22. Programming as a Means of Expressing and Exploring Ideas: Three Case Studies Situated in a Directive Educational System

Chronis Kynigos

Section of Education, Dept. of Philos., Educ. and Psych., School of Philosophy, University of Athens and Computer Technology Institute, Athens, Greece

Abstract. This chapter discusses programmability as an important property of exploratory software for education within a framework of progressive discrimination of which of its aspects are good agents for infusing pupil control over technology and their own learning, the enjoyment of personal construction and the ability to express ideas and generalizations. The ones discussed are programming a) as an agent for developing an alternative teaching and learning paradigm within a directive educational culture, b) enabling pupils to construct objects consisting of a set of ideas, understood in varying depth, but concurrently explorable, and c) providing the potential for exploring ideas from content domains other than mathematics, such as physics. The arguments are based on a description of three Logo learning environments situated in the Greek educational system.

22.1 Introduction: Programming and Directive Education

Many years of research on learning and pedagogy within environments based on computer programming, in one form or another, have generated a fair number of cases where programming has provided pupils with a means of expressing and exploring domain-specific ideas. Researchers are tapping and explicitly describing aspects of learning with which, outside this kind of learning environment, pupils would have little, if any, contact. Programming, in these cases, "obliges" pupils to formalize intuitive ideas by expressing them with the use of symbols, executing them on the computer and immediately observing their effect, comparing it with the one they intended before execution. It enables them to group a set of ideas and then either use them as a named object at a higher level of abstraction or reflect on a specific idea within that object. In this way, they can interchangeably use an idea as a tool and reflect on it as an object (to use Douady's terminology, 1985). Regarding the representation of these ideas and objects, the programming environment is often designed so that the execution of a symbolic expression produces graphical feedback. In these cases, the pupils engage in a rich interplay among representations such as moving along an enactive-iconic-symbolic representational spiral

(Bruner, 1974; Mason, 1987). As Noss and Hoyles (1992) put it "the symbolic representation is amenable to generalisation...objects have measures associated with them that are visible, quantifiable and formalizable. Thus there is the scope for the symbolic form matching the intuitive" (p. 457). Programming thus allows a merging of intuitive and reflective thinking. Furthermore, just as in language, programming allows the pupil to create, extend and enrich a domain-specific vocabulary (see Eisenberg, this volume) and, unlike spoken and written language, it enables pupils to design and construct objects (Harel and Papert, 1990).

Research on the above supports arguments that the ability to program in one form or other will be important in a wider perspective, as a means by which control over the widely available technology of the future will be widespread instead of the privilege of a few (Papert, 1980; diSessa and Abelson, 1986). Moreover, in the field of education, the need to nurture a society of flexible "learners" rather than one of "knowers" of quantities of unchanging information (Soloway, 1990) requires that educators should increasingly emphasize the meaningful use of ideas and concepts to fulfill self-initiated goals. Programming, as a means of expressing, exploring, structuring and abstracting ideas in small-group project work settings, may thus generate rich opportunity for the social construction of meaning in the classroom. At the same time such programming advances the potential for education to play an important part in infusing society with characteristics of the visionary "computer culture" elaborated by Papert (1981).

However, the educational value of programming activity in the classroom setting is controversial. Not so long ago, the only way to do something interesting with a computer was to use one symbolic programming language or another. So, apart from computer-as-textbook approaches, such as C.A.I., I.C.A.I., C.A.L. and I.T.S.[1] whose priorities were unrelated (if not opposed) to pupils having control over the technology, teaching programming was not brought so much into question, notwithstanding endless debates on which language was best for what kind of activity. Technological developments in pupil-computer interaction, such as icon-driven interfaces, visual programming and direct manipulation are rapidly allowing users to do many things with a computer (besides responding to questions and, in general, being "programmed" by one) without having to learn how to use a symbolic code to program. Unfortunately, these developments have been associated with a widespread critique against the educational value of learning to program.

In the wider setting, programming is seen as an activity for professionals who can use a technical symbolic code in order to implement abstract algorithms to make machines do complicated things. Programming by pupils has thus been seen by many as a simplified version of what these professionals do. Simpler code for sim-

[1] These refer to Computer Assisted Instruction, Intelligent Computer Assisted Instruction, Computer Assisted Learning, Intelligent Tutoring Systems, respectively.

pler tasks, but under the same philosophy of what programming (and teaching and learning) is. There is a widespread lack of awareness that, under certain circumstances, programming may provide a language for exploring and expressing ideas belonging to domains other than informatics. Although very relevant to education, such vocational programming is idiosyncratic to computer science specialists (Sinclair and Moon, 1991), who have nonetheless significantly influenced perceptions of the role of computers in the classroom (Kontogiannopoulou-Polydorides and Kynigos, 1993). The design of each programming language has, of course, an underlying philosophy, tightly related to the application in which it will be used. The philosophy underlying languages for learning like LISP (Sinclair and Moon, 1991), Scheme (Abelson and Sussman, 1985), Logo (Papert, 1980; Harvey, 1985; Abelson and diSessa, 1981) and Boxer (diSessa and Abelson, 1986), supports activity where programming (by an experienced programmer or by a novice) goes together with the evolution of the programmer's grasp of the problem at hand (Kynigos *et al.*, 1993). This is in contrast to languages like Pascal and C, which emphasize advance planning and bug-free production. It is a paradox that while the roots of evolutionary programming are very much in the field of computer science, its specific problematic and philosophy have not been associated with education by most computer scientists.

Thus, in a context where the idea of a computer-related subject added onto a curriculum is giving way to that of a widespread use of computers in all subjects (Plomp and Pelgrum, 1992; Kontogiannopoulou-Polydorides, 1992), there is a strong belief that programming activity is at the core of the old-fashioned technical approach to using computers in schools.[2] Concomitantly, there is little awareness that programming may be used for expressing, exploring and generalizing ideas in many subjects besides computer science.

