MICROCOMPUTER LEARNING IN SMALL GROUPS: COGNITIVE REQUIREMENTS AND GROUP PROCESSES

Noreen M. Webb

CSE Report No. 200

1983

CENTER FOR THE STUDY OF EVALUATION

Graduate School of Education

University of California, Los Angeles

The project presented or reported herein was supported in part by a grant from the Spencer Foundation and by the Center for the Study of Evaluation, University of California, Los Angeles.

I wish to thank Tricia Jordan for designing the instructional materials. I wish also to thank the following people who assisted in organizing activities in the workshops, providing instruction to students, or collecting data: Professor Eva Baker, Professor Philip Ender, Wanetta Jones, Dr. Brian Stecher, Steve Shaha, Melinda Castel de Oro, Jan Sutton, and Chris Baker.

Table of Contents

Introduction

Computer scientists and educators envision future classrooms with a computer for every child (see Papert, 1981). However, it is unreasonable to expect that many school districts will be able to afford to provide one machine for every student, in the near future at least. More realistically, classrooms will have one or a few computers, making group work necessary to give all students ample contact time with the computer. Little is known about computer learning in group settings, however. Nearly all of the research on how students learn computer programming is conducted in settings in which each student works at a computer. The present paper reports the results of a study of learning computer programming in small groups. The purposes of the study were fourfold: (1) to describe the group processes operating in small groups learning computer programming, (2) to investigate the cognitive abilities, cognitive styles, and demographic characteristics that predict learning of computer programming in small group settings, (3) to determine which group process variables relate to learning of computer programming, and (4) to examine the student characteristics that relate to interaction in the group.

While most educators agree that the microcomputer setting has the potential for promoting interaction among students, there has been little systematic investigation of the kinds of interaction that occur when students work in groups with the microcomputer. Anecdotal descriptions abound in reports of reading and writing programs designed to encourage communication among students in classrooms. For example, students shared ideas when using Story Maker to write stories (Rubin, 1980, 1982; Zacchei, 1982) and shared tasks when producing publications such as class newsletters (Collins, Bruce, & Rubin, 1982). Similar anecdotal descriptions also appear in reports of computer programming in group settings. Jabs (1981), for example, reported that students divided the responsibility for reading information off the screen when working on a special set of LOGO programs in four- or five-member groups. Although these studies provide some information about how students plan and organize their work in groups, we still know little about the kinds of group behavior that have been shown in other classroom settings to relate to learning, such as specific helping behavior and the group's responses to students' questions and errors (see Webb, 1982c, 1983). Furthermore, the studies described above do not shed light on the roles that individual students play in group work with microcomputers. The present study examined the helping behavior, questions, errors, and the group's responses to questions and errors that occurred when students learned LOGO in small groups, and also investigated the behavior of individual group members as a function of their relative ability within the group, age, sex, and previous experience with computers.

Most of the research investigating predictors of computer programming has focused on programming aptitude. Scores on two widely used programming aptitude tests, the IBM Programmers Aptitude Test (PAT) and the Aptitude Assessment Battery; Programming (AABP) have been shown to relate to performance during training courses (McNamara & Hughes, 1961; Katz, 1962; Hollenbeck & McNamara, 1965; Bauer, Mehrens, & Vionhaler, 1968) and to job performance (DeNelsky & McKee, 1974). The PAT has three parts: letter series, figure series, and arithmetic reasoning.

The AABP consists of five problems that require manipulation of precisely defined symbols, logical reasoning, strict adherence to instructions, and the use of flow charts. Snow (1980) also found that performance on a diagramming test, a specific aptitude test for computer programming, related to outcome measures in a short course on BASIC. While the total test scores on the aptitude tests in the above studies correlated with programming performance (in the range of .30 to .60), it is unclear which specific abilities included in the tests relate most strongly to performance. In one of the only studies to compare the importance of multiple cognitive abilities for learning computer programming, Snow (1980) related a large number of aptitude test scores to outcomes of a 15-hour course in BASIC programming for Stanford University undergraduates. Snow reported that only two factors related to outcome measures: fluid analytic reasoning and visualization (defined by such tests as the Ravens test of nonverbal reasoning and paper folding), and a personality variable reflecting self-reported flexibility and independence in academic work. Snow noted that crystallized ability, defined by prior scholastic ability, did not relate to learning in the novel instructional situation. The present study examined similar relationships for a younger student population (upper elementary school and junior high school) and for a different computer language, LOGO. The cognitive abilities examined here include mathematics computation and reasoning, spatial ability, and nonverbal reasoning ability.

The literature relating cognitive style variables to performance on a variety of cognitive tasks suggests that field independence-dependence and holistic (Gestalt) versus analytic processing may relate to programming outcomes. Because computer programming tasks are often analytic, one would expect field independent students, who perform better on analytic tasks than do field dependent students (see Gaines, 1974; Goldman & Hudson, 1973), and students showing anlaytic processing skills to perform better than field dependent students and those who show holistic processing skills. The present study tested these hypotheses.

Because there are no empirical data on verbal interaction variables that predict learning computer programming in group settings, the hypotheses for the present study come from previous research on learning academic material in classroom settings. The three main categories of verbal interaction that seem to relate to learning in small groups include giving explanations, receiving explanations, and not receiving explanations when they are needed. Giving explanations is often positively related to achievement (Peterson & Janicki, 1979; Peterson, Janicki, & Swing, 1981; Swing & Peterson, 1982; Webb, 1980a, 1980b, 1982b in press; Webb & Kenderski, in press); receiving explanations is sometimes positively related to achievement (Webb, 1980a, 1980b, 1982b), and receiving no explanation when needed is consistently negatively related to achievement (Webb, 1980a, 1980b, 1982a, 1982b, in press; Webb & Kenderski, in press). It should be emphasized that these relationships apply to explanations but not for other kinds of help. For example, giving and receiving information other than explanations does not seem to relate to achievement. The present study attempted to replicate these relationships in the computer setting. Furthermore, since it could be argued that a disadvantage of group work is that it decreases the amount of time that any one student has at the keyboard, the present study also examined the relationship between the amount of contact time with the computer and computer programming outcomes.

