ED 258 548 IR 011 699

AUTHOR Zelman, Susan
TITLE Individual Differences and the Computer Learning Environment: Motivational Constraints to Learning Logo.
PUB DATE Apr 85
NOTE 24p.; Paper presented at the Annual Meeting of the American Educational Research Association (Chicago, IL, March 31-April 4, 1985).
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Cognitive Style; Comparative Analysis; *Individual Differences; Interviews; *Learning Motivation; Problem Solving; *Programing; Questionnaires; Research Methodology; Research Needs; Secondary Education; Self Esteem; Student Attitudes; Success; Teaching Methods; Teaching Styles
IDENTIFIERS *Instructional Effectiveness; *LOGO Programing Language

ABSTRACT

This study investigated whether secondary school students of varied backgrounds, enrolled at a private school developed for the purpose of integrating computers into all aspects of the curriculum and cultivating self-motivated learners, impose their existing motivational sets on computer programming, and whether instructional settings and practices change motivational orientations over time. Using Dweck's construct of the achievement motivational process, entity and incremental students were identified, and the different approaches of the two groups to LOGO programming assignments were examined. Entity children, who made up the majority of the students, believe that intelligence is a stable, global trait judgeable by other people and that effort is risky because it might reveal low intelligence; they preferred tasks which maximized looking smart or avoiding failure. Low-confidence entity students were confused and resorted to unconstructive problem-solving behavior. For their population, the LOGO programming language could not foster self-regulatory behavior. The one incremental student, who viewed intelligence as skills expandable through one's own efforts, appeared comfortable and confident in the LOGO learning environment. Despite the school's philosophy, its LOGO programming orientation, and teachers' incremental beliefs, entity learners did not become incremental learners over time. Exposure to LOGO through an inductive teaching method was inappropriate to these students' motivational orientations. A 23-item reference list is provided. (MBR)

IR

ED258548

Individual Differences and the Computer Learning Environment: Motivational Constraints to Learning Logo

Paper Presented at the
American Educational Research Association
Annual Meeting
Chicago, Illinois
April, 1985

Dr. Susan Zelman
Chair, Department of Education
Emmanuel College

Visiting Scholar
Educational Technology Center
Harvard Graduate School of Education

IR011699

Introduction

Logo, a structured programming language for children, is taught in many elementary and high schools throughout the nation. Seymour Papert in his book Mindstorms: Children, Computers and Powerful Ideas claims that this programming language will change students' attitudes about learning and their perceptions of themselves as learners. In Logo "the child programs the computer..." and ..."acquires a sense of mastery" over the computer rather than the computer controlling the child (Papert, 1980, p. 5). As Piaget, Papert sees children as "...builders of their own intellectual structures." He advocates an inductive-learning approach where students are expected to construct understandings for themselves through direct interaction with the computer. This process of instruction will allow students to become their own psychologists and epistemologists. Using the computer as "...an object to think with" Papert hypothesizes that we will become "...less-intimidated by our fear of being wrong" (Papert, 1980, p. 23).

Yet, little systematic research has been conducted to see if students' programming in Logo changes their attitudes toward learning or their conceptions of themselves as learners. Roy Pea studied the cognitive outcomes of Logo and concluded that it was unclear whether Logo programming had an effect on planning and knowledge of computer programming concepts. His data contained high standard deviations in planning and programming skill measures (Pea, 1983). While Pea's studies examined the cognitive factors that influenced Logo programming, this study looks at certain motivational factors that may influence Logo programming outcomes. Specifically, do children impose their existing motivational sets (theories of intelligence) on the computer programming process? and, do instructional settings and practices change motivational orientations over time?

Theoretical framework

The theoretical framework of this study is developed from the work of Carol Dweck, a motivational psychologist at the Harvard Graduate School of Education. Dweck is concerned with why some children of equal abilities fail while others succeed in intense problem-solving situations. Dweck and her colleagues (Bandura & Dweck, 1981; Dweck and Elliott, 1983; Dweck and Bempechat, 1983) propose a model of children's view of their own intelligence for understanding children's achievement-related cognitions, affect and behavior. This model was developed from research on "learned helplessness" (Seligman, 1975; Abramson, Seligman, and Teasdale, 1978; Dweck, 1975; Diener and Dweck, 1978; 1980; Dweck, Davidson, Nelson and Enna, 1978; Dweck, Goetz and Strauss, 1980; Dweck and Licht, 1980) and causal attribution theory (Weiner, 1974, 1979; Weiner and Litman-Adizes, 1980).

