THE COMPUTING TEACHER

DECEMBER/JANUARY 1983-84 • VOLUME 11 NUMBER 5 • PRICE $3.50 U.S.

Photo by Doug Clements

The Journal of
The International Council for
Computers in Education

THE COMPUTING TEACHER

The Journal of The International Council for Computers in Education

December/January 1983-84
Volume 11 Number 5

Features

Look for the Senior Division problems and solutions for the 1983 International Computer Problem-Solving Contest in the February issue. To obtain a contest registration form, write D.T. Piele, ICPSC, University of Wisconsin-Parkside, Box 2000, Kenosha, WI 53141.

Departments

Index to Advertisers

Cover photo left to right: Elizabeth Hable, Christine Armontrout.

The Computing Teacher
University of Oregon
1787 Agate St.
Eugene, OR 97403 USA
503/686-4414

Editorial

Editor-in-Chief	David Moursund
Managing Editor	Dick Ricketts
Editor	Sharon Franklin
Associate Editor	Anita Best
Contributing Editors	Richard Adams
	Linda Ettinger
	Les Golub
	Mike Haney
	Bob Hilgenfeld
	Pat Konopatzke
	Kathleen Martin
	Tim Riordon
	Bob Shostak
	Bob Skapura
Book Review Editor	Regan Carey
Software Review Editor	Doris Carey
A-V Editor	Jerry Johnson

Production

Production Manager	Percy Franklin
Production Assistant	Tim Volem
Phototypesetting	ProtoType, Eugene, OR

Advertising

Marketing Director/ Advertising Manager	Sandi Standage

Staff

Office Manager/ Director, Data Processing	Kathleen Beisse
Circulation Manager	Howard Loewinger
Communications	Janice Smith
Accounting	Nancy Selvester
Data Entry	DeLoris Pape
Distribution Manager	David Ouellette
Conference/Outreach	Lynn Grimes
Leave of Absence	David Lang

THE COMPUTING TEACHER (ISSN 0278-9175) is published monthly 9 times during the academic year, August through May, for $21.50 per year by The International Council for Computers in Education, ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403. Second-class Rate paid at Eugene, OR 97401. USPS #660-130.

POSTMASTER: Send address changes to THE COMPUTING TEACHER, University of Oregon, 1787 Agate St., Eugene, Oregon 97403, USA.

THE COMPUTING TEACHER

December/January 1983-84
Volume 11 Number 5

The Journal of The International Council for Computers in Education

PURPOSE

The Computing Teacher is for persons interested in the instructional use of computers at the precollege level. The journal emphasizes teaching about computers, teaching using computers, teacher education, and the impact of computers on curriculum.

MEMBERS

The International Council for Computers in Education, ICCE, a non-profit educational organization, has two types of members:

INDIVIDUAL MEMBERS—Each person who subscribes to *The Computing Teacher* is a member of ICCE.

ORGANIZATION MEMBERS—Professional organizations working to enhance the instructional use of computers at the precollege level can become organization members at no charge. For further information contact David Moursund, Editor-in-Chief, *The Computing Teacher*, University of Oregon, 1787 Agate St., Eugene, OR 97403, USA.

AUTHORS

Articles and programs are solicited in all areas having to do with the instructional use of computers at the precollege level. Authors will not receive payment for publication of their material.

Submit two copies of the manuscript, neatly typed and double spaced, including qualifications, address and phone number of the author(s). For *Guidelines for Submission of Articles* or to submit a manuscript, write The Managing Editor, *The Computing Teacher*, University of Oregon, 1787 Agate St., Eugene, OR 97403, USA.

ADVERTISING

The Computing Teacher carries paid advertising, as well as exchange ads with other publications. Advertisers should be aware that the editorial content of the publication is determined by the editor. Some comments and articles may be unfavorable to a product or publication. The publisher reserves the right to reject advertisements. For rates and copy deadlines or a media kit, write Sandi Standage, Advertising Manager, *The Computing Teacher*, University of Oregon, 1787 Agate St., Eugene, OR 97403, USA.

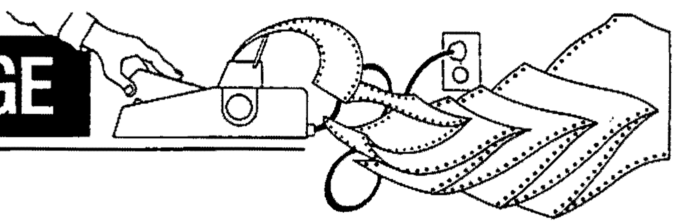
ICCE MEMBERSHIP

	U.S.	Foreign (Surface Mail)
1 yr. (receive 9 issues of TCT)	$21.50	$25.00 (US Funds)
2 yrs. (receive 18 issues of TCT)	$40.00	$47.00 (US Funds)
3 yrs. (receive 27 issues of TCT)	$58.00	$68.00 (US Funds)

Send membership dues to ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403, USA. (A $2.50 handling and billing charge will be added if payment does not accompany your dues.) Visa and MC accepted.

