# Children, Chunking, and Computing

CRAIG GRACI[1], RANDOLPH ODENDAHL[1] AND JACK NARAYAN[2]
[1]*Department of Computer Science*
[2]*Department of Mathematics*
*State University of New York, College at Oswego, NY 13126, USA*

**Abstract** The potential of educational microworlds for the development of thought processes is widely suggested. The main thesis of the present paper is that this potential, particularly as it relates to the development of structured thinking, may be more fully realizable than is typically the case. The focal point of this investigation is a point-and-click, bottom-up, structured-programming methodology requiring nearly no syntactic sophistication on the part of the participant. This methodology is presented in the context of a simple educational microworld. The significance of the methodology is that it tends to blur the distinction between cognitive chunking and procedure crafting.

The potential of educational microworlds for the development of thought processes is widely suggested. The main thesis of this paper is that this potential, particularly as it relates to the development of structured thinking, may be more fully realizable than is typically the case. The notion of structured thinking is closely related to *chunking*, the phenomenon of encapsulating "thoughts," "actions," and so forth, at a particular level for subsequent use. The computational analog to the structuring of thought via cognitive chunking is found in *procedural abstraction*. Chunking in people is often "passively" realized in that it results from reflective observation. Chunking in computer languages normally requires the explicit writing of a procedure, an "active" process. The relationship between passive chunking which results from people doing things and then thinking about what they have done, and active chunking, achieved by computer programming, is the central theme of this paper.

The focal point of this investigation is a point-and-click, bottom-up, structured-programming methodology requiring nearly no syntactic sophistication on the part of the participant. This methodology is presented in the context of a simple educational microworld. The significance of the methodology is that it tends to blur the distinction between passive cognitive chunking and active

procedure crafting. The perceived result is a greater likelihood of *faithfully capturing, in code, passively developed thought structures* than is likely in a syntax-sustained programming environment.

It is important to emphasize at the outset that the methodology to be presented transcends microworld domain. It could well be applied to a number theory microworld, to a musical melody microworld, or to a microworld in virtually any domain. In essence, the methodology discussed in this paper provides a useful integration of *microworld interface* and *microworld proper*. For purposes of presentation, the methodology is employed in the context of a variant of the traditional "turtle" microworld. This paper is not about the microworld per se, but rather it is about educational microworld *methodology*.

## Educational Microworlds

Though frequently written about, the concept of "educational microworld" remains ambiguous to the extent that a brief working description of one's intended meaning is usually warranted. The motivational aspects of *phenomenological* definitions is undeniable. Some examples of phenomenological definitions of microworld are: A microworld is . . .

> . . . a piece of reality, simple enough to "get ahold of" but rich enough to encourage exploration (Clements, 1989). Paraphrase of S. Papert.

> . . . a small playground for the mind (Clements, 1989).

> . . . a well-defined, but limited environment in which interesting things happen and in which there are important ideas to be learned (Goldenberg, 1982).

As a complement to such definitions, we find a *compositional* definition to be valuable. For example, a microworld is . . .

> . . . a collection of *objects (classes of objects)* and *associated functionality*, together with a well-defined *extensibility mechanism*.

This sort of definition provides a firm footing on which to study particular issues in learning. The nature of specific objects coupled with the extensibility mechanism define what can best be learned in the microworld. That is, they collectively serve to specify, albeit implicitly, the *learning objectives* that the given microworld can best satisfy.

## The Quarter Arc World—QuArc World

### Description of the QuArc World

The Quarter Arc World, QuArc World, is a turtle geometry microworld conceived upon two commands for drawing the quarter arc of an "underlying" circle.