Many studies of computer programming treat learning outcomes as a unitary phenomenon. Yet, as Mayer (1975, 1976, 1979, 1981) has pointed out, there are several levels of knowledge underlying computer programming. In studies of BASIC and FORTRAN computer languages, Mayer distinguished seven levels of knowledge from machine level to statements to programs, and distinguished between two types of problems involving programs: generating programs and interpreting programs already written. The present study adapted several of these components to the study of LOGO. In particular, it distinguished between knowledge of basic LOGO commands, knowledge of the correct syntax of the language, ability to interpret programs already written, and ability to generate programs. By including these components of computer programming, the present study investigated whether different profiles of abilities, student characteristics, and group processes predicted different programming outcomes.

Method

Sample

The sample consisted of 35 junior high school students. Most of the students were in grades 7, 8, and 9; their ages ranged from 11 to 14. The sample was substantially white middle to upper-middle class. There were 15 girls and 20 boys. Since the mathematics test administered in this study was a shortened version of one given to three average-ability junior high school classes (grades 7 and 8 combined) in the same city two years earlier, it was possible to assess the relative ability level of the present sample. On the 40 items included in the present test, the previous sample answered 56% of them correctly (*M* = 22.4, *SD* = 3.9). The present sample answered 66% of the items correctly (*M* = 26.2, SD = 8.8). The mean ability level in the present sample, therefore, was higher than average, but the variability within the sample was also large, indicating that some students were average or below-average in mathematics ability.

Microcomputer Setting

Instruction in microcomputer programming (here, LOGO) took place in a computer laboratory with five Apple 2-Plus microcomputers. In a week-long workshop (15-20 hours), students worked in groups of three persons, one group to a microcomputer. Therefore, the maximum size of any workshop was 15 students. Because three machines were located in one room and two machines were located in another, groups rotated across machines and rooms.

Instructional Materials

The topic of the workshop was microcomputer programming in LOGO, a laguage developed and well-suited for sophisticated computer graphics. This language was selected for several reasons. First, unlike other computer languages such as BASIC, a small amount of instruction in LOGO provides the students with powerful tools to create complex graphics. Second, since LOGO is a relatively new language and is not usually taught in public schools or learned in a home environment, there was a greater likelihood of obtaining samples with little or no knowledge of the language. Finally, exercises in LOGO programming are well-suited for the study of peer interaction.

The instruction in the workshops was based on a 14-worksheet curriculum developed for this project by an experienced teacher of computer programming who was very knowledgeable about LOGO. Since the principle underlying the instruction was guided exploration, the worksheets consisted of exercises and problems for groups to solve. The exercises ranged in complexity from directing students to try out basic commands and discover their function to writing a program to produce the picture of a field of flowers. Worksheet topics included basic commands, repeating sequences of commands, programming simple geometric shapes, combining shapes to make complex figures (for example, dog, house), writing recursive programs, incorporating random placement into pictures, changing sizes of figures within a program, and programming with words and sentences to play interactive games with the computer.

Instrumentation

Background questionnaire. During the week before students started the workshop, they completed a questionnaire asking their age, grade, and amount of previous experience in microcomputers. The questionnaire included a checklist of items about their previous experience including, for example, whether their classroom, school, or parents had a microcomputer, whether they had taken any courses in computer programming (for example, BASIC, LOGO, PILOT), and whether they had played videogames on home computers or in video arcades.

Cognitive pretests. To determine the cognitive requirements of learning computer programming, seven pretests were administered: a 40-item mathematics test composed of items measuring computational skills and mathematical reasoning, a short form of Raven's Progressive Matrices (Raven, 1958), and five tests from the ETS kit of cognitive-factor reference tests (French, Ekstrom, & Prince, 1963) including Hidden figures, Gestalt Completion, Paper Folding, Form Board, and Surface Development. Internal consistency alpha for these tests for this sample ranged from .68 to .88.

The tests represent four cognitive ability dimensions and two cognitive style dimensions. The computational skills test consists of numerical computation (addition, subtraction, multiplication, and division of whole numbers, decimals, and fractions). The mathematical reasoning test consists of numerical, algebraic, and geometric word problems. This test has a substantial verbal component. Raven's Progressive Matrices is a test of general reasoning ability and is often used as a nonverbal measure of intelligence. Paper Folding, Form Board, and Surface Development are measures of spatial ability, requiring mental rotations and translations of figures. Hidden Figures measures field independence/field dependence. Gestalt Completion measures holistic vs. analytical processing.

Because the correlations among some pretests were high, several composites were formed. The first was a mathematics composite which was the sum of the computational skills and mathematics reasoning subtests ($\underline{r} = .74$). The second composite was spatial ability, a combination of the Paper Folding, Form Board, and Surface Development tests (correlations among them ranged from .53 to .58). Because the three spatial tests had different scales, the scores in the composite were weighted by the reciprocals of their standard deviations. The mathematics and spatial ability composites are used in all further analyses.

Achievement test. At the end of the microcomputer workshop, all students took an achievement test. The test consisted of 21 items covering basic commands, simple routines, and complex programs. Some items were multiple choice, some required matching programs to resulting pictures, and some required the student to write one or more lines of computer code. Credit was given for a part of an item if one or more lines of code were correct. The scale of the test ranged from 0 to 32 points. Students' scores ranged from 3 to 32.

The test was written with items measuring five components of computer programming: knowledge of basic commands (3 items), syntax (5 items), interpreting programs written to generate specific graphics (7 items), generating programs to draw certain graphics (2 items), and generating program steps or complete programs to produce certain logical relations (4 items).

The first component, knowledge of basic commands, consisted of items measuring students' knowledge of basic LOGO commands, such as the command that places the turtle in the bottom right corner of the screen.

The second component, syntax, consisted of items that tested the student's knowledge of the correct form of various commands in LOGO but did not necessarily require understanding of what the commands did. For example, one item tested the student's awareness that the command "SETXY 100 PD" was incomplete because it lacked a numerical value for Y (the correct form of the command is "SETXY 100 100 PD").

The third component, interpreting programs for graphics, consisted of items that required the student to match a figure (or the name of a figure) to the program that would generate it. One item, for example, asked the student to select the program that would draw a square (the correct response is "REPEAT 4 RT 90 FD 20 "). Since these items presented both the picture (or its name) and the program, they involved recognition of the function of certain commands and combinations of commands.
Unlike the fourth component, to be discussed next, the items for the third component did not require the student to generate code to draw a figure but allowed the student to generate the figure from the code.

