In chapters 2 through 5 of this book I described four distinct visions of computers in education. Whether the computer is acting as an interactive textbook or as an expressive medium, we have seen that the teacher's role varies in different situations. In this chapter I focus on teachers and their roles. A general issue emerges as to how teachers might learn their practice so as to integrate the computer's potential into their classroom activities. Here I put forth a view that presents one possible route into computing for educators. This path is based on my personal experiences in working with children and teachers. This path has a long history, starting over twenty years ago when I began my collaboration with Seymour Papert.

In my view the computer is an intellectual agent, operating in a culture and reflecting ideas of the culture. Logo, as a programming language, is used as the main means of communication with the computer. But what really guides this culture is the belief that naive as well as sophisticated people should be given access to powerful ideas in a dynamic and concrete way; computers can offer easy access to such ideas as well as models for how to build on them. Furthermore, although there is an accepted and widely held attitude that experience in programming computers is in itself stimulating, enriching, motivating, and valuable in many ways for children, high school students, indeed, everyone, there is a belief that more can be offered. The computer becomes a medium for self-expression and an instrument for one's own intellectual development. This process involves people, machines, and ideas. Each is linked to the others.

Perhaps the major question to be discussed is, How do teachers acquire knowledge about computers that they can put into practice in their daily teaching? Most courses on computers in education take a skills approach, whereby within a semester they provide the students with some of the syntax of a programming language or familiarity with word processing and assorted educational games. In contrast, I talk about what you might do while you take that course or the next one. That is, how you can take advantage of the computer's potential.

For many teachers learning about computers can be frightening. The machine, the keyboard, the floppy diskettes, the typing—all contribute to a kind of bewilderment and tension. I would like to be there to tell you to relax, it's okay. I would like to continue to give you the same advice I give children. I would like to be there to set your computer up so that you are not confronted by the bewildering array of machine parts and wires. But, I can say only that in the future computers will contain a library of information as well as space for personal work. There will be a significant breakthrough in speech recognition so that the computer will understand verbal commands as well as those passed to it from a keyboard. Of course, talking out loud can be a distraction in a classroom. Anyway, that is in the future, and the frustrations of communicating with the computer through a keyboard are momentary and fade away as you gain familiarity.

I emphasize first encounters because first encounters often leave lasting impressions. I want to help make that first experience a positive, almost magical one. Rather than concentrating on the syntax of a language or even a game I want to concentrate on the computer experience itself. I want to provide you with insight into the computer's potential as a powerful personal intellectual agent. I want, therefore, for you to be able to do something interesting that you could not do before using a computer.

Turtle geometry offers a multitude of possibilities to meet these criteria. Although there are other programming domains from which to draw, turtle geometry stands out as the most concrete and obvious. After issuing a few commands to a turtle on a TV screen, you have created a square, a star, a squiggly path showing the turtle's reactions to your commands. You are strongly impressed by the first twenty minutes of your computer experience.

This is the beginning of a remarkable intellectual journey toward math-land and the world of computational ideas. By going from the concrete to the abstract and from the abstract to the concrete, you develop procedures to describe particular actions by studying individual instances of that general behavior. For example, seeing that the process by which the turtle walks in a square path is the same as walking in a triangular path leads to an understanding of the total turtle trip theorem and the POLY procedure.

Computers offer a new opportunity to help teachers to enhance their teaching and understanding of children and to keep schooling from becoming an alienating experience. Whether or not this will happen is unclear. To make it happen, action needs to be taken now to reeducate educators to develop models of what might be possible.
Teacher Preparation

In discussing teacher preparation people tend to focus on what software packages to talk about and whether a teacher should be familiar with Logo, BASIC, or Pascal. In discussions about using computers in schools people tend to extend their concerns to whether computers should be in a laboratory or in individual classrooms. Rarely, do discussions focus on what kinds of learning environments might be developed to introduce educators to a computer culture. Furthermore, the amount of time set aside for teachers to become comfortable and knowledgeable is short. The resulting tendency is to encourage the popular beliefs that (1) educators in general cannot learn to program, and (2) programming as we know it today will disappear in a few years, so why learn about it now? Of course, educators can learn to program as easily as ten-year-olds or they might find it impossible, depending on the kind of programming projects they are engaged in. As for the second objection, it is likely that programming techniques will change, but important ideas in computer science, such as naming, proceduralizing, debugging, heuristic methods, and simulations, will not disappear. What is likely to change is the syntax and power of particular programming languages and the domains over which they operate.