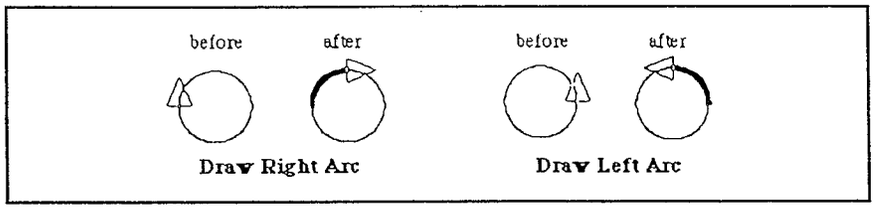


**Figure 1.** Quarter Arc Commands

It has plenty of additional functionality which can programmably be activated/deactivated as warranted by specific instructional objectives, most of which will not be discussed here.

There is one turtle in the microworld.

*The **thematic functionality** associated with the microworld includes:*

**Draw Right Arc (DRA)**: The turtle moves to the right through an arc coincident with one quarter of the underlying circle, leaving a trace.

**Draw Left Arc (DLA)**: The turtle moves to the left through an arc coincident with one quarter of the underlying circle, leaving a trace.

**Grow (G)**: The underlying circle is expanded by an amount specified by a combination of two system paramenters.

**Shrink (S)**: The underlying circle is shrunk by an amount specified by a combination of two system parameters.

*Prominent **secondary functionality** includes:*

**Jump Right Arc (JRA)**: The turtle moves to the right through an arc coincident with one quarter of the underlying circle, without leaving a trace.

**Jump Left Arc (JLA)**: The turtle moves to the left through an arc coincident with one quarter of the underlying circle, without leaving a trace.

**Turn Right (TR)**: The turtle turns to the right an amount, in degrees, specified by a system parameter.

**Turn Left (TL)**: The turtle turns to the left an amount, in degrees, specified by a system parameter.

**Jump Right (JR)**: The turtle jumps to its right an amount, in turtle steps, equal to the radius of the underlying circle (which is variable).

**Jump Left (JL)**: The turtle jumps to its left an amount, in turtle steps, equal to the radius of the underlying circle (which is variable).

**Clear (C)**: The Display Window clears.

*The **featured functionality** is:*

**Begin Program (BP)**: Readies the system to begin a program. All subsequent commands will be stored in the program until the End Program (EP) instruction is encountered.

**End Program (EP)**: Terminates the program currently being developed and makes it available for subsequent use. A box pops up which provides a mechanism to activate the new program.

**Display Programs (DP)**: Displays the active user-defined programs textually in a language called Blue Clay Logo.

**Clear Programs (CP)**: Empties the set of active user-defined programs.

### QuArc World User Interface

The user interface that we use for the QuArc World is illustrated in Figure 2. It must be emphasized that this interface, the Control Panel in particular, is programmable. Nowhere near the amount of functionality shown in the figure should ever be made available to children (or adults) in their first experiences with the QuArc World. Much more functionality is ultimately available.

### Importance of Hiding Functionality

Most domains are best introduced by focusing on various subsets of the knowledge loosely associated with the domains. One of the great strengths of educational microworlds is that they facilitate **focusing** on important concepts in isolation of one another and in collaboration with one another. Of course, in order to take full advantage of focusing, facilities must be available so that

moving from one version of a microworld (subset of functionality) to another is a very simple matter, like selecting an item from a menu. A graded set of QuArc World instances is seen in the following section.