The items for the fourth component, generating programs for graphics, required the student to generate LOGO code to produce a specific figure. For example, one item asked the student to write a program using Turtle Graphics that would draw the letter "M" (not merely print the letter). The program to draw this letter required commands to draw the correct length line segments and join them with the correct angles. As was pointed out above, this component involved generating the correct commands, rather than interpreting a given set of commands. Therefore, although the commands and pictures included in the items for both components were similar, the cognitive processes involved in the two components were different.

The fifth component, logical relations, consisted of items that required the student to produce correct logical relations in a program. For example, one item asked the student to add a line of computer code to an existing program that would stop the program at a certain point. The correct response, "IF :SIZE<5 THEN STOP," involved understanding of the logic of "if-then" statements and the relations among variables and values of variables. Another item required the student to write a program that would make the computer count by 5's and print out each number.

Procedure

Recruitment of students. The microcomputer workshops were announced through UCLA campus newspaper advertisements, flyers, and word-of-mouth. The students who sent applications for the workshops came from as far as eight miles away from campus. The students in each workshop did not know each other.

Assignment of students to groups. Students were assigned to three-person groups on the basis of age, sex, and previous experience with microcomputers. Eleven groups had three persons; one group had two. Students were randomly assigned to groups with the following constraints. All groups were mixed-age and mixed-gender with the same ratio of females to males in most groups. In the first workshop, three groups had two females and one male and two groups had one female and two males. In the second and third workshops, all groups had one female and the rest males. All groups were homogeneous with respect to previous experience with microcomputer programming. Homogeneous groups were used to minimize the possible frustration of the few students who had programming expertise. Of the twelve groups, eight consisted of students who had no previous experience with microcomputers, and four consisted of students who had received some instruction in another computer language (typically, BASIC).

The size of the group had to satisfy two constraints: opportunity for peer interaction and practicality in the microcomputer setting. Three-person groups were chosen because they allow opportunities for group work and are small enough to allow each group member contact with the machine.

Workshop activities. Three identical workshops were held. The first two workshops had 15 students each; the third workshop had five students. Each workshop consisted of five three-hour sessions on five consecutive days. All instruction, materials, and procedures were the same in all workshops.

At the beginning of the workshop, students received an introduction to the microcomputers and were given their group assignments. Students remained in the same groups throughout the workshop. After the introduction to the workshop, students completed the cognitive pretests. The pretests took approximately one hour and 20 minutes to adminster. Except for the achievement test at the end, the rest of the five-day workshop was devoted to LOGO instruction.

Since the curriculum consisted mainly of worksheets that provided guided exploration, the function of the instructor was to introduce topics, answer questions when groups could not proceed, and encourage groups to give their members equal time at the console. At several predetermined points during the workshop, the instructor gave a brief lecture to the whole group about a new topic. For the rest of the time, students worked on the worksheets in their groups. Although groups were given flexibility to work at their own pace, all groups maintained nearly the same schedule. The more experienced groups sometimes worked on more complicated problems than the inexperienced groups, but all groups spent the same amount of time on each worksheet.

In their groups, students were told to work together, to help group members experiencing difficulty, and to ask other group members for help. They were asked to consult the instructor only if no one in the group knew how to proceed. The microcomputer workshop proved an ideal setting to promote group cooperation and prevent division of labor. In contrast to typical small group situations in the classroom in which it is possible for students to work independently using paper and pencil, each group had access to only one computer and so could participate only as a group. Since groups received little direct instruction and were encouraged to try out their ideas, they formulated plans and carried them out as a group. They also corrected their errors as a group.

Students took turns at the console so that all students had equal time entering information into the computer.

Students could sign up for individual sessions with the computer before and after each day's instruction. Nearly all students took advantage of this opportunity.

At the end of the workshop, students completed the achievement test on LOGO. They worked individually on the test, with no assistance from other students or from the instructor. Upon completion of the achievement test all students were given embossed certificates with their names on them, and T-shirts with the name of the microcomputer workshop.

Observations of group work. To obtain information about group interaction, groups were tape-recorded for a minimum of 30 minutes on the third day of the workshop. All groups were working on the same set of problems. Since there were multiple sets of recording equipment, three groups could be recorded simultaneously. For tape-recording, a small microphone was clipped to each group member's shirt. The microphones were connected to one channel of a hand-held stereo tape-recorder (the size of a Sony Walkman). An observer spoke numbers that identified the speaker of each utterance into a microphone that was connected to the other channel. Since each student wore a vest with a number on it for the duration of the taping, the observer could identify each speaker even when students rotated positions.

The written transcripts of the tape-recordings were used to anlayze group interaction. The interaction variables used here included those found in previous research to relate to achievement and additional behavior categories that were unique to the computer learning setting. The interaction variables fell into three catetories: giving help, receiving help, and not receiving help when needed. The interaction variables coded here include gives an explanation, gives a suggestion for input into the computer, makes an error and receives an explanation, asks for an explanation and receives one, receives a suggestion for input into the computer, asks a question and receives no response, makes an error and receives no explanation, and asks for an explanation and does not receive one. Two variables reflecting contact time with the computer were also measured: number of turns at the keyboard and amount of time at the keyboard.

Results

Comparability of Workshops

To determine whether the students in the three workshops were comparable on background characteristics, ability, and achievement, analyses of variance were conducted for all pretests and posttests in the study. Since none of the differences among workshops were significant (*F* tests ranged from 0.16 to 2.32), the samples in the three workshops were comparable on all measured characteristics, and in all further analyses they were combined into one sample of 35 students.

Description of Groups and Group Work

Table 1 presents data on individual and group characteristics prior to group work. There was considerable variation in the sample on all ability measures. The range of abilities or other characteristics within each group, indicated by the average within-group variation in Table 1, tended to be slightly smaller than the range of the sample as a whole. All groups were heterogeneous in ability and all other characteristics except previous experience, but the highest- and lowest-ability students in the sample tended not to be in the same group.

Table 1

Individual and Group Pretest Measures

	Individual		Average within-group variation[a]	Variation in Group Means[b]
	M	*SD*		
Age	12.3	1.3	1.1	0.7
Sex	1.6	0.5	0.6	0.2
Previous Experience	0.6	0.7	0.3	0.6
Mathematics	26.2	8.8	8.5	4.6
Nonverbal Reasoning	9.5	3.2	2.8	1.6
Holistic Processing	6.6	2.0	1.8	1.1
Spatial Ability	23.8	7.6	5.8	5.7
Field Independence	4.2	2.5	2.0	1.6

[a]Mean of within-group standard deviations.