Dweck and her colleagues suggest that children hold two different types of beliefs about intelligence. They are either incremental theorists who view intelligence as a repertoire of skills which is expandable through one's own effort or entity theorists who believe that intelligence is a rather stable and global trait judgeable by other people. For entity theorists effort is risky because it might reveal low intelligence. The type of theory that an individual carries with him or her becomes very salient in difficult problem-solving situations where there may be repeated failure. In these situations, the incremental children do not emphasize failure but focus on finding better strategies, while entity children demonstrate helpless patterns which interfere with achievements such as abandoning solution-oriented monitoring. Under certain experimental conditions, when contingencies of evaluative feedback were regulated Dweck (Dweck, Davidson, Nelson, Enna, 1978; Elliott and Dweck, 1981) was able to change the behaviors of helpless children.

This research attempts to examine how children's theories of intelligence effect their computer programming activities in actual classroom settings. Using Dweck's construct of achievement motivational processes, the study will describe how entity and incremental children approach logo programming assignments differently. The study will also examine how the role of teacher and the school culture affect the achievement motivational processes of students over time.

Utility of Dweck's Theory

Dweck's model of children's theories of intelligence is useful in describing individual differences in students' beliefs and experiences in programming for a variety of reasons. First, programming in Logo is an incremental process. According to Papert, exposing students to Logo programming should allow them to develop powerful ideas in "mind-size bites" (Papert, 1980). In this programming language students can designate previously generated procedures as commands that can be used as primitives in more complex procedures. Second, programming skills require problem de[illegible]position (breaking down the problem into its logical component parts) and debugging (systematic efforts to eliminate errors in a program to obtain a desired outcome). Dweck and her colleagues (Diener and Dweck, 1978; 1980) found that when students are oriented toward effort-producing patterns they often engage in strategy analysis and plan new strategies in light of these endeavors. In addition to Dweck's work studies which rep[illegible]t su[illegible]cessful attributes of computer trainees include persistence against obstacles and ability to be self-starters as important personality characteristics (Johns, 1984).

Examining the relationship between children's theories of intelligence and programming behaviors seems to provide an appropriate metaphor for analysis. The term "artificial intelligence" expresses the idea that the computer is created in the image of human intelligence. One could hypothesize that students' beliefs about intelligence parallels their views of computer capabilities. If intelligence is viewed as self-initiated and controlled by effort, children may compare human intelligence and computer operations in the same vein.

Methodology

This research examines how students' theories of intelligence affect how they learn to program in Logo in an alternative high school setting. The study describes how entity and incremental students approach Logo programming tasks and assignments differently. The paper also describes how the teachers and school culture affected students' achievement motivational processes over time. This study is exploratory and descriptive in nature. Hypotheses that are generated are being systematically tested in ongoing research.

During the 1983-84 academic year the researcher observed students for approximately 50 hours programming in Logo at a newly formed private school developed for the purpose of integrating computers into all aspects of the curriculum and cultivating self-motivated learners. Students selected for study were the entire upper school (originally 14 students from 12 to 16 years of age). The students varied in academic skills and ethnic, social, and socioeconomic backgrounds. Initially, there were four girls in the study. The researcher selected the upper school because she was interested in whether certain intervention strategies could change what some may consider are stable characteristics in adolescence (Dweck and Elliott, 1983; Berzonsky and Barclay, 1981; Harari and Covington, 1981).

Questionnaires containing measures developed by Dweck and her colleagues were administered in October and re-administered early in June, 1984. Students were classified as entity theorists if they agreed with one of the following three statements:

> There are some things you won't be good at no matter how hard you try.
> I sometimes wonder if when school gets harder I might not be smart enough.
> You can learn new things, but how smart you are stays pretty much the same.