Another source of criticism against the educational value of learning programming is taking early visionary claims regarding computer use in education at face value, impatiently expecting that they would soon materialize more or less autonomously as a social development (Papert, 1981; Apple, 1991). For instance, research in children's learning of mathematics while programming in Logo has, in general, fallen short of what was initially envisaged by its designers, i.e., the spontaneous generation of self-created problem solvers (Papert, 1980). In a recent synthesis of this research, Noss and Hoyles interpret this as a consequence of a) perceiving Logo as a tool for general problem solving, not related to specific content and b) focusing too strongly on the psychological aspects of learning, not taking into

[2] This is where one sees criticisms of any programming language, no matter how different in underlying philosophy, (e.g., Logo and Basic) appearing interchangeably, such as "in the 80s the definition was the ability to program in Basic and Logo, two easy to use programming languages....it was clear by 1990 that programming would become a professional task and that most computer users would never have to write a program in their lives." (Jackson, P., *The Independent*, 3/9, 1993).

account the social context in which it happens (Noss and Hoyles, 1992). It would now seem that the theoretical frameworks for implementing "knowledge construction" teaching and learning environments were implicit and narrow during the 80s, especially with respect to the use of computational applications. (It is telling that there have been very recent advancements in explicitness regarding this issue, such as Cobb *et al.*, 1992; Hoyles and Noss, 1993).

Another factor not adequately taken into account is the restrictions imposed by the available technology before and during the 80s. For example, in the first study of children learning with Logo there were no screens available to them (Papert *et al.*, 1979).

Finally, there has been a general lack of evaluation theories and techniques. In a critique of technocentric approaches to the problem, Papert himself proposed that prior research perspectives and conventional research methodologies were narrow and inadequate; when the attempt is to generate an exploratory educational environment, where everything regarding the teaching and learning process is different, short-term psychometric control-and-experiment-group methodology measures very little of what is actually happening (Papert, 1987).

Programming has thus been criticized for being difficult and time-consuming, for requiring a substantial overhead of redundant technical knowledge, for having failed to deliver the promises of generating general-purpose problem solving skills transferable among subject domains, and it has been subject to the most pointed criticism when conventional evaluating methods were used.

On top of all this, the consummatory fervor promoted by the commercial world of technological applications enhances the inherent reluctance of educational systems to accept and support real innovative practice, like reflective and investigative thinking, collaborative working and qualitative pupil evaluation (Noss, 1992). Instead, the commercial world fosters the image that the importance in technological development rests on computers becoming easier to use or on learning about the new features in an endless production of new versions of computer applications. (See Eisenberg, this volume.)

Within this broad context, we discuss programmability as an important property of exploratory software for education within a framework of progressive discrimination of which of its aspects are good agents for infusing pupil control over technology and their own learning, the enjoyment of personal construction and the ability to express ideas and generalizations. We describe five episodes from three Logo learning environments. Two episodes in the first environment were set in a school context where programming was perceived explicitly as an agent for developing an alternative teaching and learning paradigm within a directive educational culture. Programming as a means to explore geometrical and physics ideas is subsequently highlighted in three decidedly more positive learning episodes where a researcher acts as teacher with one group of pupils at a time outside the school context.

The common factor among these environments is that they were set in Greece, where the prevailing educational paradigm is characterized by abstract and disembedded information transmission, which is directively implemented through a centralized educational system (Kontogiannopoulou-Polydorides and Kynigos, 1993). The first environment was specifically set up in order to challenge this educational paradigm in a school setting. It was set within a longitudinal project, involving all teachers and children of a primary school that is using Logo programming to develop alternative pedagogy. Two episodes from this environment are described, the first regarding the didactical intervention of a teacher with seven years prior Logo teaching experience within the project. This teacher's strategies to influence the learning environment, both to discourage an unreflective use of Logo, and to help the children to become aware of the interesting and powerful ideas that they use in their projects were challenged by the assumptions carried over from the wider educational paradigm. The second episode is taken from a case study of 11 year-old children with close to three years Logo experience attempting to inject structure in their procedures. It highlights the local nature of pupils' abstractions and the difficulty of timely teacher intervention in a real classroom setting. The second learning environment involves a pair of 11 year-olds from the same school, but this time working with a researcher as participant-observer out of the classroom context and using a geometrical microworld designed to highlight intrinsic and plane geometrical ideas. The potential for programming simultaneously to allow and scaffold measurement, conjecture, generalization and abstraction is discussed. The third environment concerns learning to program the dynaturtle (diSessa, 1982) with time primitives. It draws on the analysis of two episodes, one involving 12 year-olds from the above school, and one involving physics graduates training to be secondary teachers. The importance of different aspects of programming in relation to the respective subject domain is discussed.

All the above form a picture of the problematic of exploratory programming in a directive educational system. On the one hand, the generation of such learning environments is obstructed by the cultural assumptions about education and by the difficulty for pupils to progress above a certain plateau in their understandings and investigational techniques; on the other, in settings less constrained by the above, programming uniquely allows concurrent exploration of a set of ideas understood at varying depths in a content domain much wider than that of computer science.

22.2 Programming to Infuse Investigative Learning in an Adverse Culture

Within the centralized and didactically oriented Greek educational system, collaborative investigational project work and encouragement for the construction of meaning is by no means automatically understood, practiced or favored. The pedagogy 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 whether it was achieved only by rote or not. Challenge to this framework of directives and perspectives on teaching and learning can be found mainly informally and at the microlevel of specific teachers and/or schools who are working against the run of the mill. Not surprisingly, perceptions of computer use in Greek education and related policies are technocentric, with little relation to educational priorities and development ("Astrolavos" report, 1992; Kontogiannopoulou-Polidorides and Kynigos, 1993).

The two classroom episodes elaborated in this section recently took place within an atypical primary school (Psychico College) Logo project where, in contrast to the wider Greek setting, the technology was used to qualitatively reform and develop learning and teaching processes, rather than to quantitatively optimize existing ones. The first episode is about the didactical intervention of a teacher with seven years prior Logo teaching experience within the project. The second episode is about how programming was perceived and used in a typical project of a group of three 11 year-olds with 2.5 years of Logo experience.