[b]Standard deviation of group means.

Information about group work appears in Table 2. All scores in Table 2 are the frequency of interaction per hour. The means for the interaction measures show that most verbal interaction in the group consisted of students giving others suggestions for what to input into the computer. Students asked a fairly large number of questions (most of which were answered), but they tended to give few explanations and made even fewer errors.

To show the typical experience of students within a group, Table 2 also presents pooled within-group correlations between pretest and group interaction measures. For the pooled within-group correlations, scores on all variables are differences between the individual's score and the group mean. The correlations, therefore, reflect the relative experiences of different group members but do not concern the absolute abilities or frequencies of interaction of group members. The purpose of the correlations in Table 2 is to show whether some group members tend to dominate group interaction or contact with the computer. Because previous research provided few hypotheses about the expected directions of the correlations, the significance levels of the correlation coefficients presented in Table 2 were tested using two-tailed tests.

As can be seen in Table 2, the members of a group who gave the most explanations or suggestions for input tended to be the oldest, have the most previous experience with the computer, have the highest mathematics abilty, the highest nonverbal reasoning ability, or the highest scores in the group on field independence. The group members who received the most suggestions for input tended to be the youngest, have the lowest mathematics ability, or have the lowest nonverbal reasoning ability in the group.

Table 2

Pooled Within-Group Correlations between Pretest Measures and Group Interaction Measures[a]

Interaction Measure	M	SD	Age	Sex	Previous Experience	Mathematics	Nonverbal Reasoning	Holistic Processing	Spatial Ability	Field Independence
GIVES HELP										
Gives explanation	3.6	5.0	.46**	.00	.46**	.57***	.41*	-.12	.31	.51**
Gives input suggestion	27.5	22.8	.39*	.21	.37*	.45**	.34*	-.23	.24	.18
RECEIVES HELP										
Makes error, receives explanation	0.3	0.7	.21	.00	-.08	.36*	.42*	-.23	-.05	.09
Asks for explanation, receives one	1.4	1.7	.22	.10	.09	.22	.00	-.24	.24	.04
Receives input suggestion	24.6	22.8	-.48**	-.18	-.28	-.39*	-.36 *	.10	-.32	-.22
Asks question, receives response	11.1	9.3	-.12	-.07	-.03	-.01	.04	.24	-.01	-.23
RECEIVES NO HELP										
Asks question, receives no response	3.2	3.3	-.12	-.07	-.20	-.12	-.12	-.19	-.16	-.34*
Makes error, receives no explanation	0.6	1.7	-.33	.00	-.05	-.25	-.26	-.08	-.49**	-.32
Asks for explanation, receives none	1.7	3.4	-.21	-.11	-.08	-.08	-.09	-.05	-.30	-.36*
COMPUTER CONTACT										
Number of turns at keyboard	4.3	4.1	.04	-.44 **	.12	.34*	.06	-.16	.10	.41*
Time at keyboard	8.8	5.5	-.29	-.24	.28	.11	-.19	-.04	-.08	.04

[a]For pooled within-group correlations, scores on all variables are deviations between the individual's score and the group mean.

Note: All significance tests are two-tailed.

*$p < .05$

**$p < .01$

Contrary to what one would expect, the members of the group who received the most explanations after making errors tended to have the highest mathematics or nonverbal reasoning abilities in the group. Similarly, the group members who most frequently did <u>not</u> receive help when they made errors or asked questions had the lowest scores in the group on mathematics ability or on field independence.

With respect to computer contact, girls, group members with the highest mathematics ability in the group, and those with the highest scores on field independence tended to have the greatest number of turns. However, although some students may have had more turns than others, the student characteristics did not predict which students had the most contact <u>time</u> with the computer. Younger students had just as much time at the keyboard as older students, girls and boys had equal time, low-ability and high-ability group members had equal time, and group members with previous experience had the same amount of keyboard time as students with no previous experience.

In summary, two fairly consistent findings emerged. The oldest students in the group, the most experienced, and those with the highest ability gave the most help to their groupmates. Contact time with the computer, in contrast, was <u>not</u> related to the relative ability, age, sex, or previous experience of group members.

<u>Programming Performance</u>

Table 3 presents descriptive statistics for computer programming performance. As can be seen by the proportions correct, students in this study scored highest on items measuring knowledge of basic LOGO commands, scored less well on items measuring their ability to interpret graphics programs and to write such programs, and scored lowest on items

Table 3

Computer Programming Performance

Computer Programming Component	Raw Score		Proportion Correct
	M	*SD*	
Basic Command Knowledge	1.9	0.9	.63
Syntax	4.0	1.8	.67
Program Interpretation: Graphics	4.1	2.3	.59
Program Generation: Graphics	2.9	1.5	.58
Program Generation: Logical Relations	3.7	3.9	.34
Total	16.6	8.6	.52

measuring their ability to write programs involving logical relations. This sequence of performance partially supports the hierarchical structure of the test. The lower performance for interpreting and generating programs than for knowledge of basic commands and syntax is consistent with the hypothesis that interpreting and writing programs requires knowledge of commands and syntax plus logical abilities. The nearly identical performance on interpreting graphics programs and generating programs to produce graphics was unexpected, however. It was hypothesized that the processes involved in generating programs (recall and logical ordering of commands and command sequences) would be more complex than those involved in interpreting programs (recognition of functions of commands and command sequences). The emphasis during instruction on generating programs to produce a variety of graphics probably contributed to the nearly equal performance on these two components. The low performance on generating logical relations programs can best be explained by the relatively small amount of time devoted to this topic during instruction.

Predictors of Computer Programming

Cognitive measures and student characteristics.