(Dweck and her colleagues have subsequently developed a ten-item forced-choice questionnaire to differentiate entity and incremental learners. These questions are now in the process of being standardized.)

In addition, the researcher conducted one and one-half hour interviews with ten students in November, 1983 and seven students in June, 1984. The first interview addressed issues of the students' past and present academic performance and the reasons for choosing the alternative school. Student perceptions of the school's instructional and feedback practices (e.g., non-graded report cards and exams) and their satisfaction with these policies were also discussed. The interviewer also posed questions concerning the use and importance of computers in students' lives and the degree to which they valued and were interested in programming instruction in relationship to other subject matter. Another series of questions was related to psychological processes concerning programming, such as expectancies for success in Logo programming, attributions for success, the type of programming problems that students preferred (difficult vs. easy), their progress as logo programmers, characteristics of good programmers, how they handled mistakes in programming (e.g., persistence and independence), and problems that they could not solve (e.g., resilience after failure).

The second interview repeated the questions of the first, but also asked students their perceptions of the strengths and weaknesses of the school and the reasons for their choice of schooling for the 1984-1985 academic year.

In addition to the interviews and questionnaires the two programming teachers were asked to keep diaries of students' questions and teachers' responses during Logo classes.

Students were classified as either low confidence entity theorists, high confidence entity theorists or incremental theorists as a result of their answers to the taped interviews and paper and pencil questionnaires. Confidence was defined by their expectancy for success in Logo programming. While it is possible to have low confidence incremental learners, no student at this school belonged in this category. (The researcher hypothesized that low confidence incremental learners will develop programming skills more rapidly than low confidence entity theorists and will eventually take on the characteristics of incremental programmers.)

The researcher validated the students' classifications by asking the two teachers to make similar assignments. Both teachers and researcher agreed on the classifications. One student was classified as an incremental theorist, two as high confidence entity theorists, and nine as low confidence entity theorists.

Results

a. Analyses of the Paradigm

The researcher analyzed her field notes, interviews and questionnaires to develop the following paradigm (See Figure I), which describes how the motivational orientations of students affect the computer programming process.

The incremental learner was not a successful student at a suburban middle school in the Boston area. It appeared that he had a low value attached to what was taught at the public school and found the tasks required to learn the goals "boring" and "painful." For him the private school was the first school that he ever liked. His view of intelligence paralleled his view of computers. He defined computers as a set of programs designed by human beings that could be modified by them. When the researcher dropped her battery-operated tape recorder and stated that she did not have a mechanical mind for figuring out the placement of the batteries, the incremental theorists responded, "Lady, you could always reprogram your mind." He defined himself as the best programmer in class and viewed his programming skills as a function of his own efforts. "I spent a good deal of time programming. I work hard at it." He understood that making mistakes was part of the programming process and that it was useful and at times provided a unique perpsective. He enjoyed the independence that he received from the instructors, whom he also viewed as learners. He stated, "I learned about computers from the __________ Computer Project with MIT at __________ School. I didn't like the way they taught there. Sometimes teachers gave hand-outs of written programs. I hated that. They were more direct than here. Sharon and Mel give you the freedom to do what you want.... I can move at my own pace, do my own thing." He viewed knowledge as problem-solving and enjoyed classes which integrated controversial social issues with programming assignments. He enjoyed the project approach to learning, which he saw as always challenging and allowed him to increase his skills as a programmer while

Figure I

THEORIES OF INTELLIGENCE
AND THE COMPUTER LEARNING ENVIRONMENT:

Student Perceptions and Behaviors in the
Computer Learning Environment

	Incremental	Entity High Confidence	Entity Low Confidence
(1) View of Computers	An articulation of a set of programs designed by humans that can be modified by humans		Computer has a mind of its own "a fixed intelligence." Attributes human qualities to the computer
(2) View of programming skills	obtained by effort	obtained by natural ability and effort	innate ability
(3) Debugging errors	natural, useful; allows for different perspective; function of computer processing		failure; lack of ability to work with computers
(4) Desired method of instruction	Self-Directed project approach students induce-computer-programming models	Black Box -- student manipulates program with missing components to induce computer programming models	Modeling - Teacher presents conceptual model as an advanced organizer. Teacher demonstrates programming concept at computer
(5) Perception of role of teacher	resource person, guide, learner	silent partner	Judge, rewarder/punisher. Transmitter of knowledge
(6) Perception of role of student	problem-solver		Receptor of knowledge