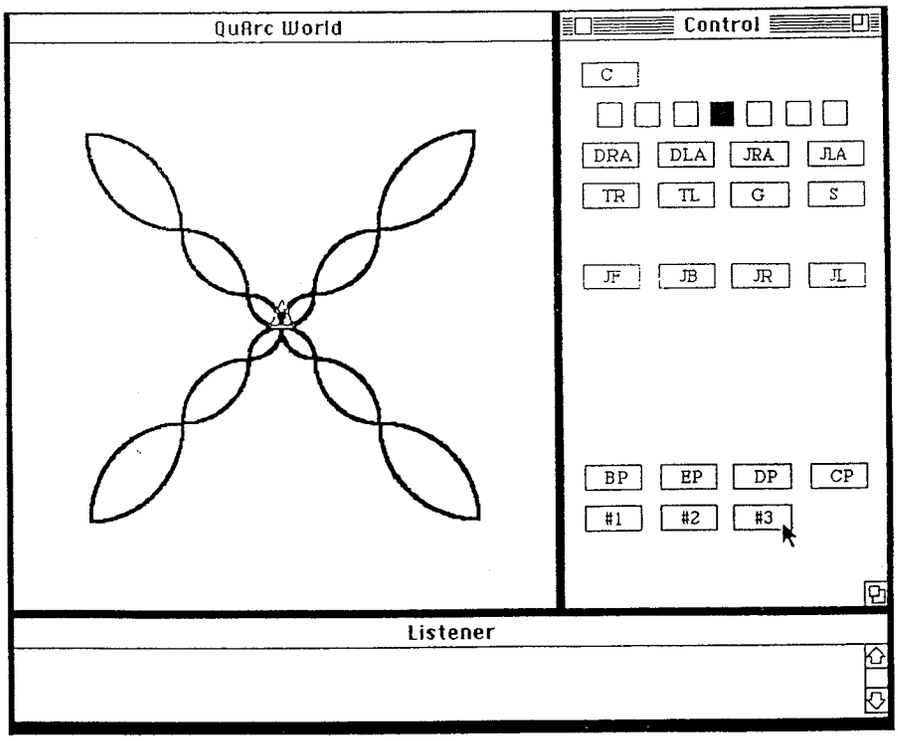


**Figure 2.** QuArc World Interface

## Basic QuArc World Functionality

This section introduces by example the basic functionality of the microworld. In first experiences with the QuArc World, the beginner is best provided with the Control Panel shown in Figure 3.

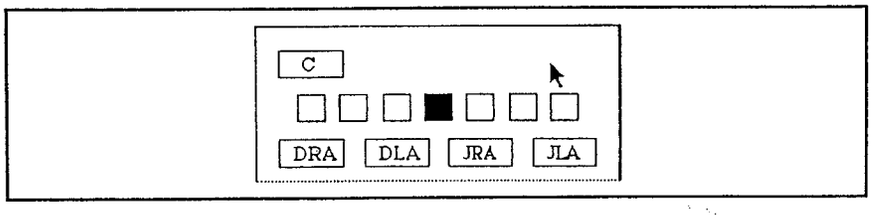


**Figure 3.** Introductory functionality

It is important that the remaining boxes be absent, as their presence tends to dissipate the learners' attention by seducing them in too many directions at one time. *The significance of this phenomenon, which is generally acknowledged, may be greatly underestimated. Functionality hiding may be just as important to microworld educational practice as information hiding is to the field of software engineering.* Using just the functionality offered, the learner can construct circles, explore movements in right and left circular arcs, and do many other things. One exercise that children as young as four years old find very engaging is to trace, in a pretty color, simple figures, such as those shown below, which are provided to them in basic black. (Colors are selected by simply clicking on one of the boxes in the second row.)