The school project, which began in September 1986 and has been based on classroom activity from the outset,[3] involves teacher training, curriculum development and research on teacher strategies and children's learning. From year 3 to 6 inclusive (i.e., children aged 8 to 12), all 24 teachers of the school take part, each one responsible for the participation of all the children in his/her class (500 children in total). Throughout a four year period, the children engage in informal collaborative investigational work for one teaching period a week and compose a written presentation on each of their investigations, which typically lasts 5 to 6 weeks. Details of the project's outline, educational objectives, working structure, classroom setup and "taught" content can be found in Kynigos, 1992b. Studies involving children's learning process can be found in Kynigos, 1991, 1992, 1993. A study of children's use of programming ideas is described in Kynigos *et al.*, (1993).

[3] School projects in Greece are few and usually restricted to an intensive teacher training course outside classroom activity and with no follow-up related to it. Alternatively, the researchers take over classroom teaching without the presence of the teachers. There is great resistance to "outsiders" participating or influencing normal classroom activity.

In this context, technology has been used to set up an unconventional classroom practice; all the teachers have been developing strategies for a pedagogy encouraging collaborative investigations centered around Logo programming for one hour a week. Pupils work in groups of two or three; they are encouraged to discuss ideas, to negotiate their mutual cooperation, to persist on a problem and deal with it in depth, to develop autonomy and responsibility towards themselves and their group peers, to 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 is in the form of informal encouragement to do better, rather than a final judgment on past and irreparable performance.

The first episode describes the attempt of a teacher to skew a group's perceptual "drawing" with the Logo turtle to a mathematical investigation of how to construct a circle, without disengaging them from their own project goals. The description is based on video recordings of the teacher's activity and transcription of all her interventions throughout the five sessions of this investigation. The three interventions described below happened during the two final periods of the second investigation of a third year class.

A group finished their "rocket" project, mainly using perceptual cues to direct-drive the turtle to construct it. The only indication of an implicit use of analytical thinking is in the symmetrical and equally-sized lines for the rocket's tail. The teacher asks them to initiate further work, since they have another hour and a half to complete the investigation. When they suggest the idea of a planet, she encourages them and asks how they can make a circle. The pupils first say they don't know how to make a circle and will thus make a square planet. When encouraged to think how they might make a circle, they decide to try moving the turtle a bit and turning it a bit many times. They type in moves and turns alternately, but with no pattern to the input quantities. The teacher does not intervene for the rest of that hour.

The following week, during the first 15 minutes or so, the pupils continue to type inputs to alternative turns and moves. Among these, there is a sequence of equal inputs to the turn commands. The teacher then intervenes, nudging the pupils to look for a pattern.

Another 15 minutes or so gone by, she asks where they're at and discovers they had not reached the desired conclusion, i.e., constant turns and moves. They were still changing inputs. She accepts this activity but asks whether they can predict what shape will come out. They say, "It won't 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 the result of the sequence of commands with equal inputs looks more like a circle than that of the other commands, she concludes that they should rethink so that it would come *out more like a circle. She asks them to compare the two results. She 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." In the end, the pupils announce the intended observation. But they do not change their figure as a result, nor do they write anything about the planet in their written essay!*

This is an obvious attempt by the teacher to encourage problem-solving activity and some autonomy on the part of the pupils. However, one has the distinct feeling that she is in an uphill battle. At the beginning she attempts to offer a chance at investigating an interesting idea stemming from the pupils' own goal. Their initial reaction was opposite: Avoid the idea, even if the result is not aesthetically satisfactory, so that we do not have to delve into the unknown! In the second intervention, she observes that they have implicitly caught on to the idea of alternating moves and turns and thus tries to point to the next problem, of equal quantities. The pupils simply ignored the teacher's cue, failing to reflect on previous actions or predict future ones. In the final intervention (15 minutes before the end of the investigation), she becomes more heavy-handed, explicitly suggesting they should reflect and predict, even pointing to a specific sequence of commands and the resulting computer feedback. The pupils provide the answer (it had become almost obvious by then), but do nothing about using it in their construction, nor show that they found enough interest or importance in these incidents to write about them in their essay. Instead, they describe how they cooperated and what their rocket looks like.

The teacher has evidently developed strategies to influence the learning environment, both to discourage an unreflective use of Logo (Leron, 1985), and also to help the children to focus on the interesting and powerful ideas that they use in their projects. However, her efforts are not facilitated by the assumptions carried over from the wider educational paradigm to the Logo work. This is true even though during the normal curriculum hours she attempts to engage the children in classroom discussions and find time for them to work in small groups, and the school explicitly supports such pedagogy. The children come into the Logo classroom with the expectation that the teacher is there either to provide information or answers, or to test whether the pupil can provide them. The teachers have, in general, gradually developed a meaningful way to communicate to the children why they are not readily giving them "answers," why they will often throw the responsibility of a situation back to the children themselves, and that it is socially acceptable and legitimate for them to conjecture, theorize and make mistakes. Even so, this does not seem to be a case of simply explaining the rules of a new game and then playing it. The teachers understand these are long-term cultural changes and see important educational value in consistently communicating these rules throughout the four years duration. An added difficulty is the conditions under which this is attempted. This is only an adjunct "investigations" hour (as if all other topics have no relation to this) and there are only 50 minutes of one teacher with a class of thirty (a proficient teacher will on average make two two-minute interventions in the work of a group during a teaching period) (see also Budin, 1991 for a related discussion).

The second episode involves students constructing windmills in Logo, and is presented as an extract from an investigation by a group of fifth year pupils, who are well into their third year of Logo project work. It was chosen as a means to elaborate characteristics of many more pupils' work, from among around 500 such presentations produced in the school each year and used as a principal research instrument. In Figure 1, the explanatory titles given to their procedures are their own, and the order in which they present them is from left to right. In order to focus on the issue at hand, which is the pupils' strategy for procedure building, the word "primitives" is substituted for a sequence of primitive commands.

