The Future of Educational Computing Research: The Case of Computer Programming

DOUGLAS H. CLEMENTS
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall
Buffalo, NY 14260, USA
Clements@acsu.buffalo.edu

What directions should research on computers in education take in the next century? We examine one application, computer programming, with a particularly long and rich research history (Clements & Meredith, 1993). We review the literature and discuss implications for future research. While emerging from the domain of computer programming, these implications have wide applicability across various applications of computers. For example, in all applications of computer to education, we need to learn to operationalize and optimize the complex webs of variables that determine efficacy. We need continuing research and development to expand our knowledge of what students and teachers learn in various environments; test conjectures and theories abstracted post hoc in extant research; and investigate how unique features of various programming environments interact with the goals and content of specific subject matters domains and the particular features of various teaching and learning situations to promote learning and development. Future research should also ask how computers might be successfully implemented in a manner consistent with systemic reform.

What directions should research on computers in education take in the next century? We examine one application, computer programming, with a particularly long and rich research history (Clements & Meredith, 1993). We review the literature and discuss implications for future research. While emerging from the domain of computer programming, these implications have wide applicability across various applications of computers.

MATHEMATICS

Early reviews concluded that effects of computer programming on overall mathematics achievement were either conflicting or positive but not consistently strong (Clements, 1985; Clements & Meredith, 1993). Recent reviews, in contrast, are quite positive (Clements & Sarama, 1997b; McCoy, 1996). The negative cases remind us that results are by no means guaranteed and that many factors must be considered (Hamada, 1987). Mere "exposure" to computer programming may often fail to provide measurable benefits. The positive cases begin to illustrate how to effectively teach mathematics with computer programming (Barker, Merryman, & Bracken, 1988; Butler & Close, 1989; Clements & Meredith, 1993; Hoyles & Noss, 1987; Miller, Kelly, & Kelly, 1988; Oprea, 1988; Salem, 1989; Tanner, 1992; Yelland, 1995).

Geometry and Spatial Sense

The emphasis on turtle graphics in the Logo language and its imitators has generated a considerable body of research in the domain of geometry and spatial sense, with generally positive results. Why is the turtle helpful? Students construct initial spatial notions from *actions* (Piaget & Inhelder, 1967), and they command the turtle to move (Clements & Battista, 1992b). In this way, Logo activities facilitate students' progression to higher levels in the van Hiele hierarchy of geometric thinking (van Hiele, 1986). That is, students move from thinking of geometric shapes as wholes ("It just looks like a rectangle") to conceptualizing them in terms of their properties ("It's a rectangle because it has opposite equal sides and 90° angles"). Guided Logo experience significantly enhances students' concepts of plane figures (Butler & Close, 1989; Clements, 1987; Clements & Battista, 1992b; Hughes & Macleod, 1986) and other geometric objects (Yusuf, 1994).

For example, one large study involved 656 K-6 students working on *Logo Geometry* activities (Battista & Clements, 1988; Battista & Clements, 1991; Clements & Battista, 1992a; Clements & Battista, 1992b). Control students (644) worked with their regular geometry curriculum. After introductory path activities (e.g., walking paths, creating Logo paths), students engaged in off- and on-computer activities exploring squares and rectangles. For example, they identified these shapes in the environment, wrote Logo procedures to draw them, and drew complex figures with these procedures. Logo students performed better over all. They demonstrated flexible consideration of multiple properties of geometric shapes that may help lay the groundwork for hierarchical classification. Similarly, in another study students were better able to apply their knowledge of geometry than a comparison group. There was no difference in knowledge of basic geometric facts; therefore, the use of Logo influenced the way in which students represented their knowledge of geometric concepts (Lehrer, Randle, & Sancilio, 1989). Middle school students move to higher levels of conceptualizing and being to integrate visual and symbolic representations (Clements & Battista, 1988; Clements & Battista, 1989; Clements & Battista, 1992a; Hoyles, Healy, & Sutherland, 1991; Hoyles & Noss, 1988; Kierna & Hillel, 1990).

Several research projects have investigated the effects of computer programming on students' conceptualizations of specific ideas in mathematics. For example, Logo experience appears to significantly affect students'; ideas about angle. Responses of control students in one study reflected either little knowledge of angle or common language usage such as "a line tilted." In comparison, the Logo students indicated more generalized and mathematically oriented conceptualizations, such as "Like where a point is. Where two lines come together at a point" (Clements & Battista, 1989). Several researchers have reported a positive effect of Logo on students' angle concepts (Clements & Battista, 1989; du Boulay, 1986; Frazier, 1987; Kieran, 1986; Kieran & Hillel, 1990; Olive, Lankenau, & Scally, 1986).

In a similar vein, Logo experiences may affect competencies in linear measurement. They permit students to manipulate units and to explore transformations of unit size and

number of units free of the dexterity demands associated with measuring instruments (Campbell, 1987). Turtle geometry is inherently measurement-based. Therefore, Logo can provide meaningful tasks and models for thinking about number and arithmetic operations as they apply to length and angle. In such an environment, students are motivated to integrate numeric and geometric notions. They receive feedback that aids them to reflect on their own thinking about these ideas and how they are connected (Clements, Battista, Sarama, Swaminathan, & McMillen, 1997). Older students can learn more sophisticated ideas about measurement, as well as about directions and coordinates (Kynigos, 1992).

Some research has not shown positive results (Johnson, 1986). Again, "exposure" without teacher guidance often yields little learning (Clements & Meredith, 1993). One reason is that students do not always think mathematically, even if the Logo environment invites such thinking. For example, some students rely excessively on visual cues and avoid analytical work (Hillel & Kieran, 1988). Such a visual approach is not related to visualization ability but to the role of visual "data" of a geometric figure in determining students' constructions. Although helpful initially, overuse inhibits students from arriving at mathematical generalizations. There is little reason for students to abandon visual approaches unless teachers present tasks whose resolution requires an analytical, generalized, mathematical approach.

In sum, studies that have shown the most positive effects involve carefully planned sequences of computer programming activities. Teacher mediation of students' work with those activities is necessary for successful construction of geometric concepts. Mediation includes helping students to forge links between computer and other experiences (Clements & Battista, 1989; Lehrer & Smith, 1986b).
Future research needs to establish the relationship between specific characteristics of computer programming environments and these gains. Simply stated: Is it the turtle or the programming? Or at least, what is the contribution of each? In a similar vein, we know that enriching the primitives and tools available to students facilitates their construction of geometric notions and increases analytical, rather than visual, approaches (Clements & Battista, 1992s; Clements & Sarama, 1995; Kynigos, 1992). Future research should establish the relationship between specific enhancements (and in the broader picture specific features of the teaching and learning environment) and specific aspects of conceptual and procedural learning (Hiebert & Carpenter, 1992).

Number, Arithmetic, and Algebra

Arithmetic has long dominated the elementary school mathematics curriculum. While results are mixed, the effect of nonstructured computer programming on arithmetic skills usually appears small (Butler & Close, 1989), though the finding that time spent does not *decrease* such skills is significant. Although computer programming does not provide efficient practice on arithmetic processes, it can provide a context in which there is a real need for these processes and in which children must clearly conceptualize which operation they should apply. For example, 1st-grade children determined the correct length for the bottom line of their drawing by adding the lengths of the three horizontal

lines that they constructed at the top of the tower: 20 + 30 + 20 = 70 (Clements, 1983-84). If used in such reflective ways, computer programming can aid growth in arithmetic skills equally to CAI drill on those same skills (Carmichael, Burnett, Higginson, Moore, & Pollard, 1985; Wilson & Lavelle, 1992). Further, studies have shown that activities with young students can facilitate basic number sense; for example, learning relationships between size of numbers and the length of a line drawn by the turtle (Bowman, 1985; Clements, 1987; Hughes & Macleod, 1986; Robinson, Gilley, & Uhlig, 1988; Robinson & Uhlig, 1988). Such experience can positively affect achievement test scores (Barker et al., 1988).

Computer programming in Logo or BASIC can help students generalize their understanding of number and arithmetic to understand variables and algebra (Carmichael et al., 1985; Findlayson, 1984a; Milner, 1973; Nelson, 1986; Oprea, 1988; Soloway, Lochhead, & Clement, 1982). However, such learning can be limited; for example, students may not fully generalize the variable idea as used in computer programming to other situations (Lehrer & Smith, 1986a). Again, mere "exposure" without teacher planning and mediation is insufficient. Students need to build a conceptual framework based on intuitions and "primitive conceptions" upon which they can build later algebraic learning (Noss & Hoyles, 1992). Significantly, compared to not only paper-and-pencil, but also spreadsheet environments, students working collaboratively with computer programming more frequently use formal language as a means of articulating general ideas (Hoyles et al., 1991). They combine natural and computer language and ideas. Supporting this beneficial tendency may be students' realization that the computer does not understand a natural language formulation. Students may come to expect that they will have to formalize to communicate their generalization and represent their task solution (Hoyles et al., 1991).