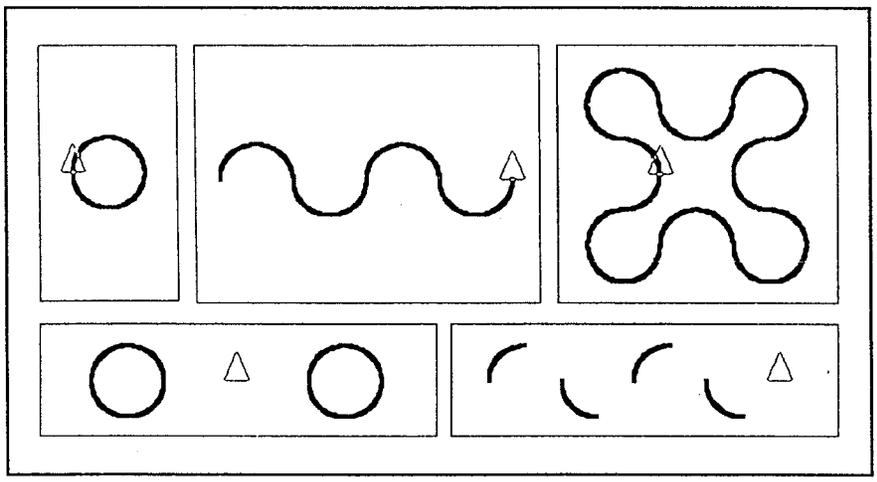

**Figure 4.** Exercises for exploration of introductory functionality

Soon experiences can be greatly enriched by adding the functionality associated with four additional boxes, as shown in Figure 5.

The drawing of figures such as those in Figure 6 proves to be a rewarding experience which acquaints the new user with the **turning** and **growth** commands, in their default mode.

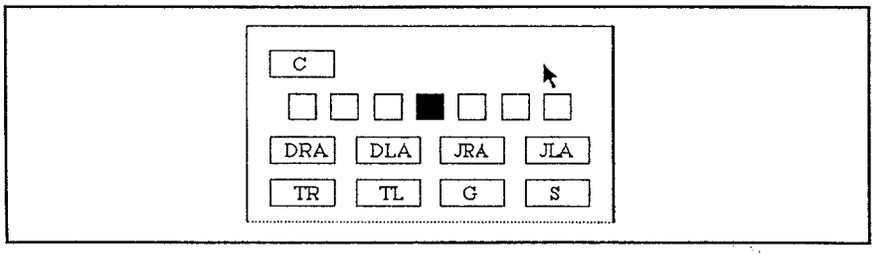


**Figure 5.** Level two functionality

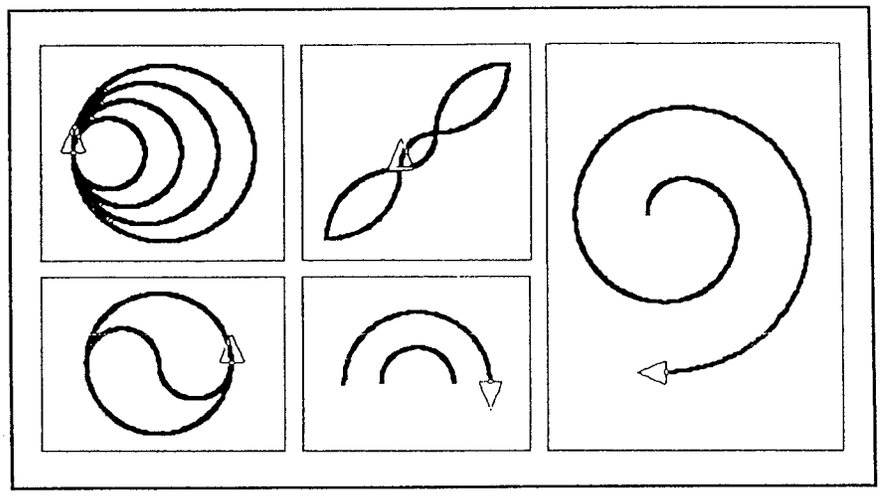

**Figure 6.** Exercises for exploring level two functionality

The straight jumping commands-JF, JB, JR, and JL-are useful enough that it makes sense to acquaint the student with them prior to embarking on programming. This is again best done by posing as exercises the drawing of a variety of figures which feature their use.

An important point is that a balance between goal-directed learning and discovery learning can be achieved by providing learners with a graded sequence of microworld *instances.* Considerable flexibility in terms of providing specific subsets of functionality is of paramount importance to the successful achievement of this important balance.

## Point-and-Click, Bottom-Up, Structured Programming

It is undesirable to introduce too much functionality before programming facilities are introduced. That which has to this point been introduced is quite sufficient.