The example indicates that the pupils' higher-order procedure building was heavily influenced by (if not based upon) the criteria of time and sequence and related to perceptual drawing cues (Hillel and Kieran, 1987). For instance, `E2` was defined and used as a first order abstraction (as implied by the pupils' choice of title and procedure sequence, `E2` signifies "the Wall of the first mill" rather than a sequence of commands), and was subsequently used in a second order abstraction, as a subprocedure to `E3`. It seems likely the pupils perceived `E2` as an object since they used it again later in `E5`. Their failure to distinguish interfaces from objects in the construction and to clearly perceive the construction of the three walls as three executions of `E2` plus interface, instead of three separate procedures, corroborates other research in children's conceptualization of procedure definition (Noss, 1985; Hoyles and Sutherland, 1989). Furthermore, it does not seem that `E4` was defined in order to express a higher-order symbolic formalization of a conceptually or perceptually defined object; if that were the case, the pupils would more likely have either incorporated all three procedures for the Walls in the definition of `E4`, or left them out, since it is obvious that the pupils saw the relation between the three by using subprocedure `E2` in all cases. Instead, `E4` consists of: an interface (`E1`), a wall (`E2`), and a composite interface-and-wall procedure (`E3`). That makes only two interfaces and two walls; the third interface and wall appear in the subsequent procedure (`E5`). The same failure to perceive superprocedure construction as a means to express higher-order objects is evident in the time at which they decided to make their next higher-order procedure, `E8`; they defined `E8` in the midst of the construction of the three roofs, instead of perceiving it as a composite "wall-and-roof" procedure. Their strategy for defining composite procedures (`E4, E8, E16, E19, E22`) is inconsistent in yet another way: `E8`, `E19` and `E22` consist of the previous higher-order procedure and a sequence of the procedures defined since then. In contrast, `E16` is a linear sequencing of all the previously defined procedures except the higher-order ones `E4` and `E8`. That means that out of the list of composite procedures, `E4` and `E8` are not used in subsequent superprocedure building. In most cases they define a procedure for a distinct part of the design, yet, in some, that is not so clear (e.g., `E10, E12`). Although their design invites the use of mathematical ideas (triangles for wings and roofs, orthogonals for walls and windows) they seem to use very few, if any (even the primitives for the wing were decided perceptually, incorporating homing-in (Noss, 1985) sequences of commands). Finally, the overall strategy for their design seems perceptual and sequential; they began by constructing the windmill walls from left to right, then the roofs from right to left, then the wings of the mill at the left and then, from left to right, putting in the finishing touches.

Turtle move	Wall of first Mill	**Move and wall of second Mill**	**Tidying up of commands**
TO E1	TO E2	**TO E3**	**TO E4**
primitives	primitives	**primitives E2**	**E1 E2 E3**
END	END	**END**	**END**

Move and wall of third Mill	Roof of third Mill	Roof of second Mill
TO E5	TO E6	TO E7
primitives E2	primitives	primitives E6
END	END	END

Tidying up of commands	Roof of first Mill	Turtle turn	Move
TO E8	TO E9	TO E10	TO E12
E4 E5 E6 E7	primitives E6	LT 90	primitives
END	END	END	END

Mill wing	**Mill wings**	**Tidying up of commands**
TO E14	**TO E15**	**TO E16**
primititves	**REPEAT 9 [E14]**	**E1 E2 E3 E5 E6 E7 E9 E10 E12 E13 E14 E15**
END	**END**	**END**

Door	Sideways Mill	**Tidying up of commands**
TO E17	TO E18	**TO E19**
primitives	primitives	**E16 E17 E18**
END	END	**END**

Window of second Mill	Window of third Mill	**Tidying up of commands**
TO E20	TO E21	**TO E22**
primitives	primitives	**E19 E20 E21**
END	END	**END**

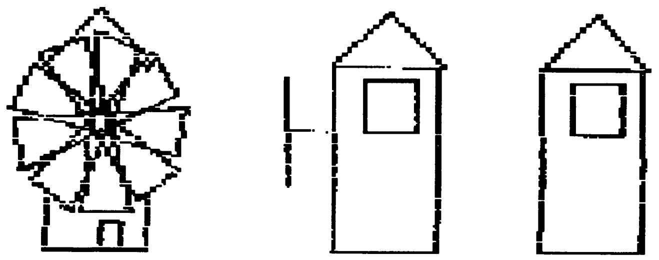


Figure 1. Superprocedure construction by a group of fifth-year pupils.

The above two episodes took place in a setting where programming was explicitly (and relatively successfully) used by the teachers to develop an alternative teaching and learning paradigm within a directive educational culture. However, a close look at the pupils' activity reveals that realizing the learning potential described in the first paragraph of this chapter leaves much to be desired, even though in many cases, such as in the first episode, the teachers' interventions indicate consistent pedagogical strategy to that effect, stemming from their seven years experience. The pupils use a symbolic code to cause a direct effect on the screen. They do not seem to convey precise meanings through procedure definitions, have little awareness of this when challenged and do not easily see the point of it upon teacher intervention. At a more experienced stage, such as in the second episode, they seem to define procedures with ease (and even ones with an impressive number of embedded levels of subprocedures e.g., `E22` has five such levels), yet show a local, inconsistent and fragmented perception and use of these as objects of a higher level of abstraction. They avoid using the mathematical ideas present and maintain a perceptual and sequential mode of action.

It may be true that the teachers themselves have some ground to cover in discriminating the relevant mathematical and programming ideas and learning more about intervention technique. We suggest, however, that the sociocultural obstacles to developing this kind of pedagogy have important bearing on what happens in the classroom and thus need very serious attention. In the following two sections, where the learning environments were set in a "laboratory" situation and the researcher-teacher provided dense, explicit and consistent feedback regarding the roles and expectations among the parties concerned (the above school teacher had to also spend many hours during the day giving a different set of signals to the same pupils), pupils of slightly above average performance, similar to the two groups described in this section, improved in their investigational techniques and attitudes to their work. In a society where consumerism is largely an end in itself, where centralized educational systems depend on measurable performances and are narrowly result-and-time oriented, it is difficult, within the classroom context, even to argue for the value of exploration, expressive wealth and precision, focusing on the learning process, working collaboratively and acquiring autonomy in decision making. After seven years, the teachers believe that active thinking, autonomy and collaboration are still at the core of what they teach with Logo programming and that the associated skills still remain a challenge for the pupils after primary school.