The relationships among student background characteristics, cognitive abilities, cognitive style, and components of computer programming appear in Table 4. The measures in Table 4 are absolute scores, not deviations from the group mean. Absolute ability and other characteristics were better predictors of achievement than were students' relative standings within the group. Nearly all pretest measures were positively related to computer programming achievement. Older students, students with previous experience in computer programming, students with high scores in mathematics ability, reasoning ability, and spatial ability, and field independent students

Table 4

Correlations between Pretest Measures and Computer Programming Achievement

Measure	Command Knowledge	Syntax	Program Interpretation Graphics	Program Generation Graphic	Program Generation Logical Relations	Total
Age	.02	.25	.28	.40*	.33*	.35*
Sex	.05	-.05	.02	-.07	.04	.01
Previous Experience	.42**	.37*	.25	.37*	.47**	.47**
Mathematics	.46**	.61***	.68***	.62***	.75***	.81***
Nonverbal Reasoning	.41**	.43**	.28	.47**	.44**	.49**
Holistic Processing	.26	.01	.24	.38	.05	.18
Spatial Ability	.52***	.40**	.53***	.62***	.54***	.63***
Field Independence	.38*	.46**	.50***	.59***	.44**	.57***

Note: $\underline{n} = 35$.

*$\underline{p}$ .05

**$\underline{p}$ .01

***$\underline{p}$ .001

obtained high scores on most programming components. Gender and holistic vs. analytic processing did not relate to microcomputer learning. The means of females and males on the programming test were identical, and students who demonstrated holistic processing learned programming just as well as students who did not demonstrate this processing style.

Although the overall pattern of correlations is fairly consistent from component to component, there are a few inconsistencies. First, age was positively related to generating programs for graphics and logical relations but was not related to knowledge of commands, syntax, or interpreting graphics programs. The positive relationships find support in Piaget's work on cognitive development. Logic is the primary ability in formal operations, which begins to be developed at ages 11 to 15 (Inhelder & Piaget, 1958), the age range in this study. Second, previous experience and nonverbal reasoning were not significantly related to interpreting graphics programs, whereas they were significantly related to all other computer programming components. Further data are needed to explain these puzzling results.

The final inconsistent result in the pattern of correlations in Table 4 is the significant relationship between holistic processing and generating programs for graphics. In order to write a computer program to draw a figure (for example, the letter "M"), it is necessary to analyze the figure into distinct parts, write the commands to draw each part, and then connect the commands for each part. Perhaps holistic processing aids in connecting the pieces of the program. An example may help illustrate this hypothesis. In writing programs to draw the letter "M," most students wrote programs that drew the line segments of appropriate length (two long segments, and two short segments). The most common errors occured in assembling the segments. Either the line segments were joined at the correct places but with the wrong angles between them, producing a zigzag pattern, or the line segments appeared at approximately correct orientations but were crossed instead of being joined at their endpoints. Students high in holistic processing tended to make few of these errors.

Best cognitive predictors. Because the patterns of relationships between cognitive measures and students characteristics and programming outcomes were not the same across programming components, multiple regression analyses were performed to determine the best predictors of computer programming achievement. All cognitive measures and student characteristics served as predictors in the analysis. Because the sample size was small, forward selection of predictors was used in the multiple regression analyses with the probability of entering the equation set at .05 to minimize the number of predictors in the equation. Backward elimination of predictors was not used because the small sample size would severely limit the stability of the coefficients of the large number of predictors at early steps of the analyses.

The results of the multiple regression analyses are presented in Table 5. Three of the programming components (syntax, interpreting graphics programs, generating programs for logical relations) and the total score were best predicted by mathematics ability alone. Knowledge of basic commands was best predicted by spatial ability alone. Generating graphics programs was best predicted by a combination of spatial ability and field independence.

It is reasonable that spatial ability was the most important ability for knowledge of basic LOGO commands and generating graphics programs because both components involve the placement and movement of the LOGO

Table 5

Multiple Regression Analyses Predicting Computer Programming from Pretests

Predictor	Command Knowledge			Syntax			Program Interpretation Graphics			Program Generation Graphics			Program Generation Logical Relations			Total		
	b	s.e.	R^2	b	s.e.	R^2	b	s.e.	R^2	b	s.e.	R^2	b	s.e.	R^2	b	s.e.	R^2
Mathematics				.12	.03	$.38^{***}$	.18	.03	$.46^{***}$				.34	.05	$.56^{***}$	.98	.12	$.65^{***}$
Spatial	.06	.02	$.27^{**}$							.09	.03	$.39^{***}$						
Field Independence										.23	.08	$.13^{**}$						
Constant	.45			.72			-.51			-.18			-5.13			5.63		

$^{**}p < .01$

$^{***}p < .001$

turtle on the screen. Based on this explanation, one would also expect spatial ability to play a large part in interpreting graphics programs. Although the correlation between spatial ability and interpreting graphics programs was substantial (.53), the correlation with mathematics was larger (.68). This finding is reasonable because the interpretation items required students to analyze numerical relationships in a sequence of commands (see also Papert, 1980, for further discussion of the mathematical requirements of LOGO).

It should be noted that the regression equations in Table 5 do _not_ imply that the other measures of cognitive abilty and cognitive style were weak predictors of computer programming achievement. Because the intercorrelations among predictors were moderate or high (ranging from .20 to .65), very few predictors could enter the multiple regression equations. When the predictors in Table 5 were deleted from the multiple regression analyses, other combinations of measures frequently were good predictors of the programming outcomes. For example, mathematics ability explained 21% of the variation in knowledge of basic commands and 39% of the variation in generating graphics programs; field independence explained 21% of the variation in syntax, spatial ability and field independence together accounted for 37% and 51% of the variation in interpreting graphics programs and the total test, respectively; and spatial ability accounted for 29% of the variation in generating programs for logical relations.

Group composition. As was described earlier, groups were homogeneous on previous experience with the computer and were heterogeneous on gender and age. Group assignment did not depend on cognitive abilities and groups varied both in mean ability level and in the range of ability within the group. To determine whether group composition influenced computer programming outcomes over and above the characteristics of the individual students, a stepwise multiple regression analysis was performed for each student ability measure or other student characteristic and each programming outcome. In each analysis, the individual score was entered on the first step and the group mean and group standard deviation on that measure were entered on subsequent steps. To take into account the possibility of high correlations among the individual's score, the group mean, and the group standard deviation, the analyses were performed twice, once with the group mean entered before the group standard deviation and once with the group standard deviation entered before the group mean. In no analysis did either the group mean or standard deviation significantly add to the prediction of programming outcomes over and above the indivdual's score (F tests of the changes in $\underline{R}^2$ ranged from 0.02 to 1.91). While these analyses do not entirely rule out group composition effects, they do suggest that group composition does not affect learning of computer programming in a straightforward way. The sample size was too small to investigate the complex interactions between student ability and group composition that have appeared in previous studies (see, for example, Webb, 1980b, 1982b, Webb & Kenderski, 1984).