### Interacting with a Computer and Programming a Computer

A computer program is often described as a sequence of instructions which can be (automatically) run by a computer. Programming means specifying such a sequence of instructions. In using the microworld in the manner described in the preceding section, it may be best to describe the learner's mode of operation as *interacting* with the computer, rather than *programming* the computer. In most conventional systems, though certainly not all, interacting and programming tend to be mutually exclusive activities. The methodology about to be described equates the two activities.

## Programming by Doing

The methodology advocated here for the construction of programs is one currently in vogue in the composition of music. This methodology, which is used by musicians who work with sequencers, computer-controlled sequencers in particular, appears to have fairly general appeal. These people "program" by doing. The tunes (riffs, whatever) that they perform are simply saved as a sequence of instructions for a music machine. This sequence of instructions serves as a program which can be replayed by the music machine, or which can be employed in the development of more elaborate compositions.

In the QuArc World under discussion, a program is "written" by clicking in the Begin Program (BP) box, doing something, and then clicking in the End Program (EP) box. The something which gets done is captured not only (visually) in the QuArc World display window, but also (textually) as a computer program, which can subsequently be run. Upon clicking in the EP box, a new box appears. Clicking on this new box will cause the associated program to be run just as, for example, clicking in the TR box causes the turtle to turn to its right. Such a new program can quite naturally be used in the development of more elaborate programs.

The following QuArc World session illustrates the drawing of a picture or, depending on how you view the activity, the writing of a program. Imagine that the control panel is that shown in Figure 2.

Initially, the Display Window and the (truncated) Control Panel look (in condensed, slightly contorted form) as shown in Figure 7.

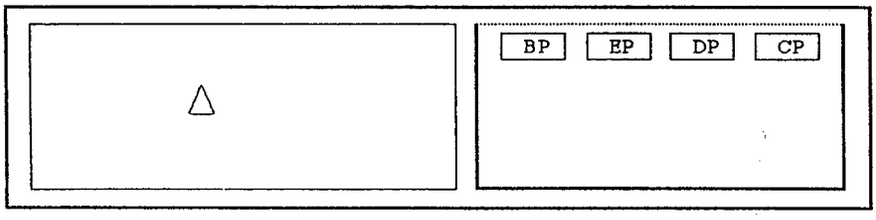


**Figure 7.** Display window and control panel

By clicking out the sequence BP DRA DRA DRA DRA EP in the Control Panel, the QuArc World in the state shown in Figure 8 is realized.

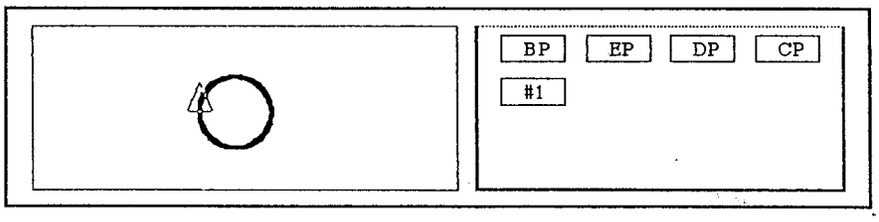


**Figure 8.** Display window and control panel after BP DRA DRA DRA DRA EP

Note that the added box in the control panel comes into existence only after the EP box is clicked in. (This particular point-and-click programming environment does not permit recursive programming.) At this point, the program DRA DRA DRA DRA, which amounts to drawing a circle, may be run simply by clicking in box #1. After clearing the screen (by clicking on box C), clicking out the sequence BP JL #1 JR EP leaves the QuArc World in the state shown in Figure 9.