The preceding work shows difficulties and limitations in attempting to change in the midst of an encompassing antithetical educational system. In the following sections, where there is a lot less of this kind of constraint on the learning environments, the promise and possibilities for success of exploratory programming emerge much more clearly.

22.3 Programming as Scaffolding for Situated Generalizations: An Intrinsic Plane Geometry Microworld

The following episode is used to discuss the range of roles programming can play in pupils' use and understanding of ideas (for a discussion on the choice of the order of "use" and "understanding" see Hoyles and Noss, 1987). It took place during a case study of two sixth-year pupils working with a participant-researcher using a geometrical microworld. The description of the episode is based on a qualitative analysis of all that was said, typed and written during the research sessions (Kynigos, 1993).

The two pupils had some prior experience in exploratory project work with Logo programming within the framework of the above school project. In the research sessions they were given a set of tasks regarding the construction of isosceles triangles by means of a special microworld (Loethe, 1985) enabling them to mark points on the plane and name them (primitive POST) and measure distances (primitive DISTANCE) and rotational quantities (primitive DIRECTION) between the Logo turtle's current state and any marked point. The children began by using these tools—simulating a point maker, a ruler and a compass—to construct isosceles triangles without using any properties of triangles. Since they had not been taught any such properties in their geometry lesson, they began simply by measuring how much to rotate the turtle and then moving it to close the figure. By measuring other elements of these triangles they began to notice equalities, to make conjectures on possible properties and to test these out by repeatedly changing quantities within fixed isosceles triangle procedures. Midway through the project, they attempted the task of writing a procedure for an isosceles triangle with a variable input for the length of one of the equal sides and another for one of the equal angles. This procedure, which they called LASER (Figure 2), is at issue here, since it is the result of having expressed mathematical ideas at varying levels of abstraction.

```
TO LASER :N :P
   POST "I
   RT 90 - :P
   FD :N
   POST "H
   RT 180 - (180 - (2 *:P))
   FD :N
   POST "M
   RT DIRECTION :I
   FD DISTANCE :I
   RT 90
END
```

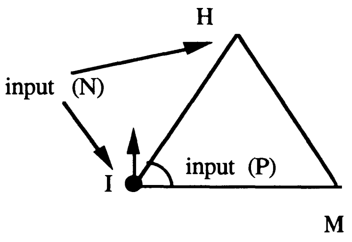


Figure 2. A generalized isosceles triangle procedure by two sixth-year pupils.

The commands that comprise the `LASER` procedure indicate the use of mathematical ideas at varying levels of abstraction. The two commands in boldface (**`RT 90-:P`** and **`RT 180-(180-(2 * :P))`**) embody the expression of generalized angle properties; they involve operations on the variable `input :P`, which signifies the size of the equal angles, in order to relate these to the appropriate turtle rotations at points I and H. The commands in normal typeface (`FD:N` and `RT 90`) signify the simple use of triangle properties in order to provide the respective quantities to change the turtle's state. In contrast, the two commands in italics (*`RT DIRECTION :I`* and *`FD DISTANCE:I`*) indicate that the children did not use geometrical properties of the figure to turn the turtle at point M and move it to construct the third side of the triangle MI. Rather, they used the measuring instruments as "scaffolding" (Hoyles and Noss, 1991; Noss and Hoyles, 1992). In prior work, they had used the same scaffolding to construct a procedure, run it with varying inputs and later conjecture on the respective properties that would enable replacing the measuring primitives. That prior work led up to the capability of these pupils to use geometric properties rather than measurement scaffolding at the other two vertices, I and H. It is interesting that they used measurement to determine the turn at M, since their use of properties at I and H suggests that they were perfectly capable of doing so in the case of M. The point, however, is that the microworld's programmability enabled the construction of an object like the `LASER` procedure that mirrored the pupils' current state of understanding geometrical properties and at the same time lent itself to further use in developing such understanding, both by observing the effect of running the procedure and by consequently changing its contents.

22.4 Domain Specificity: A Microworld for Programming the Dynaturtle

The discussion in this section focuses on the importance of different aspects of programming in a different subject domain. It is centered on diSessa's Newtonian dynaturtle microworld (diSessa, 1982) adjusted so that motions and forces can be programmed in a Logo-like style. Contrasting Logo-experienced 12 year-olds from the school project described in Section 2 and physics graduates working with the microworld (Georgiadis and Kynigos, 1993) provides some insight into how subject-domain specificities may create or highlight problems with different aspects of programming activity. According to Sherin *et al.,* (1993), programming is a good agent for expressing causal relations and time development. However, prior microworld environments (as in White, 1993; Sherin *et al.*, 1993) have not drawn much attention to the issue of being able to insert or change time parameters explicitly within a computational object such as a procedure.

The microworld in question is designed to embed ideas from both structured programming and Newtonian physics, so that exploration by students may bring about understandings from both domains and, at the same time, allow the development of methods and techniques of one domain to support the understanding of ideas belonging to the other. The programming aspect of the microworld is based on the Logo language (Harvey, 1985) and the physics aspect on diSessa's idea of a turtle driven by Newtonian primitives (diSessa, 1983, 1989). For a related discussion of the interplay of computer science and other domain-specific ideas in a microworld see Roschelle and Mason (this volume).

The microworld is based on the metaphor of a turtle driven by primitives a) changing its direction (`LT <input>, RT <input>`), b) applying impulse or constant force in the current direction (`KICK <input>, PUSH <input>`), c) allowing the turtle to continue moving under its current influences for a given time (`GO.ON <input>`) and d) "freezing" the current situation without losing the turtle's developed momentum (`WAIT` and `GO`).