Developing programming projects and debugging them can be a rich intellectual experience for teachers and children. In what follows I would like to explore in some depth some ideas about preparing people to work with children and computers. I want to situate this discussion in terms of the question, What does an elementary school computer teacher need to know? I am going to talk about my own experiences. In discussing this question, my strategy is to examine carefully the knowledge I bring to bear on teaching children to program. As a teacher, I see much of my own development as acquiring (1) a repertoire of programming projects that make the power of programming techniques and concepts apparent to beginners, (2) a vocabulary for talking about structured programming, and (3) a sensitivity to the kinds of resistance that keep many adults and children from experimenting with mathematical ideas. This knowledge, much of it tacit and intuitive, developed over many years; I now try to formulate it explicitly.

A Model for Preparing Computer Educators

Becoming comfortable with a computer takes time no matter what the computer is used for. Because this is so, we might as well learn to use the computer in a way that will maximally enhance both our intellectual development and children’s. The key to finding such a computer experience is to use a computer in a way that encourages self-expression. For some people this happens by using a spreadsheet program. Others find that using word processing programs gives them a new sense of control over their thinking and their expressiveness. For others, using drawing programs, such as Macpaint on the Macintosh, provides them with a new sense of self-expression that would not have been available to them without the computer. For other people, a different experience is needed. They find a new awareness of themselves as thinkers, theory builders, and logical developers of projects when they learn to program a computer in a domain offering immediate and concrete feedback. This is what happens for many people when they learn to control graphics turtles through the Logo language.

I emphasize people feeling comfortable with a computer because I see that as a key to making computers real intellectual tools for everyone. When you feel comfortable with a computer you can begin to approach it critically and constructively. You can begin to build your own tools or to describe the kinds of things you would like to do with a computer’s help.

I imagine the following kind of introductory experience. Imagine that you are taking a one-semester laboratory course. For this course, you might set aside a minimum of six hours a week. In that time you will interact with a computer, Logo, turtle geometry, children, other teachers, and a Logo expert. Part of that time will be spent in the classroom; part of the time will be spent with a computer by yourself; and part of the time will eventually be spent observing and interacting with a child at a computer. Your goal is to prepare yourself for sharing your experiences with children. During the first four weeks of the semester, you work on your own programming projects. As the weeks go by, you find new challenges for yourself and begin to plan introductory experiences for children. In the next four weeks, you apply your knowledge to actual work with a child. I would encourage you to work with one student at a time at the computer. In this intimate process you begin a kind of research project of your own. You have an opportunity to observe and document the child’s interactions with the computer as well as your own interactions with the child. You sharpen your observational skills and perhaps become more sensitized to what the child suggests and how he or she suggests debugging strategies or project extensions. In later weeks you might want to work with new students. You might even want to observe students working together, although you will have plenty of time to do so later. You want to focus on you and your role. You want to become familiar with your bugs as well as the child’s. You want to think about what suggestions to make that will help the child at critical moments in debugging a procedure.

This involvement is an intensive one. You might keep a journal of your own learning and your teaching experiences as you work with children or as
you develop your own projects. Part of the process is to share your experiences with other student teachers, and part of the process is to help you to understand the learning process and to remember your own learning and teaching struggles.