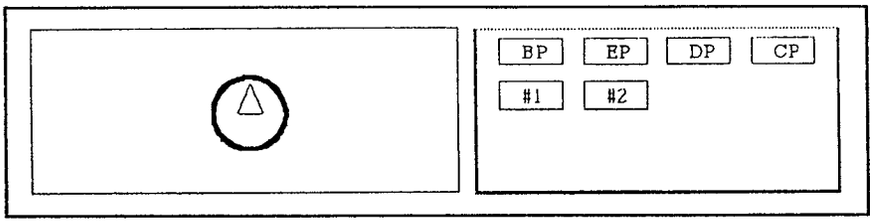


**Figure 9.** Display window and control panel after BP JL #1 JR EP

Again, the added box in the control panel comes into existence only after the user clicks inside of the EP box. At this point, the program JL #1 JR, a turtle invariant program in which the turtle draws a circle centered about itself, may be run simply by clicking in box #2. After clearing the screen, clicking out the sequence BP #2 JR JR #2 JR JR #2 JL JL JL JL EP results in the QuArc World state of Figure 10.

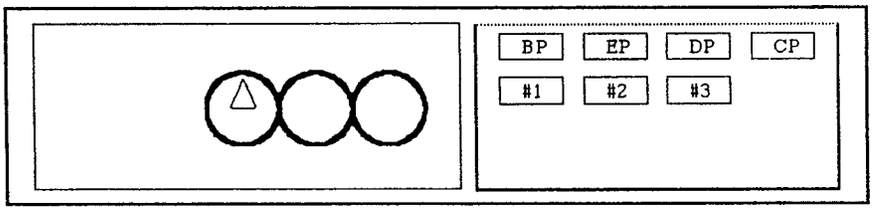


**Figure 10.** Display window and control panel after BP #2 JR JR #2 JR JR #2 JL JL JL EP

Program #4 is a very minor variation of program #3. The program is written by clicking on BP #3 TR EP after clearing the screen. (See Figure 11).

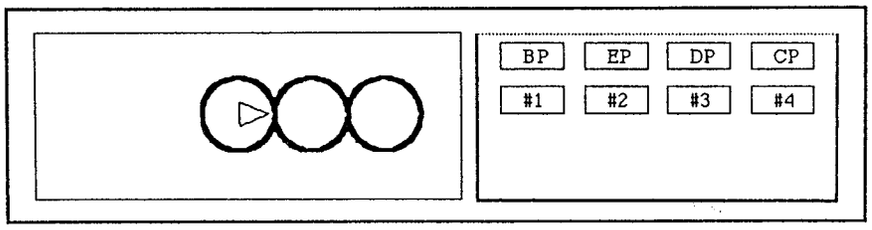


**Figure 11.** Display window and control panel after BP #3 RT EP

Finally, program #5, presumably the program of primary interest, is obtained by clicking out the sequence BP #4 #4 #4 #4 EP. (See Figure 12).

Box #5 at this point denotes a program which draws a "cross of circles."