	Incremental	Entity High Confidence	Entity Low Confidence
(7) Goals of programming assignment	Learning goal; increase competency in programming; intrinsic value of skill learned by activity; enjoyment of process		performance goal is extrinsic value of judgment obtained by completion of assignment. Focus on product which will maximize looking smart
(8) Perception of attainment of goals and process of attainment	Goal can be mastered Process challenging		Goal may not be accomplished. Process threatening
(9) Initial Questions on beginning project	How can I do it? What will I learn?		Can I do it? Will I look smart?
(10) Questions to Teacher	I tried to find the bug. I can't. Can you suggest a strategy?	How do I do it? Can I do ________? What should we do here? (Focus on teacher as partner.)	What do I do? Show me what to do? What should I do next?
(11) Ability to modify goals of project	flexible, changes goals with ease		rigid, unhappy about modifying plans
(12) Management of project	self-managed and self-directed	management by student and instructor	management by instructor
(13) Use of feedback from Program	self-monitoring self-reinforcement		views it as normative; immediate consultation with instructor or if perform/ unconstructive problem-solving strategy
(14) Teacher feedback	Gives accurate information about ability. Praises and encourages effort. Feedback can be delayed.		Gives immediate flattering information.

"having fun." He reported spending a lot of time pre-planning and thinking about his projects. He also stated that he never thought about how well other students in the class were doing. If he was puzzled, he preferred consulting the manuals first than to ask the teachers for help. The teachers reported that his questions were infrequent and focused on analysis of programming strategies. He engaged in self-monitoring of projects. If he felt the project was too difficult or could not be completed on time, he changed the goals of the project. He appeared confident that he would return to his original goals someday. He stated that he talked to himself when he programmed. He told himself when he was doing a good job and when he should "back off." He loved the school because he did not get "meaningless low grades" but rather "they tell you what's good or bad about your work."

On first glance, the two male high-confidence entity theorists resemble other incremental theorists; they are good programmers and have developed interesting Logo projects. My eight year old daughter commented about the three boys at a pot luck dinner at the school. "Mommy, look at those two [entity theorist] show-offs. They think their programs are so great. That other boy [the incremental theorist] told us to bug off. I guess he wanted to finish up his homework on the computer."

Clearly, these high-confidence entity theorists were concerned with maximizing looking smart. They defined themselves as good programmers who both had "natural abilities" and "spent a lot of time" with computers. They interacted with Logo instructors considerably more than did the incremental theorist. According to the teachers their projects were managed by both students and instructors. The instructors became the "silent partners" who helped out in diagnosing, checking, or coding problems but whose input was not acknowledged and at times denied by the students who took sole credit for the final projects.

While these high-confidence students enjoyed the "black box technique" of learning new Logo procedures, they were very vulnerable when programming difficulties occurred. The researcher observed an instructor praise a high-confidence entity student for developing a particular program. The teacher asked the student to demonstrate his program to the class while she worked on a consulting project. The student modified the program for some reason, and the program did not run. The student panicked, and on the brink of tears shut off the computer and walked out of class.

The nine low-confidence entity theorists consisted of four girls and five boys. Many doubted that they would be competent in working with computers. Not only did they attribute their own abilities as fixed and stable, but they also viewed computers as non-changeable entities which were "smart" or "dumb." One female student stated, "I prefer to work on the Apple than the Adam. The Apple is smarter than the Adam or Atari." They considered that good programmers were born, not made. Another student commented, "I wish I was like the younger children at the school. They learn to program easily. I should have started when I was young or maybe they are just smarter than I." Some of these resisted putting their work into programs and preferred working in direct mode. When procedures were placed in programs, these students resisted checking or diagnosing their work. It appeared that they spent little time pre-planning this work. Even after two months of programming some failed to see debugging errors as a useful part of the programming process but rather viewed it as a lack of ability to program.

