preschoolers Logo World
311 • *Zupanic D.*
Parallel Programming in Logo

Demonstrations

319 • *Enkenberg J., Lakotieva A.*
Modelling-Based Learning in Lego-Logo Environment
321 • *Morato P.*
Logo Environment & the Development of Spatial Cognition
325 • *Papadopoulou D, Triantafillou S.*
Logo for Motor Disabled Children
327 • *Sendov B.*
Geometric Exploration through Visual Modelling
329 • *Triantafillou S., Koulouris C., Kotsanis Y., Marouka H.*
The Development and Testing of a Greek Educational Environment ...

Plenary Addresses

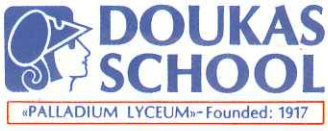
335 • *De Corte E.*
Toward Embedding Enriched Logo-Based Learning Environments in the School ...
351 • *DiSessa A.*
Collaborating via Boxer
359 • *Harvey B.*
Symbolic Programming vs. Software Engineering - Fun vs. Professionalism - ...
367 • *Hoyles C.*
Exploiting Logo-like Environments for Learning Mathematics
377 • *Kontogiannopoulou G., Kynigos C.*
An Educational Perspective of the Socio-Cultural Pre-requisites ...
391 • *Makrakis V., Papavassiliou E.*
Learning Process in an Exploratory Data Handling Environment
397 • *Maritsas D.*
The Integration of the Use of Computer Technology in Greek General Education
405 • *Noss R.*
The Politics of Logo

Sponsors

Author List

Supplement

s1 • *Blaho A., Kalas I., Tomcsanyi P.*
Comenius Logo: Environment for Teachers and Environment for Learners
s13 • *Bustamante B. V. S., Coutinho de Souza J. C., Carneiro de Souza M.*
Homeless Childhood: some Perspectives to Introduce Logo Language ...
s21 • *Bustamante B. V. S., Martins de Pina Cabral S., Abad Ferreira H.*
Strategies of Learning: Reconstructing the Knowledge from a Logo Environment Model
s27 • *Kapnia Arapostathi F.*
Teaching Geometry and Logo through Logo
s29 • *Patarakin E., Travina L.*
Psychological Education - Logo Style and Logo Media
s35 • *Sereno F.*
Fractals and School Mathematics
s45 • *Hillel J.*
Computer Algebra Systems as Mathematical Learning Environments
s55 • *Stergiopoulou L., Kynigos C., Gyftodimos G.*
Une Formation des Enseignants en Didactique Basee sur Logo dans le Systeme Educatif...
s63 • *Gyftodimos G.*
Expressing Ideas through Sound Manipulation with Logo



DOUKAS SCHOOL S.A.

36-40 Messogion St., Maroussi 151 25 - Athens, Greece Tel.: (01) 682.2500 - Fax: (01) 681.8277