A Model for Introducing People to Computers

When I go into a new teaching situation, I have several models in mind and a willingness to switch from one to another or even diverge from them. My primary goal when I introduce students to computers is for them to do something that they could not have done without a computer but something that they can clearly relate to. I also want to think about a next step: how to build on what happened in the first session in the next and the next. Perhaps flexibility is one of the most powerful ideas in this environment, but being flexible implies having a model to depart from. Thus I have a model in mind of paths a beginner might take in a first session. By now I have worked with hundreds of beginners and so this model has been compiled from many interactions. I have models of both beginning programmers and pathways that beginners might take into the computer culture. Perhaps the following discussion of some of the more apparent learning styles I have observed in this environment might illustrate what I mean.

A Model of Learning Styles

In discussing learning styles I want to emphasize that a computer environment such as Logo, equipped with a powerful, structured language and a dynamic graphics system, provides a solid support for diversity and at the same time draws on a common pool of ideas. I discuss work with turtle geometry because I have had much experience with it and can draw on many examples from my past research. I have accumulated some specific information on learning styles (Solomon 1978; Papert et al. 1979a, 1979b; Watt 1979; Lawler 1980). I mention only three styles of learning here. There are more, but these three come sharply into focus.

First there is the planner, who builds his or her structured programs from the top level down or from the bottom level up but who works from a coherent, formulated plan. Another distinct learning style is that of the macroexplorer, who likes to mess about with subprocedures or building blocks to arrive at a product rather than starting with a specific goal. In this case the learner is intent on exploring the effect of the particular building block, and so the result is open ended. Finally, there are some learners who have to explore their environment on a micro level before they can establish pat-
terns of planning or directed exploration. These students are often the most timid learners, doing such things as assuring themselves that FORWARD 100 is the same as FORWARD 1, FORWARD 9, FORWARD 11, FORWARD 23, and FORWARD 56. Others might exhibit this conservative, gradual exploration by using the same numbers as inputs to FORWARD and RIGHT or by repeating the same commands over and over.

The teaching method I have developed is based on a model of a child who might use, though to different degrees, all three of these learning styles. In an initial session I might try to “plant seeds” for all three. For example, I would encourage a beginning student to drive the turtle around the screen in a series of direct commands with no goal other than to understand the turtle’s behavior. But in the same initial session I would suggest some concrete goal, such as making the turtle walk in a square or, perhaps, having placed some squares on the screen, I would ask the child to make the turtle touch them. In this I elicit primarily a microexplorer style with some hint of the planner style.

I facilitate a macroexplorer style by seizing on something interesting the child has just done and suggesting “teaching” it to the computer. Thus I encourage the child to proceduralize and thereby turn the turtle meanderings into repeatable processes, procedures, and building blocks and then to use these procedures as subprocedures to create unanticipated designs.

I would encourage children to follow a planner style by asking them to choose a design from a collection built from a subprocedure familiar to the children already or by asking them to make a design of their own and then to develop the procedures and control structure. Being sensitive to at least these styles of learning and their natural intermixing helps to develop strategies for intervention. These styles of learning are exhibited by novices in this environment, regardless of their age.

A Model of Programming

In an environment in which the learner programs a computer, a program can easily become long, complex, labyrinthine—a long list of one instruction after another. I see one of my roles as instructor as encouraging a more intelligible style of programming, a style in which the program is broken in a rational way into intelligible subpieces, but this requires knowledge and experience.

In a first session I would want to convey the following ideas: (1) programming is a process of engaging the computer in conversation using the vocabulary the computer understands; (2) the computer’s understanding can be expanded easily; (3) giving words meanings involves describing a
A Model of Mathematics

This discussion focuses on projects in turtle geometry. Turtle geometry allows the creation of graphic designs in a way that quickly reveals relationships between aesthetics and mathematics. These relationships may be a little surprising. Often simple looking objects are a result of complicated processes, whereas complex objects frequently result from simple procedures. One goal of my teaching is to capitalize on opportunities that allow children to encounter an (almost inevitably interesting) relationship between aesthetics and mathematics. The following turtle geometry projects will serve as illustrations.

Figure 7.1
The FLOWER Project.