These students did not like the "black box" approach to teaching; they preferred teachers to be advance organizers demonstrating step-by-step how to program a particular assignment. Even when the students followed these recipes they were unsure whether the program would run and resisted pressing the final button.

Low confidence entity pupils could not initiate their own projects but needed teachers to devise projects for them. They wanted easy projects so that they could "look smart" or not "appear to be too stupid." After project goals were established, these students were rigid and unhappy about changing the objectives when it appeared that they could not complete the task because of ability or time constraints. Projects were managed by the instructors, and there were frequent questions to teachers, such as "What should I do next?" When teachers were not available to answer their immediate needs, some students asked peers, but others engaged in such unproductive behaviors as doing nothing, kicking the computer, turning off the computer and losing work in the workspace, and harassing other students.

These low-confidence entity students were unhappy about the lack of structure in the academic and non-academic aspects of the curriculum. They viewed the teachers as a rewarder or punisher and were angry when teachers failed to perform this role. In discussing controversial issues they preferred the catechism to situational ethics. These students wanted to be taught the rules and were appalled when teachers asked them to devise the rules (e.g., disciplinary codes) themselves. They wanted to be graded, and two girls consistently put letter grades on their own work to make themselves "feel better."

Dweck's model of children's theories of intelligence were useful to the researcher in describing the different beliefs and behaviors of adolescents' programming in Logo.

The incremental theorist preferred programming tasks that were challenging and understood that confusion and errors were part of the learning process. Entity theorists preferred tasks which maximized either looking smart or avoiding failure. When low-confidence entity theorists were confused or confronted with tasks that they could not handle, they resorted to unconstructive problem-solving behaviors. For their population, the Logo programming language could not foster self-regulatory types of behaviors.

b. Entity Learners in an Incremental Setting

The researcher posed the question whether entity theorists could become incremental learners over a period of time. This question seemed reasonable, for it appeared that the philosophy of the school, its strong Logo programming orientation, and the incremental beliefs of the teachers should translate into instructional and feedback practices which would promote incremental learning patterns.

The school was founded by mathematics, computer, and linguistic teachers who taught for several years at an upper-middle class suburban high school. Frustrated with school politics and bureaucracy they set out to develop an alternative school which would attract self-motivated learners or students who aspired to be "self-starters."

This alternative school was clearly different from a normal high school setting where tasks were usually well-defined, specific standards for promotion and assignments communicated, deadlines made explicit, and rewards and punishments understood by teachers and pupils. In this school learning tasks were often required to be developed by the students, standards for graduation were unclear to both students and the researcher, and personal standards rather than normative standards for students were encouraged. Teachers were not rewarders or punishers, but guides or resource personnel who were flexible in adjusting assignments and deadlines to meet individual needs of students. The founders of the school believed that providing a structure which simulated the "real world" would promote self-regulating motivational systems in students that would assist them in being successful adults.

The school also wanted to prepare students to deal with the new technology and to provide them with technological problem-solving skills. To meet these objectives the curriculum attempted to integrate Logo into most aspects of the school courses. There was a computer for every student in the upper school. At least 60% of the students' time was to involve either Logo programming or word-processing.

The researcher considered the teachers of the school to be incremental theorists. Dweck and her colleagues hypothesized that entity and incremental theorists employed different feedback and instructional practices for their students and that incremental teachers might be more successful in promoting effort-producing learning patterns in students than might entity teachers (Dweck and Elliott, 1983; Dweck and Bempechat, 1984). They speculated that entity-oriented teachers employed differential feedback processes and had differential expectations for their students and differential achievement goals based on their perceptions of different ability levels of children. On the other hand, incremental teachers employed similar feedback practices and had similar expectations and goals for all students. Entity teachers saw and set limits for learning; incremental teachers did not. A body of research exists that show that teachers who viewed failure in students, as an opportunity and challenge tended to be more effective in promoting desired learning outcomes (Lightfoot and Carew, 1982; Brophy, 1983).

The researcher viewed the instructors as incremental theorists because they saw their role as resource persons helping all students set individual goals to increase competency in programming. They attempted to devise or help all students develop projects which had initial errors and were confusing but challenging. The teachers provided all students with long-term tasks which required planning and persistence. They tried to provide students with coping strategies which would teach them that ambiguities, errors, and negative feedback from the computer were part of the learning process.