Interaction in the group. The correlations between interaction and components of computer programming are presented in Table 6. All variables represent the frequency of behavior per hour. Because the interest here is in the effects of group interaction on learning, it was necessary to present partial correlations controlling for previous ability to test the counterhypothesis that interaction is a function of achievement (for

Table 6

Correlations between Group Interaction Variables
and Computer Programming Achievement

Interaction			Command Knowledge		Syntax		Program Interpretation: Graphics		Program Generation: Graphics		Program Generation: Logical Relations		Total	
Measure	M	SD	r	r^{a}	r	r^{a}	r	r^{a}	r	r^{a}	r	r^{a}	r	r^{a}
Gives Help														
Gives explanation	3.6	5.0	.33*	.05	.42**	.12	.37*	-.04	.54***	.27	.47**	.11	.52**	.15
Gives input suggestion	27.5	22.8	.13	-.09	.28	.06	.32*	.09	.42**	.25	.24	-.09	.34*	.04
Receives Help														
Makes error, receives explanation	0.3	0.7	.33*	.24	.42**	.30*	.30*	.11	.33*	.24	.42**	.24	.45**	.34*
Asks for explanation, receives one	1.4	1.7	-.25	-.47**	.05	-.08	.12	-.03	.10	-.01	.08	-.18	.07	-.21
Receives input suggestions	24.6	22.8	.18	.31*	-.32*	-.29	-.10	.02	-.02	.10	-.08	.09	-.11	.04
Asks questions, receives response	11.1	9.3	.21	.08	.03	-.13	.10	-.09	.22	.07	.06	-.16	.12	-.12
Receives No Help														
Asks question, receives no response	3.2	3.3	-.34*	-.27	-.37*	-.31*	-.31*	-.22	-.30*	-.16	-.34*	-.32*	-.40**	-.42**
Makes error, receives no explanation	0.6	1.7	-.21	-.19	-.31*	-.35*	-.11	-.08	-.11	-.05	-.19	-.23	-.22	-.28
Asks for explanation, receives none	1.7	3.4	-.29*	-.30*	-.16	-.17	-.02	.02	-.14	-.07	-.16	-.29	-.16	-.26
Computer Contact														
Number of turns at keyboard	4.3	4.1	.25	-.02	.20	-.14	.36*	.04	.30*	-.08	.36*	.02	.38*	-.03
Time at keyboard	8.8	5.5	.19	.25	.16	.19	.19	.26	.16	.27	.28	.40**	.26	.46**

^{a}Partial correlation controlling for mathematics ability, spatial reasoning, and field independence

$^{*}p < .05$

$^{**}p < .01$

$^{***}p < .05$

example, that students who have learned the material give the most explanations). Because three pretest measures, mathematics, spatial ability, and field dependence, related most strongly to computer programming outcomes (see Table 5), the partial correlations presented in Table 6 controlled for these three measures.

Although many zero-order correlations between interaction and programming achievement were statistically significant, relatively few of the partial correlations were significant. Most of the zero-order correlations for giving help were significant, but none of the partial correlations were. Therefore, the positive zero-order relationships between giving help and programming achievement seem to have resulted from students who mastered LOGO giving explanations and suggestions for input, rather than these interaction variables helping students to learn LOGO.

Similarly, most of the significant relationships between receiving help and programming achievement disappeared when ability was controlled. Although the zero-order correlations between receiving an explanation in response to an error and programming achievement were consistenlty positive, only the partial correlation for syntax was significant. When students who made errors in syntax received explanations of how to correctly write the command, they were able to correct their misunderstanding or incomplete knowledge. Only one significant partial correlation appeared for receiving input suggestions. Receiving suggestions for what to input into the computer seemed to be beneficial only for learning basic LOGO commands. Since the instructional process for learning how to _generate_ and _interpret_ LOGO programs was based on discovery, it is reasonable that being told what to enter into the computer would not help students learning how to write programs in LOGO. Receiving responses to questions did not relate to any component of programming achievement.

The significant negative relationship between knowledge of basic commands and receiving explanations in response to requests for them was unexpected. Receiving explanations seemed to be detrimental to learning of basic commands. It is possible that students who asked for explanations were relying on other students rather than learning from the explanations.

The correlations between receiving no help when needed and programming achievement were uniformly negative, but only some of the partial correlations were statistically significant. Asking a question and receiving no response was negatively related to syntax and generating logical relations programs. When students were confused or unsure of syntax and logical relations, receiving no responses to their questions seemed to be detrimental to their learning. Also detrimental for learning of syntax was making an error and receiving no explanation. When students input a command that was formatted incorrectly, they did not learn the correct form unless another student explained it. Receiving no explanation in response to a request was negatively related to learning basic commands. When students asked how a command worked and did not receive an explanation, they tended not to learn the basic commands.

As with the verbal behavior variables, few of the partial correlations with computer contact variables were significant. Although the number of turns at the keyboard seemed to correlate positively with interpreting and generating programs, none of the relationships were significant when ability was controlled. The time at the computer keyboard related to only one component of computer programming. Students who spent more time at the keyboard of the computer scored higher on generating logical relations programs than students who spent less time, when ability was controlled.

A dominant feature of the correlations among group interaction and programming outcomes in Table 6 is that group interaction and computer time influenced learning of the basic commands and syntax, and logical relations, but did _not_ relate to interpreting or generating graphics programs. A possible explanation for this result comes from the nature of the questions that students asked. Particularly at first, students tended to ask other students whether their sequences of commands for generating graphics were correct or sensible before trying them. Or, they sometimes asked other students why a program did not work before attempting to find the errors. Responses to these kinds of questions would not be expected to help students learn, nor would lack of responses be expected to be detrimental for learning.