The FLOWER project (figure 7.1) was originated by six-year-old Esther. FLOWER is made from CB. CB, in turn, is made from BALLOON (figure 7.2). To illustrate how BALLOON is used to make CB, let’s assume the turtle is in the following starting state, with its nose in the direction of compass heading 0 (figure 7.3).

After BALLOON is run, the turtle’s heading is 90 degrees to the right of its initial heading. By running BALLOON again and again, we can see the turtle following the same behavior as in making a square (figure 7.4).

The turtle turns through 360 degrees before coming back to its starting heading, and because it moved the same amount between each rotation, it also returned to its starting position.

FLOWER is made by running CB, turning the turtle 51 degrees, and then repeating these two actions six more times. Why 51 degrees? Well, Esther just happened to pick that number. Why did 51 have that effect on CB? The answer lies in the fact that turning 51 degrees seven times results in a total turning of 357 degrees, which is very close to a complete rotation of 360
degrees. In this situation Esther was satisfied; for her purposes the design was complete.

A more interesting question is, How did Esther know to probe the turtle environment in this way? She knew certain facts about turtles and turtle-directed procedures which she had gained from her experiences with the turtle (Solomon 1976b). For example, if the turtle draws something and does not return to its starting state, repeat the procedure. Something interesting will happen, and eventually the turtle will come back to where it was initially. On the other hand, if the turtle does return to its starting state when it makes a design, then change the turtle’s heading and run the program again. In other words, Esther did not need the expert’s knowledge about the power of 360 degrees; rather what she needed was the idea of the total turtle trip, which, translated into intuitive knowledge, told her to keep repeating an action until the turtle returned to its starting state. Esther’s learning style in this project and many others was that of a macroexplorer.

The SWAN project (figure 7.5) was initiated by Martha, a sixth grader. She attacked this project as a planner. She worked on the project for many hours. In a sense her final rendition was arrived at through exploration, but her overall strategy was always worked out beforehand. Her final scenario consisted of two swans facing one another. One was the mother who waited for the baby swan to swim toward her.

The swan is made up of clearly identifiable parts (figure 7.6). These parts, in turn are made up of either a left arc or a right arc (figure 7.7). The project builds on the total turtle trip idea to make HEAD and BODY, but the control structure that puts all the pieces together is complicated, much more so than in FLOWER, and thus might involve several days of concentrated work in order to debug all the pieces and put them together.

Martha’s project also involved constructing two swan procedures, one that caused the swan to look to the left and the other to the right. Furthermore, her procedures could make different sized swans. Thus Martha ex-
Figure 7.6
The parts of SWAN.

Figure 7.7
The two arcs (right and left) that make up the head, neck, and body of SWAN.

Figure 7.8
SWAN 100, an example of scaling.

hibited a deep understanding of scaling, of relationships of the parts to the whole. For example, Martha could invoke her procedure by saying SWAN 100 (figure 7.8). BODY would be made of pieces of a circle (arcs), that had a width of 100 steps (diameter). NECK would be made from parts of a circle half that size, 50. And, finally, HEAD would be made from circle pieces with a diameter of 25. Martha used both intuitive and formal knowledge about turtle geometry in the execution of this project.

The BEAR project (figure 7.9) was initiated by Lisa, an eleven-year-old from an inner city school neighboring MIT. But BEAR and its derivatives have been developed and embellished by ten-year-olds, six-year-olds, and adults. Lisa had evinced a microexplorer style of learning in her previous turtle geometry work. This project encouraged a macroexplorer approach. Its building blocks are circles, and it grew out of her explorations with circles.

This project is somewhere between the other two (FLOWER and SWAN) in specific geometric knowledge. It illustrates more clearly than the other two that there are many ways of arriving at a particular goal. Picking a starting state for the turtle influences the construction of the program. Whether the job is thought of in terms of subprograms (subprocedures) or whether the design is tackled to the graphics screen (in fact or fantasy) and the turtle is made to trace the path etched on paper has important consequences in how the project is developed. If the design is to be taught to the turtle by break-
ing it up into parts, then the programmer has to decide what building blocks are needed.