The Computing Teacher is published 9 times annually, August-May, by The International Council for Computers in Education (ICCE). Opinions expressed in the publication are those of the authors and do not necessarily reflect or represent the official policy of ICCE.

EDITOR'S MESSAGE



David Moursund

Logo Frightens Me

I first encountered Logo about ten years ago, and even then I was quite impressed. Since then I have frequently supported Logo in my writing, teaching and public speaking engagements. ICCE has supported Logo through a column in *The Computing Teacher*, and this is the second Logo issue of TCT. ICCE is publishing a book on Logo (See "Logo in the Classroom—Session 1," p. 67).

I have learned quite a bit about Logo and its uses, and I encourage all of my computers in education students to do the same. Recently, I co-directed a doctorate thesis that centered on teaching teachers to teach students to use Logo.

On the surface I am a strong supporter of Logo. But deep down in me there is some fear associated with Logo's role in education.

My fear has two parts. First, I fear that Logo is being oversold. Some people are developing unreasonable and unrealizable expectations about what Logo can do for education.

Second, I fear that Logo will not reach its potential. Understanding the Logo phenomenon is difficult. It is accompanied by an almost-religious enthusiasm. In talking with many Logo-oriented educators, I am led to believe that Logo not only will make their students computer literate and substantially improve their problem-solving skills, but will make a major contribution to rectifying many of the current ills of education. These claims may prove to be true, but it is important to acknowledge that, to date, such deeply-held beliefs in Logo go largely unsubstantiated. A number of my graduate students have done careful surveys of the Logo literature, searching for solid research to back up the widely-voiced claims. The literature is sparse. It consists mainly of descriptions of teachers using Logo with students, most concluding that students enjoyed using Logo to draw pictures. One could say the same thing about students provided with a set of paints and a brush.

That is not to say there is no research on Logo. The Brookline project, for example, gave us a strong hint of Logo's potential. But one must view with suspicion an experiment in which the elementary school teacher has a doctorate in engineering. Indeed, few of the so-called experiments have been done making use of "ordinary" teachers—those with a very modest level of training, experience and interest in the computer field.

The very heart of much Logo-based instruction is what is often called discovery-based learning. Discovery-based learning has been extensively researched and has substantial merit. But its effective implementation requires well-trained, skilled, committed teachers. Logo by itself cannot create the teacher-related parts of a sound educational environment.

Teachers often equate a minimal level of success in using Logo with students becoming computer literate and becoming much better problem solvers. This reflects very little depth of insight into the various components of computer literacy. It reflects almost no insight into problem solving or into potential roles of computers (or Logo) as an aid to problem solving.

It feels to me like Logo has been oversold. Marketing experts have done their job, but that isn't what has oversold Logo. Educators have done it to themselves. In looking for "the answer" in computing, these educators have latched onto Logo. It obviously is part of an answer, but transforming a partial solution into a panacea is damaging, both to education and to the potential of Logo.

Logo can be a powerful aid to learning. Perhaps your definition of computer literacy includes learning to write and debug programs to solve problems. Perhaps you want your students to develop insight into top-down analysis, stepwise refinement and problem or program segmentation. Perhaps you would like your students to study aspects of a specific content area through a discovery approach.

Logo is an excellent vehicle to achieve these goals. But not without the help of a knowledgeable teacher and suitable curriculum materials. Logo by itself will do little for most students. Certainly there will be rare exceptions—students who learn Logo and explore concepts on their own to create interesting and challenging projects. But most students require knowledgeable and experienced teachers, as well as good curriculum materials.

Now we are at the very heart of my fear. Logo is a wonderful language, but a Logo-equipped computer system is not a teacher-proof educational tool. Most teachers require a substantial amount of computer education and experience to even begin to help their students to realize the potentials of a Logo computer system. And that is only a beginning. What happens to such students in the months and years that follow, as they have continued access to computers? Where will they receive guidance and help in their endeavors to learn more of the potentials of Logo and computers?

For me, a pattern of answers is beginning to appear. Computers will have a profound impact upon education. Eventually more of the needed research will be done, so we will have increased knowledge of ways to use computers effectively.

If we are willing to settle overall for a mediocre education, decreased reliance upon teachers and increased reliance upon computers can help us achieve the goal. But if we have higher aspirations, such as students achieving their full potential, then highly qualified teachers will be more important than ever. Attracting and holding good teachers, providing them with high quality preservice and inservice education, supporting them with appropriate resources—these are keys to improving our educational system. I believe in the potential of Logo, but I believe much more strongly in the part educators will play in its effective use. This Logo-oriented issue of *The Computing Teacher* gives good evidence of the progress being made.

END■

ICCE Special Interest Groups

At its October 9, 1983, meeting, the ICCE Board of Directors voted unanimously to authorize the creation of Special Interest Groups. It also voted unanimously to authorize the immediate establishment of a *SIG Bulletin*. These two actions have profound implications for the future of ICCE. These decisions commit ICCE to a new level of professional activity. Regional, provincial and state computer education organizations can continue to cooperate geographically within the current ICCE structure. ICCE SIGs will facilitate cooperation among computer educators with a specialized interest or set of needs.