In addition, the researcher discussed the work of motivational psychologists with teachers at the school and urged them to implement more consciously the following strategies which were already part of their repertoire of skills:

a. <u>Teaching students self-monitoring and self-reinforcing techniques</u>
Teachers should help students not to blame his/her ability or make other attributions but rather to work on strategy analyses as to why obstacles occur (Dweck and Elliott, 1983). Teachers were encouraged to talk about when they programmed so that students could hear their inner language which explained how their teachers went about performing the task analyses, strategy planning, and self-monitoring.

b. <u>Analyses of positive outcomes</u> (Dweck and Elliott, 1983)
Teachers were encouraged to analyze with their students the reasons for successful completion of a programming task and relate the students' successes to skills and abilities which they were developing as programmers. Teachers were encouraged to compare programming skills with demonstrated skills in other domains (e.g., composition).

c. Feedback strategies
Teachers were encouraged to praise the efforts which students demonstrated in Logo programming as well as the intellectual processes (e.g., doing a task analysis of a particular problem). Teachers were urged not to praise Logo products for fear of encouraging performance evaluation.

d. Questioning strategies
Teachers were encouraged to give students time to answer questions. Teachers were requested to provide a series of "scaffolding questions" to help students do the task analysis.

e. Setting personal rather than normative standards
To encourage students to develop their own programming goals, teachers were encouraged to have students explain to the class their progress on projects in terms of the skills which they acquired and the affect which they experienced doing the project.

Although the teachers engaged in some of the above practices, in analyzing the interviews of June, 1984 it became clear that there was little teaching going on in the school that spring. Teachers who were also the administrators became preoccupied with financial and other administrative matters. Incremental teaching practices translated into extreme permissiveness and chaos. While entity students claimed that they learned about the programming process during the school year, teachers reported that many still refused to check or diagnose their work independently. Students were disappointed in their lack of progress at the school and blamed the unstructured teaching approaches or their own abilities. Only one entity student made attributions in terms of her own lack of effort. Many of the entity theorists came to the school at the insistence of their parents in the hopes of getting in on the ground floor of computers. All but two entity theorists transferred to other schools in September, 1984.

In addition, there was an obvious mismatch of teaching styles and student personalities. The entity students wanted a hierarchical teaching and learning structure; they wanted teachers to explain concepts which the teachers felt were important to learn and wanted the teaching experience to be so structured as to help them assimilate this knowledge into their cognitive structures. The instructors wanted the students to put their knowledge to work to solve a problem and to check and diagnose their work, to see knowledge as tentative, and to examine their own inquiry processes. Neither teachers nor students would compromise these positions. For example, teachers did not expose students to drill and practice programs, tutorials, or other "tools" which might have satisfied their need for more structured approaches.

It was clear that exposure to Logo through purely an inductive teaching method was inappropriate to motivational orientations of the students. Inquiry Logo teaching did not change students' beliefs about learning. In fact, they reported that the experience made them feel more insecure and wanting more structured programmed success experiences which they had become accustomed to at previous schools.

It is unclear whether the non-optimal discrepancy between the personalities of students and models of teaching and the growing chaos of the school explains the lack of change in students' beliefs or whether adolescents already formed stable characteristics which are resistant to change (Dweck and Elliott, 1983; Bezonsky and Barclay, 1981; Harari and Covington, 1981). Clearly, more systematic research needs to be done to answer these questions.

Need for Future Research

The study poses many questions for future research. To what extent can Logo programming minimize some of the students pre-existing motivational dispositions" What role can teachers play in this process? More research needs to be conducted on whether teachers' beliefs about intelligence translate into different instructional and feedback practices. Controlled experimental studies need to be designed to see if different feedback and instructional practices can change the orientation of learners over time. Are adolescents less susceptible to manipulation and change than are younger students? What combination of computer program design and teacher instructional and feedback practices can promote facilitating patterns of learning (e.g., independence, initiation of challenge, persistence, resiliency after failure).

One of the most important questions in teaching programming skills is to know when and how to intervene in fostering facilitating learning patterns. This study suggests a conceptual framework to begin to look more systematically at the motivational orientations that students bring to the programming process.