Best group interaction predictors. To ascertain which of the interaction variables were most important for learning programming, multiple regression equations were computed with mathematics, spatial ability, and field independence and the group interaction variables in Table 6 as predictors. As would be expected from the correlations in Table 6, group interaction contributed to computer programming over and above ability only for knowledge of basic commands, syntax, and generating logical relations programs (Table 7). Controlling for spatial ability, receiving explanations in response to requests for them was negatively related to learning basic commands, and receiving explanations in response to errors was positively related to learning basic commands. When mathematics was in the equation for syntax, receiving no explanation

Table 7

Multiple Regression Analyses Predicting Computer Programming from Pretests and Group Interaction Variables

Predictor	Command Knowledge			Syntax			Program Generation: Logical Relations			Total		
	b	s.e.	R^2 change	b	s.e.	R^2 change	b	s.e.	R^2 change	b	s.e.	R^2 change
Mathematics				.12	.03	.38***	.31	.05	.56***	.72	.08	.65***
Spatial Ability	.07	.02	.27***									
Asks for explanation, receives one	-.22	.07	.14**									
Makes error, receives explanation	.33	.15	.08*									
Makes error, receives no explanation				-.29	.13	.08*						
Time at the keyboard							.21	.07	.06*	.44	.13	.08*
Asks question, receives no response							-.31	.12	.06*	-.82	.22	.06*
Constant				.95						2.76		

Note: Only equations for programming outcomes including significant group interaction variables are presented here.

*$p \leq .05$

**$p < .01$

***$p < .001$

in response to an error was negatively related to learning the correct syntax of commands. For generating logical relations programs and for the total test, receiving no response to questions was negatively related to programming achievement, and time at the keyboard was positively related to programming achievement when mathematics was taken into account. For interpreting and generating graphics programs, none of the group interaction variables examined in this study significantly predicted achievement when ability was taken into account.

Predictors of Group Interaction

As was reported in the previous section, five interaction variables contributed to computer programming over and above ability: asks for an explanation and receives one, makes an error and receives an explanation, makes an error and receives no explanation, asks a question and receives no response, and time at the keyboard. To determine whether group interaction could be predicted by student characteristics or ability, correlations were calculated between the eight pretest measures and the five group interaction variables. Only three of the correlations were statistically significant at the .05 level. Since two of the 40 correlations would be expected to be significant by chance, however, the three significant correlations are not taken seriously. Therefore, the major conclusion is that students' experiences in the group cannot be predicted from their abilities or other characteristics.

The finding that student characteristics did not predict students' interaction in the group seems to be at odds with the results presented in Table 2 concerning within-group interaction. However, the five interaction variables that best predicted achievement tended to be those with few significant relationships with student characteristics, even in

Table 2. Of the 40 correlations among the pretest measures and the five interaction variables that related to achievement, only four in Table 2 were significant, compared with three in the present analysis.

Discussion

The present study examined the cognitive abilities, cognitive styles, and demographic characteristics that predicted learning of computer programming in small groups; investigated the group process variables that predicted learning of computer programming; and examined the student characteristics that related to group processes. Based on a hypothesized hierarchy of processes involved in computer programming, five achievement outcomes or components were examined: knowledge of basic commands, knowledge of syntax, ability to interpret programs written to generate graphics, ability to generate graphics programs, and ability to generate programs for logical relations.

Although measures of mathematics ability, spatial ability, non-verbal reasoning, field independence, and previous experience were positively related to most of the components of computer programming, different components were best predicted by different abilities. Mathematics ability (a combination of numerical skills and numerical reasoning) best predicted syntax, interpreting graphics programs, generating programs for logical relations, and the total score on the achievement test. Spatial ability best predicted knowledge of basic LOGO commands, and a combination of spatial ability and field independence best predicted generating graphics programs. If one considers the highest level components, generating programs, it is clear that the cognitive requirements depend on the type and purpose of the program. Logical relations programming seems to require mathematics ability whereas graphics programming seems to depend most on spatial ability. Since the two abilities are not perfectly correlated ($\underline{r}$ = .59 in this sample), a person's profile of abilities needs to be taken into account to predict his or her learning of computer programming. Performance on one kind of ability measure (e.g., mathematics) is not enough to predict learning of all aspects of computer programming. Furthermore, the greater importance of spatial ability than mathematics ability for some types of computer programming helps discredit the notion that computer programming is only for the mathematically inclined.

It is important to note that gender did not relate to learning of computer programming. Females and males performed equally well on all components of computer programming. Females and males also performed similarly on all pretest measures, suggesting that they were comparable on abilities, age, and previous experience. It is possible that the females in the present study constituted a select group because they had as much previous experience with computers as boys did. For the population of girls at large, the relationships between previous experience and computer programming outcomes (in this study the correlations ranged from .25 to .47) suggest that it is important to provide girls (and boys) with computing experience early to prevent one group from being at a disadvantage later.

The relationships between group process variables and computer programming outcomes in this study were different from those in previous studies of classroom learning and also suggest that small group work may be a viable setting for learning computer programming. Most importantly, the number of turns and the amount of time at the keyboard had almost no relationship with computing outcomes. Indeed, the students not at the keyboard seemed to be at least as involved with the material as the students at the keyboard. This equality of involvement is seen in the large number of suggestions that students gave to the group member at the keyboard (the number of suggestions exceeded all other instances of verbal interaction combined).

The importance of specific verbal interaction variables for learning was less in this study than in previous studies of small group work in the classroom. In the present study, in contrast to nearly all previous studies, giving explanations did not help students to learn computer programming. The students who gave the explanations seemed to be the ones who had already learned the material. Receiving explanations, found in some previous studies to be beneficial for learning, influenced only learning of the basic commands, but even there the findings were inconsistent: receiving explanations in response to errors seemed to be beneficial, whereas receiving explanations in response to questions seemed to be detrimental. Receiving _no_ explanations in response to errors and receiving no responses to questions, nearly always found in previous studies to be consistently detrimental for learning, seemed to be detrimental only for learning the syntax of commands and how to generate programs for logical relations but not for other computing outcomes. In summary, then, verbal interaction in the group seemed to influence learning of basic commands and syntax (for example, having another student immediately detect that a space or semi-colon was ommitted from a command of line of code seemed to be helpful) but did _not_ seem to influence learning how to interpret and generate graphics programs, the primary focus of instruction.

Two factors may account for the lack of relationship between verbal interaction in the group and learning how to interpret and generate graphics programs. The first is the nature and purpose of the interaction, particularly the questions asked. As was noted earlier, some students asked for confirmation that their programs were correct before running them, or asked other students for help in completing programs or detecting errors. Since these questions were not seeking specific information or explanations, receiving answers would not be expected to be beneficial for learning; in fact, they might be expected to be detrimental if students were relying on others to complete or correct the program. Given the nature of students' questions, then, it is not surprising that the group's response to them had little impact on learning.