Within the computers-in-education field at least three groups have been identified as having new and challenging responsibilities. To date, there has been no forum for these educators to clarify their new roles, receive and exchange pertinent information, and establish and build their professional identity. To address these needs, ICCE has taken the initial steps in organizing three groups:

1. **A SIG for Computer Coordinators.** This SIG is for people who have building-level or district-level responsibility for computers in education. (See "Computer Coordinator Group: A Proposed ICCE Special Interest Group." *The Computing Teacher*, Vol. 11, No. 3.)
2. **A SIG for Teachers of Educators.** This SIG is for people who develop and teach courses and workshops for inservice or preservice educators. (At this point most computer coordinators are also responsible for teacher training.)
3. **A SIG for Administrators.** This SIG is for school administrators who need considerable computer knowledge to make effective decisions about school management and the integration of computers into the curriculum.

General guidelines for the establishment of SIGs are being formulated by a committee designated by the ICCE Board of Directors. A SIG is much like a self-contained, special-purpose professional society, with officers, a board of directors, a publication (which might eventually be distributed electronically), and meetings (such as professional development seminars). The volunteer editorial staff of each SIG will be responsible for the content and preparation of their publication. The ICCE headquarters staff will support SIGs by collecting SIG membership dues, maintaining membership records, and aiding in the production and distribution of all SIG publications.

Initially, one *SIG Bulletin* is being created to serve all the newly emerging SIGs. Eventually, individual SIG publications will replace this combined *SIG Bulletin*.

The *SIG Bulletin*, a quarterly publication, will carry information about the SIGs that are now forming. It will contain topics of interest to each SIG with a focus on:

- Articles of immediate importance and usefulness
- Surveys of the research
- Editorial comment to spark debate
- A forum for question/answer and the sharing of ideas

Information will be included on how to get involved with a SIG and how to help create a new SIG.

The first quarterly issue of the *SIG Bulletin* will be available in March 1984, with subsequent issues following approximately every three months. Each quarterly issue will be 50 pages or more in length.

A special introductory issue of the *SIG Bulletin* is scheduled for distribution in mid-January 1984. This **preview** issue will be 24 pages in length and will be available **free upon request**. It will invite you to become a paid member of a SIG and receive the *SIG Bulletin*. For ICCE members, a subscription to the *SIG Bulletin* will cost $10 per year.

ICCE has received a donation of $2100 to help establish the *SIG Bulletin* and to defray some of the initial expenses in starting the SIGs. It is expected that memberships will eventually allow the SIGs to be self-sustaining.

If you would like a copy of the **free introductory** issue of the *SIG Bulletin*, fill out and return the postcard included in this issue (or simply send us your name, address and position). The postcard also asks for your ideas: What would be most useful to you? What areas would you like more training in? What articles or ideas can you share with others? Your support, ideas and time are not only encouraged—they are needed. Write for a free copy of the *SIG Bulletin*.

END■

LETTERS TO THE EDITOR

Dear Sir:

My husband, who is a professor of medicine, had a cerebrovascular accident nine months ago and has expressive aphasia and a right paralysis. He can read and recognizes some words, but cannot initiate spoken or written language. He has no problem understanding spoken language, and his reading comprehension seems to be progressive.

I am looking for a program which will provide him with a vocabulary broken down by categories, subcategories, etc., which will enable him to select key words. I am also looking for learning programs to help him regain vocabulary and sentence structure.

We plan to use the IBM PC with the quadrum Apple II interface so that we can use programs written for either. I am familiar with the Words+ program, but because this requires alphabetical recognition, I do not think it is useful at this time.

If you have or know of any program which we might be able to use, I would appreciate your contacting me.

Sincerely,
Victoria R. Liddle
770 Norwood Drive
Nashville, TN 37204

Dear Ms. Franklin:

Thank you for sending me *The Computing Teacher*, Sept. 1983, Vol. 11 #2, which included a review of my book *Alas Para la Mente (Wings for the Mind)*.

In response to your question about distributors, the book may be obtained through: Ms. Teresa Malwer, Lectorum Publications Inc., 137 West 14th St., New York, NY 10011, 212/929-2833.

The book is being translated into French (title: *Des Ailes Pour L'Esprit*) by Nathan-Cedic, Paris; and into Italian (title: *Ali Per la Mente*) by Mondadori, Milano.

With regards to TI Logo software in Spanish, contact Mr. John D'Angelo, Manager, Education Strategy Support, Texas Instruments Inc., P.O. Box 225012, MS 84, Dallas, TX 75265, 214/995-4184.

Yours truly,
Horacio C. Reggini
Av. L.N. Alem 1074 — Piso 1°
1001 Buenos Aires, Argentina

Dear Editor:

We will be one of the public schools attempting the new AP Computer Science course this year. We are fortunate to have a high percentage of very able students that can take advantage of this new offering. We look forward to the challenge of presenting meaningful advanced topics to these bright young adults. But at this time I am not satisfied that the AP Computer Science course as presented in the course description booklet is what our students need for their future.