In summary, there is some evidence that Logo provides an environment in which number sense (National Council of Teachers of Mathematics, 1989) can provide an "entry" to the use of the powerful tool of algebra. It is an environment in which some students perceive the use of formalizations such as variables as natural and useful. Again, however, we find that students' ability to generalize their programming-based notion of variable may depend to a great degree on the depth of their experience and the instructional support given them.

Future research needs to determine how to connect specific programming activities to the development of specific aspects of number sense and algebraic conceptualizations. What kinds of conceptual frameworks might be developed within typical classroom situations and what types of organized instruction and thoughtfully structured tasks engender those frameworks (Hillel & Samurçay, 1985; Milner, 1973; Sutherland, 1987)?

Ratio and Proportion

Researchers have observed similar facilitative effects of computer programming environments on ratio and proportion tasks. On a geometric proportion task, students used additive strategies on paper-and-pencil tasks, but none used such strategies on the

related Logo tasks (Hoyles & Noss, 1989). The reason lay in the interaction between students' formalization and computer feedback. They formalize proportional relationships algebraically as Logo programs. They receive graphical feedback regarding their mathematical intuitions. On pencil-and-paper, the formalization is less salient; the feedback is absent. Students may have abandoned additive thinking because the computer provided a way to think about the general within the specific. Paper-and-pencil invited a fixed answer to a fixed question. The computer allowed exploration to escape from mental "blocks" and activated a dynamic answer. Posing the task of writing a superprocedure that would handle all cases promoted additional development. Thus, again, we see the encouragement of more generalized and abstract views of mathematical objects within structured computer programming environments. We also see similar implications.

Future research needs to address that most explanations for computer programming's benefits are generated post hoc. Therefore, they need to be tested by other researchers and either confirmed, altered, or rejected. In addition, particular features of computer programming environments and activities need to be causally connected to the development of sophisticated mathematical strategies. That is, researchers need to identify the specific attributes of computer programming that engenders students learning of better strategies and concepts in the domain of ratio and proportion.

PROBLEM SOLVING AND HIGHER-ORDER THINKING

As with mathematics, approaches to teaching problem-solving and higher-order thinking skills with computer programming based only on exposure usually have not been positive (Bruggeman, 1986; Dalton, 1985; LeWinter, 1986; Mitterer & Rose-Drasnor, 1986), though there are some exceptions in which it did raise problem-solving scores (Choi & Repman, 1993; Dvarskas, 1984). Other studies were based on variations of the conceptual framework hypothesis, agreeing with Papert (1980) that Logo can make the abstract concrete, accelerating cognitive development. Some reported gains (Miller et al., 1988; Rieber, 1987), but others found no significant differences (Clements & Gullo, 1984; Howell, Scott, & Diamond, 1987). With teacher planning and mediation, however, computer programming can facilitate higher-order thinking (Billings, 1986; Reed, Palumbo, & Stolar, 1988; Roblyer, Castine, & King, 1988; Wiburg, 1989). As with mathematics, then, the most positive results have involved teacher mediation based on a well-developed theoretical foundation (Clements, 1990; De Corte & Verschaffel, 1989; Delclos & Burns, 1993; Lehrer, Guckenberg, & Lee, 1988; Lehrer, Harckham, Archer, & Pruzek, 1986; Littlefield et al., 1988). Positive effects also take a considerable time (Liu, 1997, found 150 hours of experience was necessary).

In addition, strength of effects may differ for different processes. For example, planning shows weak effects in many studies, and may have to be carefully mediated. Other aspects of problem solving, such as deciding on the nature of the problem, selecting a representation for solving the problem, and especially cognitive monitoring, show more consistent effects (Clements & Nastasi, 1988; Clements & Sarama, 1997b). Indeed, certain processes and strategies, especially when mediated by the teacher, show more

growth in computer programming environments than in noncomputer environments involving paper-and-pencil or manipulative tasks (Clements, 1986; Clements, 1990; Swan, 1991; Swan & Black, 1989). A recent metaanalysis was conducted on problem solving (a metaanalysis combines the results of many studies mathematically, measuring the results of each in terms of "effect sizes," where an effect size of 1 indicates that the experiment group scored 1 standard deviation above the control group). The researchers found that 84% of the effect sizes favored computer programming over comparison groups, with an average moderate effect size of .41 (Liao & Bright, 1989).

Future research might ascertain when, how, and why such positive effects have been found. Computer languages may provide malleable representations that students can inspect, manipulate, and test, helping them connect concrete and formal understandings (Swan, 1991). The educational environment may need to include a general framework for metacognitive strategies and the embedded application of these strategies within a specific domain or metaphorical connections between computer programming processes and higher-order thinking processes (Clements, 1990). The social interactions within the computer programming educational environment might be critical (Clements & Nastasi, 1992b).

LANGUAGE ARTS

It may be surprising to some that results are also generally positive for computer programming's effect on language arts. For example, early programming experiences can engender language rich with emotion, humor, and imagination in young children (Genishi, McCollum, & Strand, 1985; Yelland, 1994a). In a similar vein, working with Logo in a narrative context (a) enhances language-impaired preschool students' perceptual-language skills (Lehrer & deBernard, 1987); (b) increases kindergartners' readiness scores on visual discrimination, visual motor skills, and visual memory (Reimer, 1985); and (c) increases first graders' scores on assessments of visual motor development, vocabulary, and listening comprehension (Robinson et al., 1988; Robinson & Uhlig, 1988). The talk students weave around their Logo is impressively task-related, other-directed, cooperative, and nonplayful (Genishi, 1988).

Logo work may also enhance reading skills. Emersion in a Logo culture can lead to increases in language mechanics and reading comprehension, even without direct instruction (Studyvin & Moninger, 1986). These effects may be delayed (Clements, 1987). There are even accounts of Logo assisting the learning of foreign languages. For example, 5th- and 6th-graders in one study learned both Logo and German vocabulary. This method also improved their attitudes (Tracy & Williams, 1990).
Future research is needed that extends and explains these positive findings. Even more than in other areas, we need to know what aspects of computer programming are responsible for growth in the various components of language arts.

CREATIVITY

Early observational research suggested that Logo drawing helps students create pictures that are more elaborate than those that they can create by hand. They transfer components of these new ideas to artwork on paper (Vaidya & McKeeby, 1984). Such computer drawing is appropriate for children as young as three years, who show signs of developmental progression in the areas of drawing and geometry during such computer use (Clements & Nastasi, 1992a; Tan, 1985). Other studies showed an increase in figural creativity on transfer tests, although gains in some were moderate (Clements & Gullo, 1984; Clements & Nastasi, 1992a; Horton & Ryba, 1986; Reimer, 1985; Roblyer et al., 1988; Wiburg, 1987) and occasionally nonsignificant (Mitterer & Rose-Drasnor, 1986; Plourde, 1987). Originality, in contrast to fluency or flexibility, was most often enhanced.

One study analyzed the reasons that these aspects would be enhanced by Logo (Clements, 1991). A Logo group significantly outperformed both a comparison group receiving nonLogo creativity experiences (word processing and graphics programs) and a nontreatment control group on an assessment of figural creativity. In addition, the Logo group significantly outperformed the control group on an assessment of verbal creativity. These results support the hypothesis that Logo was enhancing not just students' figural knowledge, but the processes involved in creative though.

Future Research. As with problem solving and higher-order thinking, we need to know more about the characteristics of computer programming, and mediated computer programming environments, that lead to the development of specific components of creativity.

SOCIAL-EMOTIOANL DEVELOPMENT

Contrary to a popular view of programmers held by nonprofessionals strong beneficial effects of computer programming have been reported in the area of social and emotional development. Teachers report that students exposed to computer programming are more likely to interact with their peers. They engage in group problem solving, sharing, and acknowledging expertise and creative thinking. Social isolates benefit the most (Carmichael et al., 1985; St. Paul Public Schools, 1985). Children are eager to cooperate and share what they have learned with others (Genishi, 1988). Thus, computer-programming environments can facilitate social interaction and positively focus that interaction on learning.