The bear has several interesting features. It is made entirely of circles. The head and the body are identical. The project is easily changed to focus only on the head or to create with minor modifications a different animal (figure 7.10). The project has appeal to children and adults. Circles have properties that arcs do not: They are closed figures and multiply symmetric; they are good building blocks for the imagination; they are suggestive of realizable and unique designs that satisfy aesthetic and mathematical criteria for a wide variety of tastes. The project is simplified by having powerful building blocks, such as variable-sized circle programs. If the programmer builds on the symmetric character of the head or body in order to attach ears or feet, then the project is further simplified. The same technique works for eyes, nose, mouth, tail, etc.

Remarks on Turtle Geometry and Other Activities

Turtle geometry is but one part of the Logo computer culture. There are other areas of activity that have been explored and many more waiting to be explored. But turtle geometry serves to illustrate key characteristics of the culture, especially the idea of exploring an environment and the objects in it by manipulating them through a complex of interactions based on procedural descriptions and by elaborating the descriptions through debugging—testing procedures in real situations, getting concrete feedback on these actions, and then adjusting the initial descriptions to take these results into account. The process of procedural description and debugging might be seen as a dynamic process of assimilation and accommodation, of making theories and revising them as a result of experience and knowledge, but doing this playfully as an enjoyable activity involving one’s whole self.

Sharing in this process is a self-empowering experience for all participants. A different way of looking at learning and teaching emerges, one that is based on the Piagetian idea that even young children have theories, although they might be incomplete. Thus teaching and learning are not a process of being wrong or right but rather one of debugging. Learning and teaching are seen as a process of developing debugging aids as knowledge gaps are discovered and filled in. The learner becomes an active participant and often takes on the role of teacher. The teacher is not an infallible object in this universe but rather a mentor or guide providing good models to emulate and modify.

Discovery Learning in the Logo Culture

Finally, I would like to describe an experience that captured for me the alienation existing in current educational practice for teachers and learners as they engage in school mathematics. From 1970 to 1971 I was working with a group of fourteen fifth graders chosen because they fell into the “average range” on standardized test or on school performance records. The children were the first group to participate in a year-long class using turtles. They initially had to share one mechanical turtle. It was a big, yellow, slow-moving cannister with a huge light in its front and a thick wire at its rear connecting it to a computer terminal. In the middle of its belly it carried a pen that could be raised or lowered so that it could trace its path on large sheets of drawing paper taped to the floor.

Winter vacation was close at hand. The children were just finishing up their projects. We had discussed the possibility of making a holiday sign with the turtle. There was not enough time before vacation for them to develop letters from scratch and then put them together into words and sentences. Instead I dashed off letters one evening at my terminal at home without a chance to debug the procedures with the turtle. The next day Papert was assisting in the class when the following incident occurred.
Peter was debugging P. It was made up of a straight line and a half circle. Peter fixed the procedure so that instead of P it drew \( \bar{P} \). Why didn’t it close? Where was the bug? Peter was not at all satisfied with the result and wanted our help. His procedure seemed all right. Papert, Peter, and I were on our hands and knees observing the turtle slowly carrying out the instructions. A visiting MIT student walked into the classroom and asked what was going on. I turned to him and stated the problem. Peter looked up in surprise and said: “You mean you really don’t know how to fix this bug?” He suddenly went into action and got the P to close. Peter told us he thought I had deliberately put bugs into the letter drawing procedures, and so he did what he usually did when he thought he was being tricked by a teacher, he withdrew and refused to play the “discovery” game. The rest of us went back to MIT and discussed the bug. Part of the problem was in the algorithm used to make the half circle, which I promptly fixed by splitting the amount the turtle turns on each round into two separate actions; part of the problem was in the turtle mechanics, involving wheel slippage, something that could not be fixed, only compensated for. But Peter’s ingenuity dealt with the bugs once he saw the difficulty as a “real” problem.