The future computer scientist should be exposed to a wider range of topics at this first level. Our students are not served well with a course that treats high level programming methods and skills to the exclusion of topics like 1) Introductory logic, logic circuits and Boolean logic; 2) Machine level programming and use of an assembler; 3) CPU architecture, control, timing and functions; 4) Operating systems and ROM functions; 5) Memory mapping, flags and pointers, etc.; and 6) Word processing and use of printers.

Educators throughout the country do not agree upon the one course that will meet student needs at this introductory level. I feel that this introductory course in computer science should reflect the future, if not the present, revolution created by microcomputer technology. In some respects this new AP

Course represents an obsolete, and unnecessarily restricted, introductory course. I believe that other educators see this as one of its weaknesses. Consider:

1. Educators should encourage student creativity, flexibility, analysis and synthesis more than in the recent past. Creativity and imagination are discouraged with the use of a programming language that requires unnecessary structure.
2. The advantages of a low-level programming language should be demonstrated and integrated with high-level programming.
3. One constant in modern computer technology is that change is rapid. We must encourage students to be willing and able to take advantage of changing technology. This suggests more than acquiring programming skills in one language.
4. If we use Pascal, hardware costs require unrealistic budgets for many schools. This is unfortunate in light of the fact that educational objectives can be met today with less expensive systems.
5. How many real-world computer systems use Pascal outside of educational applications? You will find a low percentage of Pascal users for good reasons. Can we be sure that Pascal will be the "best" language to teach within three years?
6. Good programming practices and structured programming can and should be taught through any language. They are not *unique* to Pascal.
7. Programming today and tomorrow will be much more than math oriented as ETS suggests it is now.

I would appreciate hearing from other teachers involved in teaching the AP Computer Science course. What are you doing? How do you see the problem? Maybe we can share our concerns and materials.

Richard King
Staples High School
Westport, CT 06880

Dear Ms. Franklin,

Regarding the program contained in my article, "Putting Logo to Work" (TCT Vol. 11 #2, September 1983), the following should be noted.

In the PRINT.LETTER procedure, put these three lines in the beginning:

```
PRINT [ DO YOU HAVE A PRINTER
  CONNECTED? (Y/N ]
MAKE "ANSWER READCHARACTER
IF :ANSWER = "N LETTER CLEARTEXT
  SETUP1
```

Also in this procedure, check the OUTDEV 3 line carefully. If your printer interface card is not in slot 3 your program will crash. Be sure to put in the correct slot number after this first OUTDEV. Most people keep their printer interface card in slot 1, so this line should read OUTDEV 1.

In the HLS procedure, be sure to have a space between the two single quotes.

In the BODY2 procedure, add these two lines after the last line (PRINT :TEACHER'S.NAME)

```
OUTDEV 0
PRESS.RETURN.TO.CONTINUE
END
```

I welcome any comments about this program from TCT readers. Since the publication of the article, many improvements have been made, and I would be happy to share them with anyone interested.

Yours sincerely,
Hillel Weintraub
Doshisha International High School
Tatara, Tanabe-cho, Tsuzuki-gun
Kyoto-fu, Japan 610-03

Dear Sirs:

I am a teacher and co-director of the intramural program at Hadley Junior High School in Glen Ellyn, IL. I am searching for a computer program to assist in the planning and organization of our intramural tournaments which include single elimination, double elimination and round robin tournaments. We have Apple II computers and one eighty-column printer. Any assistance you can give will be greatly appreciated.

Sincerely,
Michael Dorich
William M. Hadley Junior High School
240 Hawthorne St.
Glen Ellyn, IL 60137

Dear Dr. Moursund:

I am the principal of a large elementary school, part of a twenty-one school division. I am interested in knowing the names of any districts in the United States or Canada that are successfully using the microcomputer as an administrative tool both at the individual school level and at the district-wide level where schools may communicate directly with the central office or with each other. We have been using the microcomputer for CAL and now wish to use it for administration purposes.

Yours sincerely,
R. T. Blair, Principal
George Fitton School
1129 Third Street
Brandon, Manitoba, Canada R7A [illegible]

END■

WHAT'S NEW?

Logo for Atari

Atari Logo features collision detection, animation and four dynamic graphics "turtles." The unique features of Atari Logo include 128 colors available simultaneously on one screen and multi-voice sound capabilities. Joysticks and paddle controllers can be used with Atari Logo.

Logo users can create programs that converse in recognizable words. Such programs allow beginners to manipulate shapes on the screen the first time they use Logo.

The Atari Logo program cartridge works on all Atari Home Computers and requires as little as 16K RAM. The complete package, including a reference guide and two 200-page manuals, *Introduction to Programming Through Turtle Graphics* and *ATARI Logo Reference Manual*, has a suggested retail price of $99.95. Institutional customers may purchase manuals and program cartridges separately.

Young Peoples' Logo Association

The YPLA has gone international through new affiliations with organizations in Australia, the Netherlands, England and Japan. YPLA's monthly magazine, *Turtle News*, will now be published in different languages. And, of course, news of Logo from around the world will be reported through the YPLA newsletters.