Research also indicates that students learn to solve social problems cooperatively and flexibly in Logo classrooms (Carmichael et al., 1985). One study indicated that students work cooperatively more often on computers—with either Logo or CAI drill—than off (Clements & Nastasi, 1985). Interestingly, they also got into more conflicts (possibly because they interacted more). Conflicts alone can be important. In another study, students working in Logo, compared to those working in spreadsheet and paper-and-pencil environments, experienced more conflicts, which had the effect of destabilizing inappropriate solutions before students formed incorrect generalizations. This helped

students keep "on track" (Hoyles et al., 1991). These conflicts, and the social interaction at the computer, have been identified as leading to positive gains in learning (Healy, Pozzi, & Hoyles, 1995).

Additional research has found that not only does Logo generate useful conflicts, but also that students working with Logo, compared to students working with CAI programs, are more likely to resolve these conflicts (Clements & Nastasi, 1985). After experiencing CAI drill, students generated more oppositional behaviors in their noncomputer drill work. They may have found that this drill lacked the excitement of the CAI drill (Clements & Nastasi, 1985).

In a similar vein, students working together on Logo tasks spent a significant proportion of their time resolving conflicts (Lehrer & Smith, 1986b). Finally, research indicates that the *type* of conflict—social or cognitive—is critical (Nastasi, Clements, & Battista, 1990). Students working in Logo evinced more *cognitive* conflict, and attempts at and successes in resolution of these conflicts, than those working with CAI did. Differences were not evident for social conflict or its resolution. Thus, the effects of Logo seemed to be specific to disagreements about *ideas*. Moreover, only those behaviors indicative of cognitive conflict were related to scores on a measure of problem solving (higher-order thinking). In particular, it was the successful resolution of those conflicts, more than the occurrence or attempts to resolve, that accounted for variability in problem-solving performance. Opportunities to experience and resolve conflicts are necessary for the development of problem-solving competencies. Therefore, Logo contexts may enhance the development of social and cognitive problem-solving skills.

To optimize learning, educators must also consider goals and tasks. Paired work at the computer coupled with the coordination of others' perspectives in group discussions, may be most advantageous for learning conceptually based mathematics. In contrast, for certain "technology-driven" tasks, in which ideas are directly generated from computer-based actions, concentrated work at the computer may prove to be more efficient (Healy et al., 1995). Thus, there may be times when individual work is more beneficial and computer feedback (alone) adequate (Hughes & Greenhough, 1995).

According to their teachers, students working with Logo experience an increase in self-esteem and confidence. This may occur only *if* their teacher gives them greater autonomy over their learning and fosters social interaction (Carmichael et al., 1985; Fire Dog, 1985; Kull, 1986). Logo can provide special needs students with prestige and respect from their peers, enhancing their self-esteem (Michayluk, Saklofske, & Yackulic, 1984). Attitudes toward learning are positively affected in some classrooms (Assaf, 1986; Blumenthal, 1986; Carmichael et al., 1985; Findlayson, 1984b), but not others (Horner & Maddux, 1985; Milojkovic, 1984). So, this effect may be sensitive to many factors in the classroom environment that remain to be researched.

Students working in Logo environments engage in more self-directed explorations and show more pleasure at discovering phenomena (Clements & Nastasi, 1985; Clements & Nastasi, 1988). Findings regarding locus of control are mixed, but possess one consistent

and interesting pattern: Students experiencing Logo did appear to judge situations for themselves and accept responsibility for their actions (Blumenthal, 1986; Horner & Maddux, 1985). Recent studies indicate that Logo may enhance internal locus of control for preschool children (Bernhard & Siegel, 1994). Similarly, Logo can increase mastery-oriented thinking and a belief that one can work to become more intelligent (Burns & Hagerman, 1989).

In summary, research suggests that (a) educators build classroom cultures that encourage students to take responsibility for their own learning; (b) to engage in tasks that are challenging, but not too difficult or too easy; and (c) to work cooperatively, asking each other questions (King, 1989), engaging in cognitive conflicts, and always working to resolve them through discussion of ideas and negotiation (Clements & Nastasi, 1988; Hoyles et al., 1991). This should not be taken to mean that children should always work together, however; a balance of cooperative and individual work may be ideal. A combination of structured interdependence and individual autonomy, with a high-status student coordinator, may be best (Hoyles, Healy, & Pozzi, 1994).

Future research. Although this is already a comprehensive list of implications, future research on social development could go further. What is the contribution of each of these characteristics of computer programming environments? Do other computer and noncomputer activities offer similar advantages? If so, what features are critical? Similarly, computer programming may have the power to enhance students' self-esteem and attitudes toward school. However, not much is known about the mechanisms of that enhancement. Such knowledge would help address contradictions in the literature, such as those concerning locus of control. More broadly, it would help us learn how to more effectively enhance a broad range of social and emotional competencies.

INNOVATIONS AND A NEED FOR RESEARCH

Several new approaches to computer programming have created new areas in need of research. Examples are presented in the following section.

Projects With Computer Programming

One promising educational application of computer programming is the creation of projects. For example, 4th-grade students designed software to teach fractions to 3rd-graders (Harel, 1991). Students were given both the freedom and the responsibility to create their own designs. Similarly, they were responsible for learning about fractions and about ways to represent fractions so that other students could learn about them. The students were divided into three groups: the instructional design (ID) group and two comparison groups. The first comparison group was given the same amount of exposure to Logo programming as the ID group. This exposure was integrated with various curriculum topics; however, the projects tended to be short and assigned by the teacher. The second comparison group received Logo once a week in a computer literacy course. The ID group showed greater mastery of both Logo and fractions than the two comparison groups, because they had created a rich variety of ways to represent fractions

for a real audience. They (a) divided a circle into four regions and flashed two on and off to show two-fourths with the written text "two-fourths;" (b) showed an animated clock; and (c) showed a one-dollar bill with four quarters underneath, two of which moved and stopped near the written words "two-fourths of one dollar."

In another study, a different group of 4th-graders designed computer games to teach 3rd-graders about fractions (Kafai, 1993). Compared to other groups, students who designed games improved significantly in their knowledge of Logo programming and fractions, although fraction knowledge did not increase as much as in the Harel study. Therefore, conducting computer programming design projects has potential for learning. A good environment—one with features that support the designing—offers real advantages to this type of design activity. A useful aspect of Logo in these studies appears to be giving students control over their own representations of mathematical ideas. Various pictures, animations, and texts can be composed, selected, and combined. Future research might conjecture and test what other types of tools and affordances support rich projects. Further, these two projects have shown what intensive four months of meaningful Logo experience can do. We have yet to see an investigation of several year's use of programming (Clements & Meredith, 1993). This may represent the least realized potential of computer programming. Only after years of consistent, creative use will children truly use programming as a powerful thinking tool. A substantive contribution to educational research would be made by those willing to investigate what students can do and learn after programming computer projects throughout their educational career.

Robotics

Using LEGO-Logo, students create Lego structures, including lights, sensors, motors, gears, and pulleys, as well as Logo programs that control these structures. As with the computer programming projects previously discussed, such activity can be fruitful. For example, 4th-grader Kevin started by building a car out of LEGO (Resnick, 1988). The car moved forward a bit, but then the motor fell off and vibrated across the table. The movement interested Kevin. He wondered if he could use the vibrations to power the vehicle. He mounted a motor on a LEGO base and learned that he could control his "walker," turning it to the right when the motor rotated in one direction, left when It rotated in the other. There are only a few studies on such robotic environments. These studies suggest that such experiences can positively affect mathematical achievement (Browning, 1991; Enkenberg, 1994; Flake, 1990; Weir, 1992), although one control group showed higher gains in computation than did the LEGO-Logo group (Flake, 1990). LEGO-Logo appears to provide authentic learning tasks (Lafer & Markert, 1993), motivate and empower students as well, and possibly develop self-esteem (Silverman, 1990; Weir, 1992).

Future research is sorely needed. We have but scratched the surface of investigating what might be done in LEGO-Logo and other robotic environments in which computer languages are used to control machines.

Redesigning Computer Languages Based on Research

Turtle Math was born in previous research on educational computer programming. We abstracted five principles and designed *Turtle Math* based on these principles so as to fine-tune it for the learning of geometry and other mathematical topics (Clements & Sarama, 1995).

For example, the nature of programming creates the need to make relationships between symbols (code) and drawings explicit. Research indicates that this is a crucial advantage of programming, but also that students often lose the psychological connection between the two. There are, then, two issues for a Logo programming environment. First is the issue of immediate mode programming vs. the use of procedures. Students (especially novices) often prefer this exploratory mode and find it easier to make sense of tasks. Further, immediate mode encourages students to take a more global perspective on the task and to look for structure within their program design (Hoyles & Noss, 1987). In contrast, use of procedures can lead to a separation between symbols and drawings (Hoyles & Noss, 1988). Finally, students working in immediate mode also are more likely to abandon inappropriate solution strategies before they make incorrect generalizations, which keeps them on track. These results lead to the following features of *Turtle Math*. Students enter commands in "immediate mode" in a command window, or as procedures in a "teach" window, but usually in the former. Any change to commands in either location, once accepted, are reflected automatically in the drawing. A tool copies these into the teach window, applies a student-supplied name, and thus defines the procedure. The dynamic link between the commands in the command window and the geometry of the figure is critical. Any change in the commands leads to a corresponding change in the figure, so that the commands in the command window precisely reflect the geometry in the figure. So, the Logo code in the command window stands halfway between traditional immediate mode records and procedures created in an editor, helping link the symbols and drawing.