This story is not atypical in what it reveals about children’s perception of schooling or of their ability to solve problems in original ways. Peter balked at what he sensed was another application of the “discovery method,” a trick often used on him, but he jumped at the chance to engage in real discovery. Over and over again children have seen simple, direct solutions to problems for which I see only complexity. So I learn from them. On the other hand, I have often shown them simple and direct paths at times when they see only complexity.

An Image of the Teacher

Out of experiences in this culture a new breed of teacher emerges: This teacher is thoroughly imbued with a coherent computer culture and its language. She knows how to use this language to talk interestingly about things people from outside the culture know and care about. This teacher has a fluent mastery of certain powerful ideas. She is thoroughly familiar with project terrains through which she will guide those who come for “instruction” (but will be given something better!). She has been there often! She knows how to observe people engaged in thinking, learning, puzzling, agonizing, rejoicing .... She knows (and can only know this through experience) when to intervene and when to let the learner struggle. She believes that the key goal for any learner is to improve his image of himself as a learner, as an active intellectual agent.

A Proposal

My intention here has been to provide some ideas for teachers who want to integrate computers into children’s everyday learning environments. I suggest a way that individual teachers might educate themselves, making use of courses not really designed for this purpose. At the same time, I foresee the possibility of providing teachers with learning centers, in which this integrated process is the norm.

I hesitate to suggest setting up teacher centers as a way to meet the needs of the future because there are already centers that have been set up to meet teachers’ needs. What I have in mind is different in quality. The center I have in mind will give teachers the same rich learning experience as we would like children to have. Each teacher will have access to computers at home as well as in the center. The computer will be an accepted part of their everyday life and thus easily accessible as a debugging aid and intellectual agent. The center will be a place to explore and develop personal styles of teaching and learning. It will demand a deep personal commitment from each of the participating teachers. This learning center will draw on the ideas of Papert and Kay in developing computer environments and also on the work of Suppes, Davis, and Dwyer.

Some Research Areas

Whether the computer will be used to replace teachers and control what children are taught or whether the computer will be used as a teacher’s aid and as an individual instrument of learning is an issue that cannot be resolved now. But we must take action now toward working for whichever view we think will enhance children’s lives.

The question then becomes, What do we want to learn from computers, and what do we want to learn about computers? If we imagine the computer as an interactive textbook, what dimensions of the textbook do we want to imbue the computer with, and what is possible to do in the next ten years? If we see the computer as an expressive medium, in which the computer as interactive textbook is one dimension, we still must respond to the question, How do we demonstrate the computer’s creative potential in a way that can be generalized to different settings? We might see the computer as a bold learning environment capable of dispensing information about a wide range of topics in different teaching styles that we can directly influence through
our interactions, changing our roles from teacher to learner back to teacher. The computer with personalities, which can reflect our own needs, serves as an arbiter of knowledge from a variety of sources. This vision is a distant one, and one that needs hard work to build. It is a large group effort needing support, ideas, input from all kinds of educators including teachers, children, computer scientists, and psychologists.

I end this book with some questions. I do not question whether we can set up new learning centers; I know we can. I see centers in which everyone is a learner, a teacher, a theory builder, and a theory tester. In this respect the centers provide a dramatic and perhaps revolutionary change in schooling. Learning is not seen as a passive process in which the teacher (human or computer) pours knowledge into the student’s head. The learning process becomes a shared responsibility among all the participants. But some of the major research questions in these centers are, What are appropriate activities and what kind of knowledge do we expect each to bring to the learning process and take away as well? For example, what kind of mathematical activities do we see children engaged in? Do they really have to become skilled calculators? Does knowing about number and being able to solve problems require calculating dexterity, or is it sufficient to be able to intuit number relationships and use calculating devices for precision and speed?