The first two books from the YPLA are now in circulation. New editions of *1,2,3, My Computer and Me* will soon be appearing for Atari Logo and Commodore Logo. Following these will be a series of books, *The Adventures of Logy and Mof*, the turtle and rabbit mascots of YPLA. The first of these will be curriculum-oriented activities for kindergarten; the second will feature on- and off-computer activities for elementary grades.

To further the effective use of computers by the handicapped, the YPLA has joined forces with a new organization, the Center for Computer Assistance to the Disabled. While C-CAD is a new organization, it already has been able to place ten computer systems to train physically or mentally-disabled clients. It has also developed a substantial list of resources for those seeking information on computers in special education and training. The board includes a number of special education teachers and faculty members from local universities.

YPLA, 1208 Hillsdale Dr., Richardson, TX 75081.

The National Logo Exchange

The National Logo Exchange is a monthly newsletter which provides practical ideas and tips for teachers who use the powerful Logo computer language in their classrooms. Published September through May and mailed first class, *The National Logo Exchange* acts as a forum for sharing Logo ideas, teaching techniques and philosophies. Now in its second full year of publication, the NLX features articles by classroom teachers, columns by well-known professional educators, reviews and reports of the latest Logo versions and resources, and helpful Logo tips. $25 per year for USA, Canada and Mexico; $30 per year elsewhere.

For information, contact *The National Logo Exchange*, P.O. Box 5341, Charlottesville, VA 22905.

Do you feel as if you are alone in the Logo world? Would you like to be able to talk with Logo folks from other schools once in a while? Do you want to know someone nearby to call for Logo help? Then think about forming a Local Logo Exchange.

Local Logo Exchanges (LLX) are made up of Logo teachers and other workers who want to meet periodically to exchange ideas and share Logo activities in an informal setting. The NLX supports the spontaneous formation of such groups by furnishing suggested guidelines for organizing, giving publicity in the NLX newsletter, and helping to identify other Logo workers nearby as prospective members.

If you are interested in organizing such a group, then write to NLX Local Logo Exchange Program, P.O. Box 5341, Charlottesville, VA 22905. Tell us something about your situation and what you hope to do. We will send you a copy of the LLX Guidelines and suggestions for getting started.

The National Logo Exchange is compiling a directory of all persons conducting Logo-related research. If you would like to be listed, send your name, address, telephone number, position and a brief summary of your research interests and activities. Copies of the directory will be made available in mid-1984. Mail your information to Research Directory, Attn: NLX, P.O. Box 5341, Charlottesville, VA 22905.

If you are teaching a Logo course at the college or university level during the summer of 1984, please send information to the NLX as soon as possible. Our readers would be interested in the location, dates, number of hours, prerequisites, cost and general course content. We will include your course announcement in our spring 1984 issues. Send information to: 1984 Logo Courses, Attn: NLX, P.O. Box 5341, Charlottesville, VA 22905.

Coordinated Research on Computers and Young Children

No media revolution of the past has fulfilled the promises of its proponents concerning its educational potential. Research on each new innovation has often been less than helpful, as it often fails to build on previous studies. The potential of computers to enrich young children's environments necessitates that those working in this area coordinate their efforts through communication, symposiums, and cooperative research efforts to maximize both the quality of this work and their own effectiveness as child advocates. Any professionals who are conducting or planning to conduct research concerning young children and computers and who would like to cooperate in such infor-

mation sharing are invited to contact Douglas H. Clements, Early Childhood Department, 300 White Hall, Kent State University, Kent, OH 44242.

Commodore and MECC Agreement Reached

Commodore Business Machines, Inc. has reached an agreement with the Minnesota Educational Computing Consortium (MECC) to convert and market MECC software for the Commodore 64.

For more information write MECC, 2520 Broadway Dr., St. Paul, MN 55113.

55,765 Schools Now Teach with Computers

A just-completed survey of U.S. school districts has determined that 55,765 public schools now use computers in instruction. That is more than double the number using computers one year ago. Both surveys were conducted by Market Data Retrieval, Westport, CT, which phoned all 15,275 U.S. school districts between July 1 and September 15.

Key findings of the survey were:

1. 68% of all schools have computers; 62% of elementary schools; 81% of junior highs; 86% of senior highs.
2. There are 325,000 microcomputers in U.S. public schools; 110,000 in elementary schools; 55,000 in junior high schools; 135,000 in senior high schools; and the balance in K-12 and special ed schools.
3. The number of computers in a school varies with the grade level; the average high school has 11 computers; the average junior high has 7; the average elementary school 3.5.
4. Apple, Radio Shack and Commodore are the leading brands of computers used in U.S. schools. Schools are using 160,000 Apples, 68,000 Radio Shacks and 49,000 Commodores. Next in order of units used are Atari, IBM and Texas Instruments.

A major concern of educators has been the disparity between small, low-spending schools and larger, higher spending schools in the use of microcomputers," stated Sharon Sanford, MDR's Director of Research. "Last year our survey showed that 80% of the 2,000 largest, richest high schools used computers for instruction, while only 40% of the smaller, poorer high schools did. The surge in buying during the last year has helped the poorer schools to start catching up," Sanford noted. "By the end of this year we expect every school in the country to have at least one microcomputer for school instruction. It remains to be seen whether these schools with one or two microcomputers will ever catch up with the hundreds of high schools that already have 25 or more micros."