The second basic issue is the direction of the symbol-drawing connection. One of Logo's main strengths has been its support of linkages between drawings and symbols. One of its limitations has been in the lack of two-way connection between these modes. That is, one creates or modifies symbolic code to produce visual drawings, but not the reverse. *Turtle Math* provides two tools to support that reversal. A "draw commands" tool allows the student to use the mouse to turn and move the turtle, with corresponding Logo commands created automatically. A "change shape" tool allows students to click on a corner or side of a path and drag it to a new location. The commands in the Command window are updated automatically.

A series of classroom-based studies have indicated that *Turtle Math* as implemented does realize the potential posited in the five principles upon which it was designed and was efficacious in supporting mathematical development along the lings of current mathematics reform recommendations (Clements & Sarama, 1997a; Sarama, 1995). Students' integration of number and geometry was especially potent in the *Turtle Math* environment, which provided meaningful tasks. The geometric setting provided both

motivations and models for thinking about number and arithmetic together. The motivations included game settings and the desire to create shapes and designs. The models included length and turn as settings for building a strong sense of both numbers and operations on numbers, with meaning and labeling tools supporting such construction. Separate studies revealed positive effects on students' concepts and length (Clements et al., 1997) and angle (Clements, Battista, Sarama, & Swaminathan, 1996). In addition, the numerical aspects of the measures provided a context in which students had to attend to certain properties of geometric forms. The measure made such properties (e.g., opposite sides equal) more concrete and meaningful to the students. In addition, the change in problem situation encouraged the use of larger number units. The dynamic links between these two domains structured in the *Turtle Math* environment (e.g., a change in code automatically reflected in a corresponding change in the geometric figure) facilitated students' construction of connections between their own number and spatial schemes.

In regular, sequential, computer languages there is a single process that runs instructions one-step at a time. One of the many recent innovations in various versions is the addition of parallel programming. Also called "concurrent" or "multiprocessing," this feature allows programmers to control multiple, interacting processes. Fourth and fifth-graders used a multiprocessing Logo to control the concurrent actions of robotic machines; one of the main findings was the emergence of new types of conceptual errors, or "bugs" in students' programming due to parallel processing (Resnick, 1990).
Resnick extended this work to investigate people's thinking about decentralized systems in the context of *StarLogo*, a version of Logo that simultaneously controls hundreds or thousands of turtles. He worked with high school students on projects ranging from (a) simulations of slime mold (which, when food is scarce, stop reproducing and move toward one another, forming a cluster with tens of thousands of cells that act as a whole), (b) ants, (c) traffic jams, and (d) geometry. Students' design of *StarLogo* programs often was based on an unquestioned assumption of a "leader" or an "outside force" or "seed" for change. For instance, people would quickly assume that a radar trap might cause traffic jams, or an accident. They predicted that without such outside forces, "there was nothing," so traffic would proceed smoothly. However, traffic slowdowns and jams emerge even with no seed and no leader. Across a variety of areas, the simplest and most accurate programs did *not* use "leads or seeds." Instead, they used myriad interactions among objects or beings following a few simple rules. Centralized thinking seems a strong bias in our thinking.

Future research is needed to explain more fully the benefits of such new versions of computer programming languages. In addition, there are many other versions of computer programming languages, many with graphical interfaces. A gaping hole in the research is the lack of systematic investigations of the advantages and disadvantages of these languages for learning and doing computer programming and for learning via computer programming. We also need research on programming environments that go beyond the traditional text-based language.

LEARNING COMPUTER PROGRAMMING

Early research has identified that students employ different strategies in programming tasks. Often two categories are used, usually reflecting variations on a top-down vs. bottom-up dichotomy (Papert, Watt, diSessa, & Weir, 1979; Singh, 1992). The top-down, analytic style is often associated with more proficiency (Dytman & Wang, 1984). However, students often combine the styles (Lemerise, 1992) and some benefit from being allowed to "tinker" in a bottom-up manner at first before planning in a top-down manner (Noss, 1984). Individual differences affect such strategies (Allen, Watson, & Howard, 1993; Van Merriënboer, 1990; Yelland, 1994b).

Researchers have also identified stages students pass through in learning computer programming. For example, 1st-graders showed three phases (Kull, 1986). At the first, children practiced in immediate mode, took notes, then copied the code from the notebook onto the screen in the editor, often including the errors. They then ran the procedures, noted the bugs, and learned how to edit. They soon wrote down only the moves that worked. At the second stage, children recognized that this process was cumbersome, and began planning several moves ahead, writing them down, then checking them for accuracy in immediate mode, and finally adding them to a procedure. At the third stage, children began writing whole procedures at once. The time spent drawing and taking notes constituted an important developmental step in learning to program effectively.

Older students show three stages of learning Logo programming (Howe, 1980): (a) product oriented, in which the student attempts to produce effects without concern for the method used; (b) style-conscious, in which effort is made to program in a correct style (as defined by worksheets); and (c) creative problem solving, in which Logo was used for analytic activities, including the adaptation of other procedures and the use of plans for solving problems.

These and other (Singh, 1992) attempts to delineate stages of learning computer programming may serve as useful frameworks and provide educators with insights into children's learning. However, they are sensitive to the educational context. In addition, with few exceptions, this body of work is based on anecdotal observation and intuition. Future work, involving increased specificity and assessment reliability, needs to be done if developmental stages are to be validated and applied to teaching.

Some studies have investigated factors that predict students' ability to learn computer programming and thus may be required for that learning. Many such factors have been mentioned in the literature, including (a) mathematical ability, b) processing capacity, (c) analogical reasoning, (d) conditional reasoning, (e) procedural thinking, and (f) temporal reasoning (Pea & Kurland, 1984). However, empirical evidence on such factors and their role has been mixed. General development, such as in Piagetian theory, appears to predict the ability to learn some concepts of computer programming, although not precisely (Folk, 1973). Different abilities are required for different programming tasks (Bishop-Clark, 1995; Clements, 1985). Certain mathematical (Singh, 1992), spatial

(Easton & Watson, 1993), and problem-solving abilities appear to be involved. A review of cognitive style and personality factors concluded that success in learning computer programming can be predicted by field independence, divergent thinking, and to a lesser extent, a reflective (rather than impulsive) cognitive style and internal locus of control (Bishop-Clark, 1995).

Students prefer working with computers in a computer laboratory rather than learning about them in the classroom (Schofield, 1995). This increase in enjoyment and motivation seems due to relations between students and teachers becoming more collegial, a change consistent with the constructivist emphasis on challenging learners' existing notions in ways that will foster the development of even more viable ones. Future research needs to address the roles of each and determine whether these abilities must be taught as prerequisites (or, for example, they might be learned simultaneously with computer programming). Some factors may be culturally determined. For example, previous experience with computers predicts performance in college computer science courses (Kersteen, Linn, Clancy, & Hardyck, 1988). Males have more prior experience, especially in advanced computer science topics, than females, often gained outside of school through "hacking" and unguided exploration. It may be that an initial course should be graded pass/fail and decisions concerning who will be permitted into the major be help in abeyance until the experience level of students is somewhat equalized. Further, more educational computer experience, and new appealing learning opportunities, should be offered to girls before college.

Research in all three of these areas, strategies, stages, and requisites, needs to be updated, extended, and combined in the future. Many issues have been addressed only by a small number of studies. In addition, computer programming usually has been measured as a single activity; however, it has separate and distinct phases, such as (a) problem representation, (b) program design, (c) coding, and (d) debugging. It may be that certain cognitive styles and personality dimensions affect some phases but not others (Bishop-Clark, 1995). Further, even the directionality of the connections must be addressed. For example, while certain factors may influence how well or how quickly students learn computer programming, it might also be that computer programming affects these factors. For example, a small amount of research indicates that computer programming can cause students to become (a) more field independent, (b) more reflective, and (c) more divergent thinkers (Bishop-Clark, 1995; Clements, 1995). Longitudinal and qualitative studies will further help address such questions.