22.4.1 A Conflict of Metaphors and the Meaning of Momentum

Despite the limitations and difficulties discussed in the second section, the children's prior extensive three year experience with small-group and child-directed project work in their classroom provided a critical background for their use of software designed for this kind of learning context. However, the fact that they continually employed the turtle metaphor of a *geometrical* entity did come into an all-important conflict with the properties of a turtle simulating a Newtonian particle. At the center of this conflict was the notion of momentum, influenced by both their understanding of the notion itself and by the way it was portrayed by the microworld.

Not surprisingly (diSessa, 1982), the children's invoking of an "Aristotelian" interpretation of the turtle's behavior was pervasive to begin with, and not easily challenged beyond shallow patches to their understanding in specific situations of discrepancy between their expectations and the actual computer feedback. The classic conflict arose from a sequence of commands such as `KICK 50 RT 90 KICK 50` (Figure 3) and the unexpected non-perpendicular change of the direction of the turtle's motion. This was soon followed by a strategy of annulling the turtle's momentum in order to cause such a change. For example, one of their first projects was to "draw a right angle," a task suited much more to geometry than physics. This, the children did by "reversing" an impulse given to the turtle after the appropriate time period, using command sequences such as: `KICK 50 GO.ON 20 RT 180 KICK 50 LT 90` in order to construct one side of the Figure.

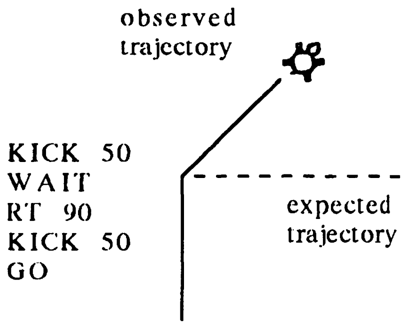


Figure 3. Sixth-year pupil's attempt to "draw a right angle."

Metaphors are designed as a means for children to be able to use a cognitive construct that they understand very well in order to build intuitions regarding the targeted learning. In this microworld, the metaphor of a live frictionful animal was problematic in conveying the meaning of a frictionless and tiny Newtonian particle. However, physics as a learning content area perhaps lends itself more than mathematics or even chemistry to devising metaphors directly associated with children's real world experiences (Noss and Hoyles, 1992).

Investigations with different microworlds may well give rise to conflicting experiences: How does extensive experience with a specific metaphor (e.g., geometrical turtle) influence learning with a microworld based on another, especially if the microworlds are based on the same ideas of programmability? An interesting activity with a geometrical turtle is to construct geometrical figures. For a Newtonian turtle, it is a different kind of activity altogether, involving the process of "getting there," involving time and changes in momentum. In the first case, a recursive tree is of interest. In the second, a specific motion pattern in time (e.g., a waltzing turtle or one which follows a spiral motion, as in Figure 4). Children with over three years of experience with a geometrical turtle started programming with the Newtonian one by choosing projects such as "drawing" a rectangle.

A metaphor expressed via a computational object needs to mobilize the learner's knowledge of the content domain and to express his or her understandings within this domain. Hoyles and Noss (1992) propose the term "evocative computational objects," those that matter within the content domain and that matter to the learner. Designing Newtonian evocative computational objects poses interesting problems. For instance, how is it possible to simulate motion and momentum accurately and at the same time allow for the ability to intervene at any instant and change parameters? The first problem requires a focus on processes of change in time, unlike in the case of geometry. Intervention would thus mean interrupting a physical phenomenon without causing other alterations of what is happening—something like what happens when the pause button is pressed on a video. In fact, this is the metaphor understood by these particular children when they had difficulty in understanding the effect of the command WAIT. At the heart of the matter is the issue of feedback, which in a Newtonian microworld is either continuous, or after intervals of time (with the help of the GO.ON command). State transparency in physics should mean motion transparency, or better still, momentum transparency. If in direct drive programming such issues are interesting, it is even more the case with procedural programming, as elaborated in the following section.

22.4.2 Structured Programming and Recursion to Simulate Velocity Composition

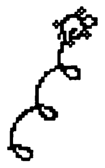
This final episode was situated in a 300-hour course on the use of computer technology in physics education, given to pre-service physics graduates with little or no prior computing and teaching experience. We analyze a programming solution to a problem posed by the students themselves after considerable experience with Logo project work during the course, that of the trajectory of an electrical particle entering a magnetic field with a certain velocity in a hypothetical two-dimensional space. The problem of explaining the particle's motion was broken down into two subproblems: that of a constant velocity and that of a constant circular motion. Each was dealt with by means of a procedure, thus decomposing the particle's behavior into that due to its momentum prior to its entry into the magnetic field and that due to the field's influence on the particle. Figure 4 shows the final procedures, which were the product of the students' investigation. TOXO causes the turtle to move along the arc of a circle (to be more precise, a polygon approximation) of radius R by means of recursively implementing the appropriate centrifugal force (see inputs to RT, KICK, LT) at appropriate time intervals (see input to GO.ON). Once the problem of constant circular motion is solved (SPIRA3), the remaining task is to compose that with a constant linear motion (inputs to KICK and SETH in SPIRA4). What we get is the turtle behaving like an electrically charged particle entering a magnetic field, observed for a small time period (so that a constant linear motion would be a good approximation of a constantly changing linear motion). The variable :ANGLE1 stands for the direction of the particle's velocity upon entry to the magnetic field and is also shown by the direction of the spiral trajectory of the turtle, while :ANGLE2 is related to the direction of the field for which the students did not construct a symbol. Examples of two invocations of SPIRA4 are shown in Figure 4.

```
TO TOXO :U :R :F
GO.ON (100*PI*:R*:F)/(180*:U)
RT 90 + :F / 2
KICK (SQRT(2*(1-(COS:F))))*:U
LT 90-:F / 2
TOXO :U :R :F
END
```

```
TO SPIRA3 :U :R
LOCAL "F
MAKE "F 3
KICK :U
TOXO :U :R :F
END
```

```
TO SPIRA4 :ANGLE1 :ANGLE2 :U :R :UM
START
SETH :ANGLE1
KICK :UM
SETH :ANGLE2
SPIRA3 :U :R
END
```

```
SPIRA4 30 90 20 5 10
```



```
SPIRA4 180 30 30 6 10
```

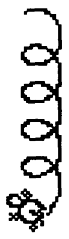


Figure 4. Physics graduates' simulation of an electrically charged particle entering a magnetic field.