PC Owners Can Now Access PLATO Network

Control Data Corporation announces PLATO MICROLINK, a new software product enabling consumers direct access to selected portions of the PLATO on-line library.

PLATO MICROLINK is currently available for use with the IBM Personal Computer.

Personal computer owners will have to purchase an access disk ($50) containing PLATO interface software and will pay both a $10 registration fee and $5 an hour for using the PLATO system. At present, users will be able to access more than 150 PLATO courses, games and services from 6 p.m. to 3 a.m. weekdays, and 8 a.m. to 3 a.m. weekends and holidays.

An agreement between Source Telecomputing Corporation and Control Data Corporation will allow subscribers to THE SOURCE to purchase PLATO MICROLINK services at a reduced cost.

For more information, contact Susan J. Busch, Public Relations Dept., Control Data Corporation, Box O, Minneapolis, MN 55440, 612/853-6605.

School & Home CourseWare, Inc. Down Under

Liberty (Imports) Australia is now an exclusive distributor for educational microcomputer products manufactured by School & Home CourseWare, Inc. for TRS-80, PET and Apple diskette or cassette systems.

The *Digest of Software Reviews: Education; HARTS III Administrative System; School CourseWare Journal;* and S&HC's Software Library Subscription Series will now be available for the first time in Australia and New Zealand.

For information in Australia and New Zealand, write to Robin F. Hancox, Liberty (Imports) Ltd., Gold Coast Mail Centre, Queensland 04217, Australia, (075) 532 411. Or, contact Bill Botzong, School & Home CourseWare, Inc., 1341 Bulldog Lane, Fresno, CA 93710, USA, 209/227-4341.

Adventure User Group Formed

Maurice Dow has formed an international Adventure User Group for microcomputer users. Members will maintain contact through a monthly newsletter. Special emphasis will be placed on the use of adventures for teaching specific topics.

For more information contact Maurice Dow, 84 Camberley Cres., Brampton, Ontario, Canada L6V 3L4, phone 416/451-9452.

Commodore Donates Computer Systems

Commodore Business Machines, Inc. recently donated 120 computer systems to the State Departments of Education in California, New York, Pennsylvania and Texas. The systems include computers, data storage units, printers, modems and educational software.

The computer systems will be distributed to educational support centers where they will be used for hands-on in-service teacher training and for evaluation of instructional software.

Commodore dealers in the area of each training center have agreed to provide support for the donated units and training for program coordinators, who will instruct teachers.

New Journal Scheduled

Education & Computing, an international journal, is scheduled as a quarterly publication of 90 pages per issue for $61.50 (US). The first issue will appear in 1983. The journal will explore theory, applications and research, with emphasis on the linkages among them.

For a free copy, write to Elsevier Science Pub. Co., Journal Information Center, P.O. Box 1663, Grand Central Station, New York, NY 10163 (USA and Canada); or Elsevier Science Publishers, Subscription Order Dept., P.O. Box 211, 1000 AE Amsterdam, The Netherlands. END■

NEW FOR YOUR CLASSROOM

Newest additions to the best-selling I Speak BASIC series

I Speak BASIC to My Commodore 64™
I Speak BASIC to My TI-99/4A™

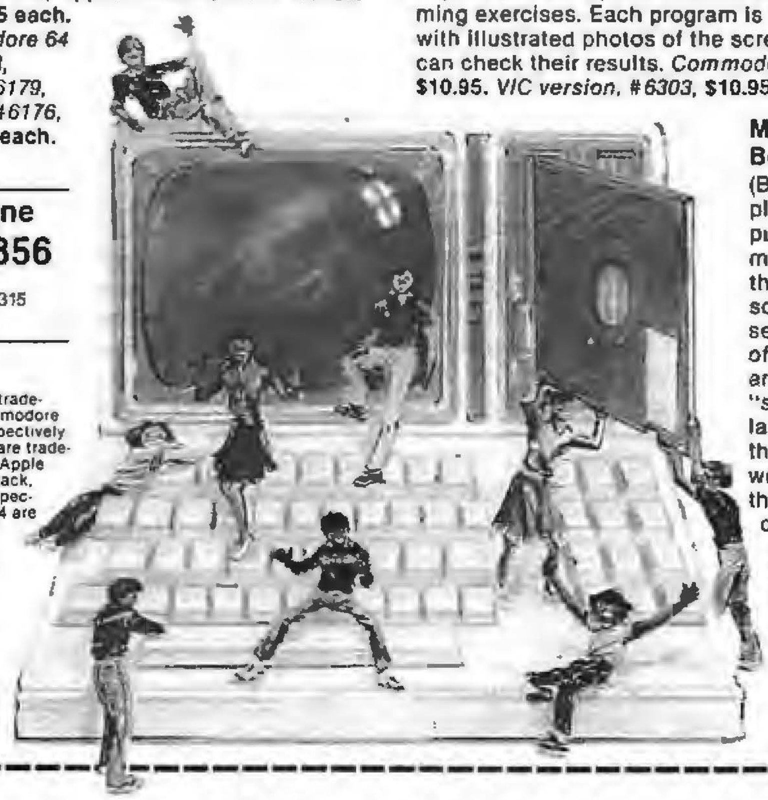
(Jones) Field-tested computer literacy courses that introduce students (and teachers) to BASIC language programming. No previous computer experience is required. Texts are machine specific for the Commodore 64 and TI-99/4A and provide a complete understanding of the machines. The Student Texts begin each lesson with learning objectives, followed by definitions of key terms, programming examples, in-class exercises, assignments, a summary, programming practices, and a quiz.