We also need qualitative and experimental studies that examine how such information might be used in teaching. Should students with different profiles be taught in different ways? Is there a way for teachers to structure activities to help ensure that dependent students achieve their highest potential (Bishop-Clark, 1995)? Such research is important, as teaching strategies are not straightforward. For example, one might assume that encouraging or "forcing"" impulsive students to be more reflective would be effective. However, the opposite has been found (Van Merriënboer, 1990). It appears that forcing students (either impulsive or reflective) to use the opposite strategy does not

allow them to apply their own cognitive style, and thus they learn less. However, more intensive cognitive style training may still be effective.

TEACHING AND TEACHING WITH COMPUTER PROGRAMMING

Research has only begun to address the complex questions of teaching computer programming as a discipline and teaching with computer programming as a means to reach other goals. Teaching in computer environments seems to differ from off-computer instruction; for example, student-teacher interactions may be more student-centered and individualized. There can be up to 17 times more individual interactions (Swan & Mitrani, 1993). Many factors affect such teaching. The amount of inquiry-based teaching, for example, is affected by the teaching style that predominates in the school (Tanner, 1992). Individual teachers pose different tasks and hold different aims for teaching computer programming (many of which are uncertain). Instructional practices influence what students learn in Pascal programming classes (Husic, Linn, & Sloane, 1989). For example, often introductory students, who learn mostly syntax, benefit from direct instruction; in comparison, AP (Advanced Placement) students learned to plan and debug complex problems most effectively with less direct guidance and more opportunities for autonomy.

Research also has affirmed that the teacher's role is critical (Dalbey & Linn, 1986; Delclos & Burns, 1993; Keller, 1990; McGill & Volet, 1997). Teacher mediation appears to involve multiple actions. Effective teachers appear to plan and oversee computer programming experiences to ensure that students reflect on and understand the mathematical concepts (McCoy, 1996, p. 443). Research indicates that they (a) select or create tasks designed to achieve educational goals, (b) focus students' attention on particular aspects of their experience, (c) educe informal language and provide formal mathematical language for the mathematical concepts, (d) emphasize planning for algorithm development, (e) suggest paths to pursue, (f) provide metacognitive prompts and asking higher-order questions, (g) facilitate disequilibrium using computer feedback as a catalyst, (h) provide tailored feedback regarding students' problem-solving efforts, (I) discuss errors and common misunderstandings, (j) continually connect the ideas developed to those embedded in other contexts, (k) provide modeling and coaching, and (l) promote both student-teacher and student-student interaction.
Future research should ascertain which aspects or combination of aspects on this rather intimidating list are necessary and sufficient. It is needed to answer related questions. How are these (effectively) instantiated in a computer-programming environment? Are they different in computer programming than in other environments? How d they interact with the environment in unique ways?

Future research also needs to build on the following additional initial findings. Learning in unstructured computer programming environments may be troubled by unreflective use of tools and avoidance of mathematical analysis (Noss & Hoyles, 1992). What versions of languages, curricula, and teaching strategies could maintain the benefits while mitigating the disadvantages? Effects may be more positive if programming is integrated into the curriculum (Clements & Meredith, 1994; Hoyles & Noss, 1987). How might that

best be done (especially considering the lack of knowledge and even resistance to such integration that is undoubtedly present in some educators; see the following section)? Delicate balances have been necessary both between teacher structuring and students explorations, and cooperative endeavors and time for solitary work (Heller, 1986). Structured curricula and activities may enable teachers with less knowledge of computer programming (Tanner, 1992). Young children appear to benefit from modifications to computer programming environments (Genishi, 1988; Watson, Lange & Brinkley, 1992). What types of environments are ideal? Equity should remain a concern; home owners of computer out perform nonowners (Nichols, 1992). What can be done to achieve and maintain equal access?

IMPLEMENTATION AND PROFESSIONAL DEVELOPMENT

Integrating computer programming into the curriculum is a challenge, especially in that many implicit beliefs and structures of teachers and schools stand in contraposition to goals of this integration (c.f., Moreira & Noss, 1995). In one study (Sarama, Clements, & Henry, 1998), both teachers and administrators initially thought the computer lab was ideal. Later, teachers realized that one or two blocks of time per week represented inadequate access, but though this was communicated, the administrators believed that schedule readjustment was adequate. As another example, administrators saw multiple simultaneous reform efforts as mutually reinforcing. However, they overwhelmed the teachers, who named ten different reforms they were implementing *that year*—reforms that they believed were separate demands. Thus, personal, emotional, and social dimensions are at least as critical as professional and cognitive dimensions. Some of these dimensions might be addressed in professional development. Those in charge of such development should recognize that change of critical beliefs and attitudes takes considerable time. It can be beneficial, but if difficulty levels are not carefully monitored, some teachers may develop more negative attitudes toward programming per se (Brownell, 1993; Moreira & Noss, 1992).

Future research faces an important challenge in determining how to address the myriad systemic factors that affect the success of implementations of computer programming curricula both in elementary schools and in professional development schools. This may be especially important if such research confirms that

Programming, especially when using LOGO as a discovery environment, allows for a diverse range of styles to manifest themselves, not only to teachers, but to the instructors of teachers. The programming assignment [in comparison to other applications, such as CAI, spreadsheets, databases, and word processing] becomes a mirror for the teachers to recognize their own style strengths and weaknesses. They are able to reflect on that information and use it as insight into how they would teach computer literacy and, more significantly, how they would deliver and design curricula in general. As an observational tool for instructors and a platform for teachers to examine their won learning, the Logo environment provides the springboard for mindful consideration and reflection of learning and teaching preferences that impact teaching practice (Howard & Howard, 1994, p. 27).

FINAL WORDS

Depending on the environment in which it is embedded, computer programming can constitute a trivial enterprise or a variegated educational experience. From an optimistic perspective, it could be claimed that few educational environments have shown consistent benefits of such a wide scope, from the development of academic knowledge and cognitive processes to the facilitation of positive social and emotional climates. Yet, somewhat paradoxically, realizing these multifarious benefits does not imply lack of focus: Integration into one or more subject matter areas maximizes positive effects. A critical factors, however, is a clear and elaborated vision of the goals of Logo experience-—shared among administrators, curriculum developers, teachers, and students. Such a vision provides a gyroscope that guides the myriad activities of educators: (a) administration, (b0 curriculum development, (c) lesson guidance, and (d) moment-by-moment interactions with students.

Research could do much more, however, to help us fully understand and realize this potential. We need to learn to operationalize and optimize the complex webs of variables that determine the effectiveness of educational computer programming. We need continuing research and development to expand our knowledge of what students and teachers learn in various computer-programming classrooms. This research should include a wide range of methodologies and assessments. Standardized tests do not measure many concepts and skills developed in Logo (Butler & Close, 1989).
Many of the conjectures and theories related to educational computer programming have been abstracted post hoc. We need studies that provide experimental tests of these hypotheses. We know far too little about how the unique features of computer programming environments interact with the goals and content of the domain and the particular features of various teaching and learning situations to promote learning and development.

We need to know if other computer environments offer more, the same, or less than computer programming environments. In a similar vein, we need to know whether aspects of computer programming should be embedded into other environments, as it is being increasingly embedded in other applications, from spreadsheets to word processing to operating systems.

Future research could also address several related and far-reaching questions. How can computer programming be successfully implemented in elementary schools and graduate schools of education in a manner consistent with systemic reform? Does the fascination of our country's desire for new technology relegate computer programming to history? That is, can we expect to have educators produce and use mathematically simple, but multimedia-enhanced word problems because they are today's bandwagon rather than mathematically richer programming environments? The creation of new version of programming languages will not mean much if such a narrow view predominates. On the other, optimistic side, the philosophy, goals, and pedagogies of computer programming in education have closely matched those in reform recommendations (National Council of Teachers of Mathematics, 1989). Ironically, educators who support those reform

movements previously called such philosophies "romantic" and "unrealistic." The next century is the time to rebuild bridges. A major issue to investigate is what students can do and learn after programming computer projects throughout their educational career. The tantalizing suggestions of long-term and delayed effects of computer programming suggest unrealized potential of such a long-rage approach (Clements & Gullo, 1984; Johnson-Gentile, Clements & Battista, 1994; Ortiz & Miller, 1991).

We need research across the spectrum. From the traditional hypothesis tests that will evaluate the many (usually post hoc) conjectures and initial results in the literature to exploratory studies that stretch the conceptions of what computer programming and education might be.

References

Allen, J., Watson, J.A., & Howard, J.R. (1993). The impact of cognitive styles on the problem solving strategies used by preschool minority children in Logo microworlds. *Journal of Computing in Childhood Education, 4,* 203-217.