The programming environment encouraged students to construct generalized computational tools to simulate a physical phenomenon and to perceive the physical aspect of the problem in a recursive way. Furthermore, the problem is characterized by its modular structure as expressed in programming; the procedure TOXO stands as an independent object as well as part of a composition of velocities. The students' investigation thus provides evidence of the coexistence of programming and physics ideas in an investigation with the software. Moreover, they used programming strategy to conceptualize the physics problem: by means of perceiving constant circular motion as the recursive element of a procedure; by using a procedure simulating a physical phenomenon as a module in a more complex one; and by providing simulations with parameters lending themselves to experimentation through the use of variables. The two episodes in this section support the argument that certain programming ideas may provide a means for expressing formalizations of physical phenomena in the context of their simulation on the screen, once the metaphors of physical entities and the microworld primitives evoke the construction of meanings within the learning environments.

22.5 Conclusions

The nature of the problem of building an exploratory culture in education does not seem to be one where a technical fix, such as equipping classrooms with programmable software and providing precise pedagogical directions for teachers suffices for its solution. The learning potential of programming and the development of pedagogical strategy to encourage it has been elaborated in this and several other research settings, as has the problematic regarding the sociocultural, cognitive, domain specific and application design aspects of realizing such potential. To develop exploratory learning contexts, especially in wider than experimental group settings, all these aspects must be taken into account in order to design an application, a microworld environment, or a pedagogical strategy. In cases such as Greece, the sociocultural aspect seems all important since exploratory learning is rather a social nonentity. In Psychico College, the forming of a culture accepting and nurturing the development of investigational work is happening slowly, without overtly threatening conventional assumptions regarding education. In this framework, the teachers are finding the learning of programming to be useful in communicating the value of active thinking, autonomy in decision making and collaborative work to their pupils. So, in an environment where exploration focused on content has still not matured, programming is used by the teachers mainly as an agent to infuse such an exploratory culture.

Microworld environments, where the primitive vocabulary and the pedagogical agenda for investigative work is more focused, need to be cultivated in settings where the important features of exploratory learning are understood and valued. If not, it is easy for those environments to be used in a very different way, as tools to solve preset exercises with given procedures. (Greece is not alone. Such examples regarding Logo are evident in the U.K. National Curriculum; Hoyles, 1992). In the geometrical and physics microworld environments described in this chapter, pupils found the interplay between set tasks and personal projects natural due to the relatively rich negotiation with the researcher and the "laboratory" setup away from the classroom context.

Within such a setting, the range of roles programming can play in expressing and exploring domain-specific ideas is wide. During their activities with isosceles triangles, the two children used the computer to formalize their intuitive ideas regarding turtle state-changes, tried out perceptual approximations, carried out measurements, and by conjecture, inductively formed understandings of generalized geometrical properties. They later used them in projects of their own (see also

Kynigos, 1993). Programmability of the software allowed them to disengage from necessarily following a sequential and artificially defined path regarding the cognitive demand posed by their activities. Instead, they were able to define objects reflecting their current state of understanding of each part of a problem. They used these objects to reflect on and improve their understanding. In this case, they were, of course, influenced by the researcher's interventions. However, they did seem to come to appreciate and use more and more on their own accord the mathematical ideas within the environment.

Programming for exploration in a range of domains generates design problems specific to each domain. Another issue explored here was the ability of learners to meaningfully explore computational objects of a different nature by means of the same (or similar) programming language. One problem is how to achieve some resonance between programming techniques and concepts within the domain, such as for example, a recursive conception of constant circular motion or a modular program to express velocity composition. Another problem is the design of metaphors enabling stimulation based on intuitive understandings as an entry into exploration; children experienced in programming with a geometrical turtle found it difficult to "switch" to their physics intuitions, attempting to "draw" a right angle with the primitives of a Newtonian turtle. Finally, it takes time for activity to move focus away from language and center on the meaning conveyed by a procedure or computational tool. To make this time available, exploratory activity through programming must become more widely accepted and practiced in society.

Acknowledgments

The Psychico College Project for the Educational use of Computers has been funded by the C. and M. Maliotis Foundation and by the University of Athens, Departments of Informatics and Philosophy, Education and Psychology, via the General Secretariat for Research and Technology, PENED. The school's direction, staff and pupils are acknowledged for their participation and support.

References

Abelson, H., diSessa, A. (1981) *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*, Cambridge, MA: MIT Press

Abelson, H., Sussmann, J. (1985) *Structure and Interpretation of Computer Programs*, Cambridge, MA: MIT Press

Apple, M. W. (1991) The new technology: Is it part of the solution or part of the problem, in *Computers in the Schools*, 8 (1/2/3), 59-81

"Astrolavos" (1992) A program to utilize computing technologies to support education in the Hellenic secondary schools, Proposal to the Ministry of Education, under the supervision of Prof. D. Maritsas, Dir. Computer Technology Institute

Bruner, J. (1974) *Beyond the Information Given*, London: Allen and Unwin

Budin, H.R. (1991) Technology and the teacher's role, *Computers in the Schools*, 8 (1/2/3), 15-26