All of these best-selling I Speak BASIC titles are available: *Teacher's Manual: Commodore 64 #6162, TI-99/4A #6163, Atari® #6168, VIC™ #6169, Apple™ #6165, PET® #6166, TRS-80™ #6164.* **$18.75 each.** *Student Text: Commodore 64 #6172, TI-99/4A #6173, Atari® #6178, VIC™ #6179, Apple™ #6175, PET® #6176, TRS-80™ #6174,* **$9.75 each.**

Explorer's Guide to Logo
(Von Mertens and Webb) This fascinating guide leads children on an exciting "treasure hunt." The treasure is computer literacy for children—learning Logo and microcomputer operations. Introduced are pre-computer activities, Logo commands, hands-on examples, and imaginative games. Comes complete with teacher's manual and activities on spirit masters. *Apple™ version: #6226 teacher's manual* **$3.95**; *#6227 student text* **$12.50.** *MIT version: #6231 teacher's manual* **$3.95**; *#6232 student text* **$12.50.**

The Commodore 64™ Picture Book
The VIC™ Picture Book
The TI-99/4A™ Picture Book
(Nadler) A unique "picture book" introduction for first-time computer owners. These books explain how to unpack and set up the machine and offer easy programming exercises. Each program is "pictured" in the book with illustrated photos of the screen against which users can check their results. *Commodore 64 version, #6453,* **$10.95.** *VIC version, #6303,* **$10.95.** *TI version, #6408,* **$10.95.**

Microcomputers Can Be Kidstuff
(Burke) Makes "child's play" out of learning computers—from the excitement of game playing to the challenges of problem solving. The author presents clear explanations of hardware and software and prepares readers to "speak" BASIC and Pilot languages; she then covers the basic techniques of writing programs, saving them on diskettes or cassettes, and using commercial software. Includes an easy-to-use glossary and a checklist for using the computer. **#5202, $11.95.**



Order by Phone
1-800-631-0856
operator CTD3
In NJ call (201) 393-6315
Book Sales Dept.

Atari and PET are registered trademarks of Atari, Inc., and Commodore Business Machines, Inc., respectively. TI-99/4A, Apple, and TRS-80 are trademarks of Texas Instruments, Apple Computer, Inc., and Radio Shack, a division of Tandy Corp., respectively. VIC and Commodore 64 are trademarks of Commodore Business Machines. None is affiliated with Hayden Book Company, Inc.

Educators are entitled to a 20% discount and a FREE 30-day examination.

Mail to:
Hayden Book Company, Inc.
Dept. CTD3
10 Mulholland Drive
Hasbrouck Heights, NJ 07604

Please send me the book(s) indicated by code number. If I am not completely satisfied, I may return the book(s) undamaged within 10 days for a complete refund. I am **enclosing $2.00 to cover postage and handling.**
☐ Enclosed is my check/money order
Bill my ☐ Visa ☐ MasterCard

Name ____________

Position/Dept. ____________

School ____________ Purchase Order # ____________

Address ____________

City/State/Zip ____________

MasterCard/Visa ____________ Exp. Date ____________

Signature ____________

Residents of NJ and CA must add sales tax. Prices subject to change.

FREE: the educator's carrying case with your subscription to:

MEDIA & METHODS

America's Magazine of the Teaching Technologies

Treat yourself to the very best...a full year of **MEDIA & METHODS** for just $18...plus the Educator's Carrying Case free of charge...It's time you had your own copy of **MEDIA & METHODS** to use and use all year long.

Order **MEDIA & METHODS** now and save...Enter a new one-year subscription...or send your renewal...or extend your current order and pay only $18...You save $9 off the regular cover price...and you get the extra large, attractive Educator's Carrying Case designed to meet your special needs free of charge.

SUBSCRIBE NOW MEDIA & METHODS

the most widely read source for educational media in the 80s

- **practical, instructional strategies—the complete and informative guide to computer-based teaching technologies.**
- **innovative, teacher-tested projects plus computer-related study guides that enhance curriculum experiences.**
- reviews and reports on books, computer hardware and software, films, AV equipment, filmstrips, media management...and more...

MEDIA & METHODS
"...the most exciting magazine in education today."

THE NEW YORK TIMES

Mail to:

MEDIA & METHODS
1511 Walnut St.
Philadelphia, Pa. 19102

Please send me the free Educator's Carrying Case and

☐ Enter my one-year subscription to MEDIA & METHODS for only $18... (that's $9 off the cover price)

☐ Extend my current subscription for 9 more issues at $18.