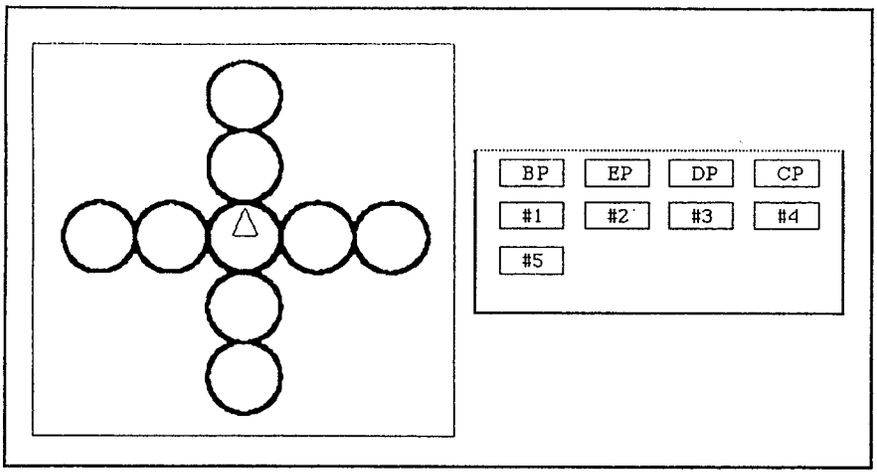


**Figure 12.** Display window and control panel after BP #4 #4 #4 #4 EP

## Textual Representation of Programs

It is often interesting (for instructors and for participants) to review the structure of the programs written via the programming-by-doing methodology. Clicking on the Display Programs (DP) button results in the programs being displayed to a Listener Window in the Blue Clay Logo programming language. Blue Clay Logo is an "object-oriented microworld implementation language" which was used to implement the QuArc World. Blue Clay Logo is itself written in yet another object-oriented Logo, ObjectLogo (Paradigm, 1990).

```
to perform #5
do
  #4, #4, #4, #4
end

to perform #4
do
  #3, TR
end
```

```
to perform #3
do
  #2, JR, JR, #2, JR, JR, #2, JL, JL, JL, JL
end
to perform #2
do
  JL, #1, JR
end
to perform #1
do
  DRA, DRA, DRA, DRA
end
```

Note that the programs are displayed in a top-down manner. The highest level program, #5, is a "chunk" of four lower level programs. Program #3 is a "chunk" of three programs, and so forth. It is worthwhile to note the structure inherent in this program and to contrast its structure with the following sequence, functionally equivalent to program #5, of (unstructured) QuArc World primitives.

JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JL JL JL JL TR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JL JL JL JL TR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JL JL JL JL TR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JR JR JL DRA DRA DRA DRA JR JL JL JL JL TR

## An Educational Research Agenda

It is inviting to conjecture about the degree to which the structure of programs is reflective of the thought processes of the individuals who "write by doing" the programs. The question of relevance to educational microworld practice is twofold: Does extended (prolonged) "programming-by-doing" in a particular domain yield richer and richer program structures, as measured by the programs which are essentially *side effects* of actually working in the domain? Assuming so, does this enriched structure indicate significantly enriched thinking? These are issues which may be determinable through extensive (rigidly controlled) experimentation in well-defined environments (educational microworlds).

## References

Clements, H. (1989). *Computers in elementary mathematics education.* Englewood Cliffs, NJ: Prentice Hall.

Goldenberg, E.P. (1982). *Byte*, 2, pp. 210-229.

Paradigm Software. (1990). *ObjectLogo* [computer program]. Cambridge, MA: Paradigm Software.

## Resources

Graci, C., Narayan, J., & Odendahl, R. (1989). Bunny numerics: A number theory microworld. In E. Kaltofen & S.M Watt (Eds.), *Computers and mathematics.* New York: Springer-Verlag.

Graci, C. (1990). *BLUE Clay logo: Language reference manual.* BLUE Report #12. Computer Science Department, State University of New York, College at Oswego, Oswego, New York.

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

Papert, S. (1987). Microworlds: Transforming education. In R. Lawler & M. Yazdani (Eds.), *Artificial Intelligence and Education, Vol. 1.* Norwood, NJ: Ablex.

Piaget, J. (1971). *Genetic epistemology.* New York: W.W. Norton.

Polya, G. (1973). *How to solve it.* Princeton, NJ: Princeton University Press.

Silver, E. (1987). Foundations of cognitive theory and research for mathematics problem-solving instruction. In A. Shoenfeld (Ed.), *Cognitive science and mathematics education.* Hillsdale, NJ: Lawrence Erlbaum.

Simon, S. (1981). *The sciences of the artificial.* Cambridge, MA: The MIT Press.

Solomon, C. (1987). *Computer environments for children.* Cambridge, MA: The MIT Press.

Swan, K. (1989). Logo programming and the teaching and learning of problem solving. *Journal of Artificial Intelligence in Education, 1*(1), 73-92.