Will children in this kind of learning environment learn to read and write and do mathematics in a significantly more improved style? Will they be better learners in general? Will they be more aware better prepared citizens? Will what is done in these centers spin off to other environments and thus be reproducible? We need to do research to answer these questions. This center will provide a forum for this research.

Notes

Chapter 2
1. The computer was initially shared with John McCarthy’s Artificial Intelligence Laboratory; eventually both laboratories acquired their own computers for research.
2. The original programs were written in ALGOL, the high-level programming language then available on the system. Later the logic programs would be written in LISP, a high-level language that required a large amount of computer workspace but permitted easier and more powerful programming methods.
3. The computer chosen for this purpose was Data General’s Nova line. More recently, CCC has been using a 68000 processor in its package.

Chapter 3
1. Historically, this group descends from TICCIT, an unsuccessful rival to Plato as a delivery system. It was difficult to debug and redo the material.

Chapter 5
1. Various versions of Sprite Logo on machines including Texas Instruments’ 99/4, the Coleco Adam computers, and the Apple II with a hardware extension have about thirty turtles. All can be commanded to move and turn at the same time. Other computers, such as the Atari 800 and the Commodore 64, have fewer turtles. These turtles take advantage of color graphics and can write in different pen colors. On the Apple Macintosh, some Logo implementations support many turtles in the same window, whereas other versions support many windows, each with its own turtle.
2. We see traces of this tradition in Chomsky’s syntactic theory of language.
3. Actually, Papert has been a critic of standardized testing as a way of determining people’s intelligence or educational achievement. Although he might agree that standardized tests indicate something about people’s perceptions of particular subject areas, he does not think that these tests are a measure of a person’s understanding. But, although he vigorously opposes standardized tests and grades, he still expects that children in a Logo computer culture will show improvement in their standardized tests in reading, language arts, and computational skills.

Chapter 6
1. The complete set of six reports were issued during the period from April 1983 to
November 1984. Principals from 2209 public, parochial, and private schools in the sample responded to questionnaires and telephone interviews. Additionally, there were 1082 microcomputer-using elementary and secondary schools for which a computer-using teacher responded to an eighteen-page questionnaire.

2. More than half of the responding secondary school teachers saw computers used for programming or literacy. In secondary schools there was an increase in the number of teachers from subject areas outside of mathematics and science using computers in their teaching.

3. For example, in elementary schools with over 500 students, a mean number of 280 got access to computers with about 35 having daily access over a period of months. In secondary schools with more than 500 students (usually considerably larger), a mean number of 300 had access over the school year with a mean of 107 having daily access over a period of months.

4. The following software titles had the greatest school penetration in the 1985–86 school year: The Factory, Apple Logo, Apple Writer, Master Type, Microzine, Bank Street Writer, Snoop Troops, PFS:File, Typing Tutor, MECC Elementary Volumes.

Chapter 7

1. Martha later appeared on an educational television program discussing this project.

Bibliography


Johns Hopkins University, Baltimore, Md., July 29-August 2.


Dwyer, T. 1977b. "The eight hour wonder: All about BASIC in one long day, II." Creative Computing 3(5).

Dwyer, T. 1977c. "The eight hour wonder: All about BASIC in one long day, III." Creative Computing 3(6).


Second printing, 1987

© 1986 by The Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

The discussion on pages 148 to 160 is based on material that originally appeared in *BYTE* magazine. It is reprinted with permission from the August 1982 issue of *BYTE* magazine. Copyright © 1982 by McGraw-Hill, Inc., New York 10020. All rights reserved.

This book was set in Times Roman by E/F Typographic Lab and printed and bound by Edwards Brothers Inc. in the United States of America.

Library of Congress Cataloging-in-Publication Data

Solomon, Cynthia.

Computer environments for children.

Originally presented as the author's thesis
(doctoral—Harvard University, 1985).

Bibliography: p.

Includes index.


QA20.C65S64 1986 372.13'9445 86-3018

ISBN 0-262-19249-7

—

The MIT Press
Cambridge, Massachusetts
London, England