Assaf, S.A. (1986). The effects of using Logo turtle graphics in teaching geometry on eighth grade students' level of thought, attitudes toward geometry and knowledge of geometry. *Dissertation Abstracts International, 46,*_2952A. (University Microfilms No. DA8512288)

Barker, W.F., Merryman, J.D., & Bracken, J. (1988, April). *Microcomputers, math CAI, Logo and mathematics education in elementary school: A pilot study*. Paper presented at the meeting of the American Educational Research Association, New Orleans.

Battista, M.T., & Clements, D.H. (1988). A case for a Logo-based elementary school geometry curriculum. *Arithmetic Teacher, 36,* 11-17.

Battista, M.T., & Clements, D.H. (1991). *Logo geometry*. Morristown, NJ: Silver Burdett & Ginn.

Bernhard, J.K., & Siegel, L.S. (1994). Increasing internal locus of control for a disadvantaged group: A computer intervention. *Computers in the Schools, 11*(1), 59-77.

Billings, L.J., Jr. (1986). Development of mathematical task persistence and problem-solving ability in fifth and sixth grade students through the use of Logo and heuristic methodologies. *Dissertation Abstracts International, 47,* 2433A.

Bishop-Clark, C. (1995). Cognitive style, personality, and computer programming. *Computers in Human Behavior, 11*(2), 241-260.

Blumenthal, W. (1986). *The effects of computer instruction on low achieving children's academic self-beliefs and performance*. Unpublished doctoral dissertation, Nova University, Fort Lauderdale, FL.

Bowman, B.T. (1985, November). *Computers and young children*. Paper presented at the meeting of the National Association for the Education of Young Children, New Orleans, LA.

Brownell, G. (1993). Preservice teachers in a computer utilization in the classroom course: An overview of four studies. In N. Estes & M. Thomas (Eds.), *Rethinking the roles of technology in education* (pp. 143-145). Cambridge, MA: MIT.

Browning, C.A. (1991). Reflections on using Lego®TC Logo in an elementary classroom. In E. Calabrese (Ed.), *Proceedings of the Third European Logo Conference* (pp. 173-185). Parma, Italy: Associazione Scuola e Informatica.

Bruggeman, J.G. (1986). The effects of modeling and inspection methods upon problem solving in a computer programming course. *Dissertation Abstracts International, 47,* 1821A. (University Microfilms No. DA8619363)

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.

Butler, D., & Close, S. (1989). Assessing the benefits of a Logo problem-solving course. *Irish Educational Studies, 8,* 168-190.

Campbell, P.F. (1987). *Measuring distance: Children's use of number and unit. Final report submitted to the National Institute of Mental Health Under the ADAMHA Small Grant Award Program. Grant No. MSMA 1 R03 MH423425-01.* University of Maryland, College Park.

Carmichael, H.W., Burnett, J.D., Higginson, W.C., Moore, B.G., & Pollard, P.J. (1985). *Computers, children and classrooms: A multisite evaluation of the creative use of microcomputers by elementary school children.* Toronto, Ontario, Canada: Ministry of Education.

Choi, W.S., & Repman, J. (1993). Effects of Pascal and FORTRAN programming on the problem-solving abilities of college students. *Journal of Research on Computing Education, 25*(3), 290-302.

Clements, D.H. (1983-84). Supporting young children's Logo programming. *The Computing Teacher, 11*(5), 24-30.

Clements, D.H. (1985). Research on Logo in education: Is the turtle slow but steady, or not even in the race? *Computers in the Schools, 2,* 55-71.

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). 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. (1991). Enhancement of creativity in computer environments. *American Educational Research Journal, 28,* 173-187.

Clements, D.H. (1995). Teaching creativity with computers. *Educational Psychology Review, 7*(2), 141-161.

Clements, D.H., & Battista, M.T. (1988, November). *The development of geometric conceptualizations in Logo.* Paper presented at the meeting of the International Group for the Psychology in Mathematics Education—North American Chapter, DeKalb, IL.

Clements, D.H., & Battista, M.T. (1989). Learning of geometric concepts in a Logo environment. *Journal for Research in Mathematics Education, 20,* 450-467.

Clements, D.H., & Battista, M.T. (1992a). *The development of a Logo-based elementary school geometry curriculum (Final Report: NSF Grant No.: MDR-8651668).*

Buffalo, NY/Kent, OH: State University of New York at Buffalo/Kent State University.

Clements, D.H., & Battista, M.T. (1992b). Geometry and spatial reasoning. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching and learning,* (pp. 420-464). New York: Macmillan.

Clements, D.H., Battista, M.T., Sarama, J., Swaminathan, S., & McMillen, S. (1997). Students' development of length measurement concepts in a Logo-based unit on geometric paths. *Journal for Research in Mathematics Education, 28*(1), 70-95.

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., & Meredith, J.S. (1993). Research on Logo: Effects and efficacy. *Journal of Computing in Childhood Education, 4,* 263-290.

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.

Clements, D.H., & Nastasi, B.K. (1992a). Computers and early childhood education. In M. Gettinger, S.N. Elliott, & T.R. Kratochwill (Eds.), *Advances in school psychology: Preschool and early childhood treatment directions* (pp. 187-246). Hillsdale, NJ: Lawrence Erlbaum.

Clements, D.H., & Nastasi, B.K. (1992b). The role of social interaction in the development of higher-order thinking in Logo environments. In E. De Corte, M. C. Linn, H. Mandl, & L. Verschaffel (Eds.), *Computer-based learning environments and problem solving* (pp. 229-248). Berlin-Heidelberg-New York: Springer-Verlag.

Clements, D.H., & Sarama, J. (1995). Design of a Logo environment for elementary geometry. *Journal of Mathematical Behavior, 14,* 381-398.

Clements, D.H., & Sarama, J. (1997a). Children's mathematical reasoning with the turtle metaphor. In L.D. English (Ed.), *Mathematical reasoning: Analogies, metaphors, and images* (pp. 313-337). Hillsdale, NJ: Lawrence Erlbaum.

Clements, D.H., & Sarama, J. (1997b). Research on Logo: A decade of progress. *Computers in the Schools, 14*(1-2), 9-46.

Dalbey, J., & Linn, M. (1986). Cognitive consequences of programming: Augmentations to BASIC instruction. *Journal of Educational Computing Research, 2,* 75-93.

Dalton, D.W. (1985). *A comparison of the effects of Logo and problem-solving strategy instruction on learner achievement, attitude, and problem-solving skills.* Unpublished doctoral dissertation, University of Colorado.

De Corte, E., & Verschaffel, L. (1989). Logo: A vehicle for thinking. IN B. Greer & G. Mulhern (Eds.), *New directions in mathematics education* (pp. 63-81). London/New York: Routledge.

Delclos, V.R., & Burns, S. (1993). Mediational elements in computer programming instruction: An exploratory study. *Journal of Computing in Childhood Education, 4,* 137-152.

du Boulay, B. (1986). Part II: Logo confessions. In R. Lawler, B. du Boulay, M. Hughes, & H. Macleod (Eds.), *Cognition and computers: Studies in learning* (pp. 81-178). Chichester, England: Ellis Horwood.

Dvarskas, D.P. (1984). The effects of introductory computer programming lessons on the learners' ability to analyze mathematical word problems. *Dissertation Abstracts International, 44,* 2665A. (University Microfilms No. DA8400949)

Dytman, J.A., & Wang, M.C. (1984, April). *Elementary school children's accuracy and strategy use in problem solving*. Paper presented at the meeting of the American Educational Research Association, New Orleans, LA.

Easton, C.E., & Watson, J.A. (1993). Spatial strategy use during Logo mastery: The impact of cognitive style and developmental level. *Journal of Computing in Childhood Education, 4,* 77-96.

Enkenberg, J. (1994). Situated programming in a LEGOLogo environment. *Computers and Education, 22*(1-2), 119-28.

Findlayson, H.M. (1984a). *The transfer of mathematical problem solving skills from Logo experience. D.A.I. Research Paper No. 238.* Unpublished manuscript, University of Edinburgh, Edinburgh, Scotland.

Findlayson, H.M. (1984b, September). *What do children learn through using Logo? D.A.I. Research Paper No. 237.* Paper presented at the meeting of the British Logo Users Group Conference, Loughborough, U.K.

Fire Dog, P. (1985). Exciting effects of Logo in an urban public school system. *Educational Leadership, 43,* 45-47.

Flake, J.L. (1990). An exploratory study of Lego Logo. *Journal of Computing in Childhood Education, 1*(3), 15-22.

Folk, M.J. (1973). Influences of developmental level on a child's ability to learn concepts of computer programming. *Dissertation Abstractions International, 34,* 1125A. (University Microfilms No. 73-19, 806)

Frazier, M.K. (1987). *The effects of Logo on angle estimation skills of 7th graders*. Unpublished master's thesis, Wichita State University.