Cobb, P., Yackel, E., Wood, T. (1992) Interaction and learning in mathematics classroom situations, *Educational Studies in Mathematics*, 23, 99-122

diSessa, A. (1989) A child's science of motion: Overview and first results, *Proceedings of the Fourth International Conference for Logo and Mathematics Education*, U. Leron and N. Krumholtz (eds.), The Israeli Logo Centre, Department of Education in Science and Technology, Technion-Israel Institute of Technology, 211-231

diSessa, A., Abelson, H. (1986) BOXER: A reconstructible computational medium, *Communications of the ACM*, 29/9, 859-868

diSessa, A. (1983) Phenomenology and the evolution of intuition, in *Mental Models*, D. Gentner and A. Stevens (eds.), 15-33, Hillsdale, NJ: Lawrence Erlbaum

diSessa, A. (1982) Unlearning Aristotelian physics: A study of knowledge-based learning, *Cognitive Science*, 6, 37-75

Douady, R. (1985) The interplay between the different settings, tool-object dialectic in the extension of mathematical ability, *Proceedings of the Ninth International Conference for the Psychology of Mathematics Education*, 2, 33-52, Montreal

Harvey, B. (1985) *Computer Science Logo Style*, vols. 1, 2 and 3, Cambridge, MA: MIT Press

Georgiadis, P. and Kynigos, C. (1993) Designing a microworld for learning Newtonian physics through structured programming, *Proceedings of the IFIP Conference: Informatics and Changes in Learning*, A. Knierzinger and M. Moser (eds.) Austria, Thema F, Session 4/2, 9-12

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*, 8/12, 61-103

Hoyles, C. (1992) Computer-based microworlds: A radical vision or a Trojan mouse? *Proceedings of the Seventh International Congress on Mathematics Education*, University of Laval, Quebec, Canada

Hoyles, C. and Noss, R. (1987) Children working in a structured Logo environment: From doing to understanding, *Recherches en Didactiques des Mathematiques*, 8/12, 131-174

Hoyles, C. and Noss, R. (1991) Deconstructing microworlds, in D.L. Ferguson (ed.) *Advanced Educational Technologies for Mathematics and Science*, NATO ASI Series F, Vol. 107, Berlin: Springer Verlag

Hoyles, C. and Noss, R. (1992) A pedagogy for mathematical microworlds, *Educational Studies in Mathematics*, 23, 31-57

Hoyles, C. and Sutherland, R. (1989) *Logo Mathematics in the Classroom*, London: Routledge

Hoyles, C., Healy, L., Pozzi, S. (1992) Interdependence and autonomy: Aspects of groupwork with computers, in H. Mandel, E. DeCorte, S.N. Bennett, and H.F. Friedrich (eds.), *Learning and Instruction, European Research in International Context*, 2, 239-257

Kynigos, C. (1992) The turtle metaphor as a tool for children doing geometry, in C. Hoyles, R. Noss (eds.) *Learning Logo and Mathematics*, 97-126, Cambridge, MA: MIT Press

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 J.P. Ponte et al. (eds.), *Mathematical Problem Solving and New Information Technologies: Research in Contexts of Practice*, NATO ASI Series F, Vol. 89, 219-238, Berlin: Springer-Verlag

Kynigos, C. (1991) Centration sur le processus avant le contenu: Peut-on pratiquer Logo dans une ecole primaire en Grece?, in *Logo et Apprentssages*, 93-104, J.L. Gurtner, and J. Retschitzki (eds.), Geneve: Delachaux et Niestlé

Kynigos, C. (1993) Children's inductive thinking during intrinsic and Euclidean geometrical activities in a computer programming environment, *Educational Studies in Mathematics*, 24, 177-197

Kynigos, C., Gyftodimos, G., Georgiadis, P. (1993) Empowering a society of future users of information technology: A longitudinal study of application in early education, *European Journal of Information Systems*, 2/2, 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, 77-93

Kontogiannopoulou-Polydorides, G. and Kynigos, C. (1993) An educational perspective of the socio-cultural prerequisites for Logo-like education in Greece, *Proceedings of the Fourth European Logo Conference*, C. Kynigos *et al.* (eds.), Doukas School Publication, 377-389

Leron, U. (1985) Logo today: Vision and reality, *The Computing Teacher*, 12, 26-32

Loethe, H. (1985) Geometrical problems for a turtle with direction and distance finder, *Proceedings of the First Logo and Mathematics Education Conference*, 123-129

Mason, J. (1987) What do symbols represent?, in C. Janvier (ed.), *Problems of Representation in the Teaching and Learning of Mathematics*, Hillsdale, NJ: Lawrence Erlbaum Associates

Noss, R. (1985) Creating a mathematical environment through programming: A study of young children learning Logo, Doctoral thesis published by University of London Institute of Education

Noss, R. and Hoyles, C. (1992) Looking back and looking forward, in C. Hoyles and R. Noss (eds.) *Learning Logo and Mathematics*, Cambridge, MA: MIT Press

Noss, R., Hoyles, C. (1992) Logo mathematics and Boxer mathematics: Some preliminary comparisons, *Proceedings of the Sixteenth International Conference of the Psychology of Mathematics Education*, 2, 186-193

Noss, R. (1992) The social shaping of computing in mathematics education, in D. Pimm and E. Love (eds.), *The Teaching and Learning of School Mathematics*, Hodder and Stoughton

Papert, S., Watt, D., *et al.* (1979) *Final Report of the Brookline Logo Project, Part 2*, AI Memo No. 545, MIT, Cambridge, MA

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

Papert, S. (1981) Computers and computer cultures, *Creative Computing*, 7, 82-92

Papert, S. (1987) Computer criticism versus technocentric thinking, *Educational R esearcher*, 16/1, 22-30

Plomp, T.J. and Pelgrum, W. J. (1992) Introduction of computers in education: State of the art in eight countries, *Computers and Education*, 17/3, 249-258

Sherin, B., diSessa, A., Hammer, D. (1993) Dynaturtle revisited: Learning physics through collaborative design of a computer model, in *Interactive Learning Environments*, 3 /2, 91-118

Sinclair, K., Moon, D. (1991) The philosophy of LISP, *Communications of the ACM*, 34/9, 41-47

Soloway, E. (1990) Quick, where do the computers go?, *Communications of the ACM*, 34/2, 29-33

White, B. (1993) Thinker tools: Causal models, conceptual change, and science education, *Cognition and Instruction*, 10/1, 1-100