The second factor explaining the lack of relationship between verbal interaction and interpreting and generating graphics programs may be the learning medium: the computer. In group work in the typical classroom setting, students can verbally explain how to do the work or can show another student the solution, for example, by writing the solution to a mathematics problem on paper or on the blackboard. Even while "showing" the work, students often rely on verbal cues when the written solution is not legible, is not understandable, or is not complete. With a computer, however, the strategies or approaches to solving a problem (the program) and the results are clearly seen by everyone because they appear on the screen in standardized fashion. In this way, students can learn from what other group members _do_ as well as from what they _say_. Thus, the group processes influencing learning how to interpret and generate graphics programs in the present study may have been predominantly nonverbal.

Finally, background characteristics of the students and their abilities and cognitive styles tended not to predict their experience in the group setting. First, student characteristics did not relate to contact time with the computer. This finding relieves the concern of many educators that high-ability students, boys, and students with previous experience would try to monopolize the computer. However, since there was some variability within the group on time at the keybaord, further research should investigate whether other student characteristics--such as personality variables--might predict contact time. Second, student characteristics did not relate to the categories of verbal interaction that were important for programming achievement: receiving explanations in response to questions and errors, and receiving no responses to questions and errors. Third, high-ability students tended to give the most help, but giving help was not related to computer programming outcomes. The latter two findings are consistent with previous research showing a positive relationship between ability and giving help, and no consistent relationship between ability and other verbal interaction measures (see Webb, 1982c).

In summary, the present study shows that learning computer programming can be accomplished successfully in group settings and that a variety of student abilities, cognitive styles, and group process variables are needed to predict performance in computer programming. Furthermore, different profiles of student characteristics and experiences in the group predict different computing outcomes.

References

Bauer, R., Mehrens, W. A., & Vinsonhaler, J. R. Predicting performance in a computer programming course. Educational and Psychological Measurement, 1968, 28, 1159-1164.

Collins, A., Bruce, B. C., & Rubin, A. Microcomputer-based writing activities for the upper elementary grades. Proceedings of the Fourth International Learning Technology Congress and Exposition. Warrenton, VA: Society for Applied Learning Technology, 1982.

DeNelsky, G. Y., & McKee, M. G. Prediction of computer programmer training and job performance using the AABP test. Personnel Psychology, 1974, 27, 129-137.

French, J. W., Ekstrom, R., & Price, L. Kit of reference tests for cognitive factors. Princeton, NJ: Educational Testing Service, 1963.

Gaines, R. Developmental perception and cognitive styles: From young children to master artists. Perceptual and Motor Skills, 1975, 40, 983-998.

Goldman, R. D., & Hudson, D. J. A multivariate analysis of academic abilities and strategies for successful and unsuccessful college students in different major fields. Journal of Educational Psychology, 1973, 65, 364-370.

Hollenbeck, G. P., & McNamara, W. J. CUCPAT and programming aptitude. Personnel Psychology, 1965, 18, 101-106.

Jabs, C. Game playing allowed. Electronic Learning, 1981, 1.

Mayer, R. E. A psychology of learning BASIC. Communications of the ACM, 1979, 22, 589-593.

Mayer, R. E. Different problem-solving competencies established in learning computer programming with and without meaningful models. Journal of Educational Psychology, 1975, 67, 725-734.

Mayer, R. E. Some conditions of meaningful learning for computer programming: Advance organizers and subject control of frame order. Journal of Educational Psychology, 1976, 68, 143-150.

Mayer, R. E. The psychology of how novices learn computer programming. Computing Surveys, 1981, 13, 121-141.

McNamara, W. J., & Hughes, J. L. A review of research on the selection of computer programmers. Personnel Psychology, 1961, 14, 39-51.

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

Papert, S. Society will balk, but the future may demand a computer for each child. Electronic Education, 1981, 1, 5.

Peterson, P. L., & Janicki, T. C. Individual characteristics and children's learning in large-group and small-group approaches. Journal of Educational Psychology, 1979, 71, 677-687.

Peterson, P. L., Janicki, T. C., & Swing, S. R. Ability x treatment interaction effects on children's learning in large-group and small-group approaches. American Educational Research Journal, 1981, 18, 453-473.

Raven, J. C. Standard progressive matrices. London, England: H. K. Lewis & Co., Ltd., 1958.

Rubin, A. Making stories, making sense. Language Arts, 1980, 285-298.

Rubin, A. The computer confronts language arts: Cans and shoulds for education. In A. C. Wilkinson (Ed.), Classroom computers and cognitive science. New York: Academic Press, 1982.

Snow, R. E. Aptitude processes. In R. E. Snow, P. A. Federico, & W. E. Montague (Eds.), Aptitude, learning, and instruction (Vol. 1): Cognitive process analyses of aptitude. New Jersey: Lawrence Erlbaum Associates, 1980.

Swing, S. R., & Peterson, P. L. The relationship of student ability and small-group interaction to student achievement. American Educational Research Journal, 1982, 19, 259-274.

Webb, N. M. An analysis of group interaction and mathematical errors in heterogeneous ability groups. British Journal of Educational Psychology, 1980, 50, 1-11. (a)

Webb, N. M. A process-outcome analysis of learning in group and individual settings. Educational Psychologist, 1980, 15, 69-83. (b)

Webb, N. M. Group composition, group interaction and achievement in cooperative small groups. Journal of Educational Psychology, 1982, 74, 475-484. (a)

Webb, N. M. Peer interaction and learning in cooperative small groups. Journal of Educational Psychology, 1982, 74, 642-655. (b)

Webb, N. M. Student interaction and learning in small groups. Review of Educational Research, 1982, 52, 421-445. (c)

Webb, N. M. Predicting learning from student interaction: Defining the interaction variables. Educational Psychologist, 1983, 18, 33-41.

Webb, N. M., & Kenderski, C. Student interaction and learning in small group and whole class settings. In P. L. Peterson, L. C. Wilkinson, & M. Hallinan (Eds.), The social context of instruction: Group organization and group processes. New York: Academic Press, 1984.

Zacchei, D. The adventures and exploits of the dynamic storymaker and textman. Classroom Computer News, 1982, 2, 28-30.