Genishi, C. (1988). Kindergartners and computers: A case study of six children. *The Elementary School Journal, 89,* 184-201.

Genishi, C., McCollum, P., & Strand, E.B. (1985). Research currents: The interactional richness of children's computer use. *Language Arts, 62*(5), 526-532.

Hamada, R.M. (1987). The relationship between learning Logo and proficiency in mathematics. *Dissertation Abstracts International, 47,* 2510-A.

Harel, I. (1991). *Children designers*. Norwood, NJ: Ablex.

Healy, L., Pozzi, S., & Hoyles, C. (1995). Making sense of groups, computers, and mathematics. *Cognition and Instruction, 13*(4), 505-523.

Heller, R.S. (1986). Different Logo teaching styles: Do they really matter. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 117-127). Norwood, NJ: Ablex.

Hiebert, J., & Carpenter, T.P. (1992). Learning and teaching with understanding. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching and learning* (pp. 65-97). New York, Macmillan.

Hillel, J., & Kieran, C. (1988). Schemas used by 12-year-olds in solving selected turtle geometry tasks. *Recherches en Didactique des Mathématiques, 8/1.2,* 61-103.

Hillel, J., & Samurçay, R. (1985). *Analysis of a Logo environment for learning the concept of procedures with variable*. Unpublished manuscript, Concordia University, Montreal.

Horner, C.M., & Maddux, C.D. (1985). The effects of Logo on attributions toward success. *Computers in the Schools, 2*(2-3), 45-54.

Horton, J., & Ryba, K. (1986). Assessing learning with Logo: A pilot study. *The computing Teacher, 14*(1), 24-28.

Howard, D.C.P., & Howard, P.A. (1994). Learning technology: Implications for practice. *Journal of Technology and Teacher Education, 2*(1), 17-28.

Howe, J.A.M. (1980). Developmental stages in learning to program. In F. Klix & J. Hoffmann (Eds.), *Cognition and memory: Interdisciplinary research of human memory activities*. Amsterdam, NY: North-Holland.

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., Healy, L., & Pozzi, S. (1994). Groupwork with computers: An overview of findings. *Journal of Computer Assisted Learning, 10,* 202-215.

Hoyles, C., Healy, L., & Sutherland, R. (1991). Patterns of discussion between pupil pairs in computer and non-computer environments. *Journal of Computer Assisted Learning, 7,* 210-228.

Hoyles, C., & Noss, R. (1988). Formalising intuitive descriptions in a parallelogram Logo microworld. In A. Borbás (Ed.), *Proceedings of the 12th Annual Conference of the International Group for the Psychology of Mathematics Education* (pp. 417-424). Veszprem, Hungary: International Group for the Psychology of Mathematics Education.

Hoyles, C., & Noss, R. (1989). The computer as a catalyst in children's proportion strategies. *Journal of Mathematical Behavior, 8,* 53-75.

Hughes, M., & Greenhough, P. (1995). Feedback, adult intervention, and peer collaboration in Initial LOGO learning. *Cognition and Instruction, 13*(4), 525-539.

Hughes, M., & Macleod, H. (1986). Part II: Using Logo with very young children. In R. Lawler, B. du Boulay, M. Hughes, & H. Macleod (Eds.), *Cognition and computers: Studies in learning* (pp. 179-219). Chichester, England: Ellis Harwood.

Husic, F.T., Linn, M.C., & Sloane, K.D. (1989). Adapting instruction to the cognitive demands of learning to program. *Journal Educational Psychology, 81*, 570-583.

Johnson, P.A. (1986). *Effects of computer-assisted instruction compared to teacher-directed instruction on comprehension of abstract concepts by the deaf.* Unpublished doctoral dissertation, Northern Illinois University.

Johnson-Gentile, K., Clements, D.H., & Battista, M.T. (1994). The effects of computer and noncomputer environments on students' conceptualizations of geometric motions. *Journal of Educational Computing Research, 11,* 121-140.

Kafai, Y.B. (1993). *Minds in play: Computer game design as a context for children's learning*. Unpublished Doctoral dissertation, Harvard University.

Keller, J.K. (1990). Characteristics of Logo instruction promoting transfer of learning: A research review. *Journal of Research on Computing in Education, 23,* 55-71.

Kersteen, Z.A., Linn, M.C., Clancy, M., & Hardyck, C. (1988). Previous experience and the learning of computer programming: The computer helps those who help themselves. *Journal of Educational Computing Research, 4,* 321-333.

Kieran, C. (1986). Logo and the notion of angle among fourth and sixth grade children. In *Proceedings of Psychology in Mathematics Education 10* (pp. 99-104). London, England: City University.

Kieran, C., & Hillel, J. (1990). "It's tough when you have to make the triangles angles": Insights from a computer-based geometry environment. *Journal of Mathematical Behavior, 9,* 99-127.

King, A. (1989). Verbal interaction and problem-solving within computer-assisted cooperative learning groups. *Journal of Educational Computing Research, 5,* 1-15.

Kull, J.A. (1986). Learning and Logo. In P.F. Campbell & G.G. Fein (Eds.), *Young children and microcomputers* (pp. 103-130). Englewood Cliffs, NJ: Prentice-Hall.

Kynigos, C. (1992). The turtle metaphor as a tool for children's geometry. In C. Hoyles & R. Noss (Eds.), *Learning mathematics and Logo* (pp. 97-126). Cambridge, MA: MIT.

Lafer, S., & Markert, A. (1994). Authentic learning situations and the potential of Lego TC Logo. *Computers in the Schools, 11*(1), 79-94.

Lehrer, R., Guckenberg, T., & Lee, O. (1988). Comparative study of the cognitive consequences of inquiry-based Logo instruction. *Journal of Educational Psychology, 80,* 543-553.

Lehrer, R., Harckham, L.D., Archer, P., & Pruzek, R.M. (1986). Microcomputer-based instruction in special education. *Journal of Educational Computing Research, 2,* 337-355.

Lehrer, R., Randle, L., & Sancilio, L. (1989). Learning pre-proof geometry with Logo. *Cognition and Instruction, 6,* 159-184.

Lehrer, R., & Smith, P. (1986a, April). *Logo learning: Is more better?* Paper presented at the meeting of the American Educational Research Association, San Francisco.

Lehrer, R., & Smith, P. (1986b, April). *Logo learning: Are two heads better than one?* Paper presented at the meeting of the American Educational Research Association, San Francisco, CA.

Lemerise, T. (1992). On intra- and interindividual differences in children's learning styles. In C. Hoyles & R. Noss (Eds.), *Learning mathematics and Logo* (pp. 191-221). Cambridge, MA: MIT.

LeWinter, B.W. (1986). A study of the influence of Logo on locus of control, attitudes toward mathematics, and problem-solving ability in children in grades 4, 5, 6. *Dissertation Abstracts International, 47,* 1640A. (University Microfilms No. DA8616959)

Liao, Y.K., & Bright, G.W. (1989). Computer programming and problem solving abilities: A meta-analysis. In W. C. Ryan (Ed.), *Proceedings of the National Educational Computing Conference* (pp. 10-17). Eugene, OR: International Council on Computers for Education.

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. Mayer (Ed.), *Teaching and learning computer programming: Multiple research perspectives* (pp. 111-135). Hillsdale, NJ: Erlbaum.

Liu, M. (1997). The effects of HyperCard programming on teacher education students' problem-solving ability and computer anxiety. *Journal of Research on Computing in Education, 29,* 248-262.

McCoy, L.P. (1996). Computer-based mathematics learning. *Journal of Research on Computing in Education, 28,* 438-460.

McGill, T.J., & Volet, S.E. (1997). A conceptual framework for analyzing students' knowledge of programming. *Journal of Research on Computing in Education, 29,* 276-297.

Michayluk, J.O., Saklofske, D.H., & Yackulic, R.A. (1984, June). *Logo*. Paper presented at the meeting of the CAP Convention, Ottawa, Ontario.

Miller, R.B., Kelly, G.N., & Kelly, J.T. (1988). Effects of Logo computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology, 13,* 348-357.

Milner, S. (1973, February). *The effects of computer programming on performance in mathematics*. Paper presented at the meeting of the American Educational Research Association, New Orleans, LA. (ERIC Document Reproduction Service No. ED 076 391)

Milojkovic, J.D. (1984). Children learning computer programming: Cognitive and motivational consequences. *Dissertation Abstracts International, 45,* 385B (University Microfilms No. 84-08330)

Mitterer, J., & Rose-Drasnor, L. (1986). LOGO and the transfer of problem solving: An empirical test. *The Alberta Journal of Educational Research, 32,* 176-194.