References

Bandura, M., & Dweck, C. S. Children's theories of intelligence as predictors of achievement goals. Unpublished manuscript, Harvard University, 1981.

Berzonsky, M. D., & Barclay, C. R. Formal reasoning and identity formation: A reconceptualization. In T. A. Meachim, R. Santilli (Eds.), Social Development in Youth: Structure and Content. Basel: Karger, 1981.

Brophy, T. Teacher behavior and its effect. Journal of Educational Psychology, 1979, 71, 733-750.

Brophy, T. Fostering student learning and motivation on the elementary school classroom. In S. G. Paris, G. M. Olson, & H. W. Stevenson (Eds.), Learning and Motivation in the Classroom. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

Diener, C. I. & Dweck, C. S. An analyses of learned helplessness: Continuous changes in performance, strategy, and achievement cognitions following failure. Journal of Personality and Social Psychology, 1978, 36, 451-462.

Diener, C. I. & Dweck, C. S. An analyses of learned helplessness: II. The processing of success. Journal of Personality and Social Psychology, 1980, 39, 940-952.

Dweck, C. S. The role of expectations and attributions in the alleviation of learned helplessness. Journal of Personality and Social Psychology, 1975, 31, 674-685.

Dweck, C. S., Davidson, W., Nelson, S., Enna, B. Sex differences in learned helplessness: II. The contingencies of evaluative feedback in the classroom, and III. An experimental analyses. Developmental Psychology, 1978, 14, 268-276.

Dweck, C. S., and Elliott, E. S. Achievement motivation. In P. Mussen (general editor) and E. M. Hetherington (volume editor), Carmichael's Manual of Child Psychology: Social and Personality Development. New York: Wiley, 1983.

Dweck, C. S. & Bempechat, J. Children's theories of intelligence: Consequences for learning. In S. G. Paris, G. M. Olson, H. W. Stevenson (Eds.), Learning and Motivation in the Classroom. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

Dweck, C. S. & Goetz, T. E. Attributions and learned helplessness. In J. H. Harvey, W. Ickles, & R. F. Kidd (Eds.), New Directions in Attribution Research (Vol. 2), Hillsdale, NJ: Erlbaum, 1978.

Dweck, C. S. & Licht, B. C. Learned helplessness and intellectual achievement. In J. Garber & M. E. P. Seligman (Eds.), Human Helplessness: Theory and Applications. New York: Academic Press, 1980.

Dweck, C. S., & Reppucci, N. D. Learned helplessness and reinforcement responsibility in children. Journal of Personality and Social Psychology, 1973, 25, 109-116.

Dweck, C. S. & Wortman, C. B. Learned helplessness, anxiety, and achievement motivation: Neglected parallels in cognitive, affective, and coping responses. In H. W. Krohne & L. Laux (Eds.), Achievement, Stress, and Anxiety. Washington, D. C.: Hemisphere, 1982.

Harari, O. & Covington, M. V. Reactions to achievement behavior from a teacher and student perspective: A developmental analyses. American Educational Research Journal, 1981, 18, 15-28.

Johns, R. P. (May 28, 1984) Don't judge a programmer by expertise alone. Computerworld 18.

Licht, B. G. & Dweck, C. S. Determinants of academic achievement: The interaction of children's achievement orientations in skill area. Manuscript submitted for publication, 1981.

Licht, B. G. & Dweck, C. S. Sex differences in achievement orientations: Consequences for academic choices and attainments. In M. Marland (Ed.), Sex Differentiation and Schooling. London: Heinemann, 1983.

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

Pea, R. D. & Kurland, P. M. Logo Programming and the Development of Planning Skills. (Technical Report No. 16) New York: Center for Children and Technology, Bank Street College, April 1983.

Weiner, B. Theories of Motivation: From Mechanism to Cognition. Chicago: Markham, 1972.

Weiner, B. A Theory of motivation for some classroom experiences. Journal of Educational Psychology, 1979, Vol. 71, No. 1, 3-23.

Weiner, B. (Ed.), Achievement Motivation and Attribution Theory. Morristown, NJ: General Learning Corporation, 1974.