Moreira, C., & Noss, R. (1992). The teacher's view of Logomaths. In S. Dawson & R. Zazkis (Eds.), *Proceedings of the Six International Conference for Logo and Mathematics Education* (pp. 27-50). Vancouver, B.C., Canada: Simon Fraser University.

Moreira, C., & Noss, R. (1995). Understanding teachers' attitudes to change in a Logo mathematics environment. *Educational Studies in Mathematics, 28*(2), 155-76.

Nastasi, B.K., Clements, D.H., & Battista, M.T. (1990). Social-cognitive interactions, motivation, and cognitive growth in Logo programming and CAI problem-solving environments. *Journal of Educational Psychology, 82,* 150-158.

National Council of Teachers of Mathematics. (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: Author.

Nelson, G.T. (1986). Development of fourth-graders' concept of literal symbols through computer-oriented problem-solving activities. *Dissertation Abstracts International, 47,* 2607A. (University Microfilms No. DA8526359)

Nichols, L.M. (1992). The influence of student computer-ownership and in-home use on achievement in an elementary school computer programming curriculum. *Journal of Educational Computing Research, 8*(4), 407-21.

Noss, R. (1984). *Children learning Logo programming: Interim report No. 2 of the Chiltern Logo Project*. Hatfield, England: Advisory Unit for Computer Based Education.

Noss, R., & Hoyles, C. (1992). Afterword: Looking back and looking forward. In C. Hoyles & R. Noss (Eds.), *Learning mathematics and Logo* (pp. 427-268). Cambridge, MA: MIT.

Olive, J., Lankenau, C.A., & Scally, S.P. (1986). *Teaching and understanding geometric relationships through Logo: Phase II. Interim Report: The Atlanta-Emory Logo Project.* Atlanta, GA: Emory University.

Oprea, J.M. (1988). Computer programming and mathematical thinking. *Journal of Mathematical Behavior, 7,* 175-190.

Ortiz, E., & Miller, D. (1991, April). *A Logo vs. a textbook approach in teaching the concept of variable*. Paper presented at the meeting of the National Council of Teachers of Mathematics, New Orleans, LA.

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

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). *Final report of the Brookline Logo Project. Part II: Project summary and data analysis (Logo Memo No. 53).* Cambridge, MA: MIT, Artificial Intelligence Laboratory.

Pea, R.D., & Kurland, D.M. (1984). On the cognitive and educational benefits of teaching children programming: A critical look. *New Ideas in Psychology, 2,* 137-168.

Piaget, J., & Inhelder, B. (1967). *The child's conception of space*. New York: W. W. Norton.

Plourde, R.R. (1987, December). The insignificance of Logo—Stop "mucking around" with computers. *Mirco-scope* 30-31.

Reed, W.M., Palumbo, D.B., & Stolar, A.L. (1988). The comparative effects of BASIC and Logo instruction on problem-solving skills. *Computers in the Schools, 4,* 105-118.

Reimer, G. (1985). Effects of a Logo computer programming experience on readiness for first grade, creativity, and self concept. "A pilot study in kindergarten." *AEDS Monitor, 23*(7-8), 8-12.

Resnick, M. (1988). LEGO, Logo, and design. *Children's Environments Quarterly, 5*(4), 14-18.

Resnick, M. (1990). Multilogo: A study of children and concurrent programming. *Interactive learning environments, 1*(3), 153-170.

Rieber, L.P. (1987, February). Logo and its promise: A research report. *Educational Technology,* 12-16.

Robinson, M.A., Gilley, W.F., & Uhlig, G.E. (1988). The effects of guided discovery Logo on SAT performance of first grade students. *Education, 109,* 226-230.

Robinson, M.A., & Uhlig, G.E. (1988). The effects of guided discovery Logo instruction on mathematical readiness and visual motor development in first grade students. *Journal of Human Behavior and Learning, 5,* 1-13.

Roblyer, M.D., Castine, W.H., & King, F.J. (1988). *Assessing the impact of computer-based instruction: A review of recent research.* New York: Haworth.

Salem, J.R. (1989). Using Logo and BASIC to teach mathematics to fifth and sixth graders. *Dissertation Abstractions International, 50,* 1608A. (University Microfilms No. DA8914935)

Sarama, J. (1995). *Redesigning Logo: The turtle metaphor in mathematics education.* Unpublished Doctoral Dissertation, State University of New York at Buffalo.
Sarama, J., Clements, D., & Henry, J.J. (1998). Network of influences in an implementation of a mathematics curriculum innovation. *International Journal of Computers for Mathematical Learning, 3,* 113-148.
Schofield, J.W. (1995). Computers and classroom culture. Cambridge, MA: Cambridge University.
Silverman, N.S. (1990). *Logo and underachievers.* Unpublished Masters thesis, University of the Virgin Islands.
Singh, J.K. (1992). Cognitive effects of programming in Logo: A review of literature and synthesis of strategies for research. *Journal of Research on Computing in Education, 25*(1), 88-104.
Soloway, E., Lochhead, J., & Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R. J. Seidel, R. E. Anderson, & B. Hunter (Eds.), *Computer literacy* (pp. 171-185). New York: Academic.
St. Paul Public Schools. (1985). *Logo Studies.* St. Paul, MN: Author.
Studyvin, D., & Moninger, M. (1986, July). *Logo as an enhancement to critical thinking.* Paper presented at the meeting of the Logo 86 Conference, Cambridge, MA.
Sutherland, R. (1987). *What are the links between variable in Logo and variable in algebra?* Unpublished manuscript, University of London Institute of Education, London, England.
Swan, K. (1991). Programming objects to think with: Logo and the teaching and learning of problem solving. *Journal of Educational Computing Research, 7*(1), 89-112.
Swan, K., & Black, J.B. (1989). *Logo programming, problem solving, and knowledge-based instruction.* Unpublished manuscript, University of Albany, Albany, NY.
Swan, K., & Mitrani, M. (1993). The changing nature of teaching and learning in computer-based classrooms. *Journal of Research on Computing in Education, 26*(1), 40-54.
Tan, L.E. (1985). Computers in pre-school education. *Early Child Development and Care, 19,* 319-336.
Tanner, H. (1992). Developing the use of IT within mathematics through action research. *Computers and Education, 18*(1-3), 143-48.
Tracy, D.M., & Williams, M.A. (1990). Fifth and sixth grade German students learn turtle Logo: A pilot study. *Journal of Computing in Childhood Education, 1*(4), 55-66.
Vaidya, S., & McKeeby, J. (1984, September). Computer turtle graphics: Do they affect children's thought processes? *Educational Technology, 24,* 46-47.
Van Hiele, P.M. (1986). *Structure and insight.* Orlando, FL: Academic.
Van Merriënboer, J. J. G. (1990). Instructional strategies for teaching computer programming: Interactions with the cognitive style reflection-impulsivity. *Journal of Research on Computing in Education, 23,* 45-53.
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.
Weir, S. (1992). LEGO-Logo: A vehicle for learning. In C. Hoyles & R. Noss (Eds.), *Learning mathematics and Logo* (pp. 165-190). Cambridge, MA: MIT.
Wiburg, K.M. (1987). *The effect of different computer-based learning environments on fourth grade students' cognitive abilities*. Unpublished doctoral dissertation, United States International University.
Wiburg, K.M. (1989). Does programming deserve a place in the school curriculum? *The Computing Teacher, 17*(2), 8-11.
Wilson, D., & Lavelle, S. (1992). Effects of Logo and computer-asided instruction on arithmetical ability among 7- and 8-year-old Zimbabwean children. *Journal of Computing in Childhood Education, 3*(1), 85-91.
Yelland, N. (1994a). A case study of six children learning with Logo. *Gender and Education, 6,* 19-33.
Yelland, N. (1994b). The strategies and interactions of young children in Logo tasks. Journal of Computer Assisted Learning, 10, 33-49.
Yelland, N. (1995). Mindstorms or a storm in a teacup? A review of research with Logo. *International Journal of Mathematics Education, Science, and Technology, 26*(6), 853-869.
Yusuf, M. M. (1994, April). *Mathematics instruction with Logo tutorials and activities*. Paper presented at the meeting of the American Educational Research Association, New Orleans, LA.

Note

This paper was funded in part by the National Science Foundation, grant numbers MDR-8954664, "An Investigation of the Development of Elementary Children's Geometric Thinking in Computer and Noncomputer Environments," and ESI-9730804, "Building Blocks—Foundations for Mathematical Thinking, Pre-Kindergarten to Grade 2: Research-based Materials Development." Opinions expressed are those of the authors and not necessarily those of the Foundation.