Name: ____________________
Title/position ____________________
School/Library ____________________
Address ____________________
City, State, Zip ____________________
☐ Payment enclosed ☐ Bill me
☐ Bill school/library Purchase Order # ____________

Charge my credit card:
☐ Mastercard ☐ Visa
Card # ____________ exp. date ____________
Signature ____________________

Please allow 4-6 weeks for delivery of first issue. Add $5.00 postage for subscriptions outside the U.S.

Classroom Management for Logo

by
Shirley Torgerson

The coming of Logo brings a mixture of excitement and hesitancy to many teachers in self-contained classrooms. It is exciting to contemplate all the ways Logo enhances the curriculum and helps develop concepts and skills. The hesitancy begins when the day actually arrives.

We at Paxson were faced with one computer for 20-30 students.[1] Our experience may be helpful to other teachers using Logo in the classroom for the first time.

A discussion of classroom organization and management techniques using Logo in an elementary classroom is incomplete without a rationale for choosing Logo as the principal use of a computer.

Elementary teachers are well aware of the importance of a student's early years in establishing thinking patterns and approaches to solving problems. D. Benjamin Bloom, University of Chicago, stated in a summary of 1000 research studies:

> Put in terms of intelligence measured at age 17, from conception to age 4 the individual develops 50% of his/her mature intelligence; from age 4 to 8 s/he develops another 30%; and from ages 8 to 17, the remaining 20%.[2]

Basic habits and skills established in the elementary grades have great impact on students' success in school later on. They develop individual styles of thinking through their actions on the environment and from the feedback they receive. A Logo environment encourages the development of these skills. Young children learn spatial relationships first through their own bodily involvement in spatial movements. The turtle provides an object for extending these concepts into complex movements that go beyond children's own physical abilities, at the same time allowing children control over the action. Such control allows the child to repeat actions for the feedback necessary to assimilate new concepts into his/her own mental structures.

Logo provides an activity-based setting for developing important and sometimes difficult-to-teach concepts such as sequencing, patterning, estimating, probability, measuring, graphing, coordinate systems, simplifying, language arts activities, communication skills and geometric concepts. It is a powerful, motivating medium that fosters development of thinking skills and problem-solving strategies.

Logo became an integral part of our classroom after we had answered seven questions. The immediate question was:

WHERE WILL I PUT THE COMPUTER?

The computer should be placed to serve the needs of the class and of each student user.

Privacy

Think back upon your first contact with a computer. You might recall your hesitancy because others were watching. Children as well as adults are sometimes unwilling to risk trying something new in the presence of others for fear of making a foolish error. Others may simply be so conscious of classmates observing them that they cannot concentrate on their own task. Watching the Logo action on the screen is also tempting for those working on other assignments. Placing the computer in a corner of the room allows for a degree of privacy while permitting the user to hear important announcements.

Access for Demonstrations

The computer will sometimes be used for demonstrations such as teaching a new primitive or explaining a subject matter concept. The computer should be located where it can easily be turned, without dismantling, for good whole-class viewing.

Glare

When considering placement of the computer, consider glare from windows. A screen that faces a window reflects the light, making it difficult to see the screen image. Likewise, if the viewer faces a window, it is difficult to look at the darker image.

Once the computer is set up in a good location, Logo is ready to become an integral part of the classroom.

HOW DOES LOGO FIT INTO THE DAILY SCHEDULE?

One of the first questions entering a teacher's mind is usually, "How do I find time to add one more thing to my crowded schedule?" In recent years, schools have assumed so much responsibility for students' total education, development and socialization that instructional time is precious. However, a half-hour Logo lesson now has the potential for saving a child hours later on in school. Geometry concepts, language arts skills and problem-solving strategies can be introduced and developed while giving students the opportunity to become "computer literate." Rearranging the daily schedule once a week may yield the needed half-hour. Considering the correlation with mathematics and language arts, perhaps Logo could use fifteen minutes from each once a week, or students may agree to extra homework one night a week to compensate for the "lost" time. A half-hour, whole-class lesson presented once a week will introduce students to the Logo primitives and some beginning programming constructs. As they become proficient in using the language, Logo can become a tool for learning subject matter, developing skills and reinforcing concepts.

HOW DO I CYCLE CHILDREN ON TO THE COMPUTER WITHOUT LOSING VALUABLE INSTRUCTIONAL TIME?

Most teachers have some kind of system using sign-up charts or schedules for individualized activities. At Paxson, we use a TIME-ON chart, so the computer can be used most of the day. The 22"x28" poster board chart is divided into five columns, one column for each school day, with horizontal lines representing half-hour time slots. The chart is laminated so student names can be entered with water-based markers and easily changed from week to week.

The half-hour slots are marked on the left side of the chart. Time slots not available for computer use (library time, gym classes, music class, field trips, introduction of new concepts, etc.) are marked off. On Monday, each student selects one time slot. After all students have selected one time, a second turn is allowed—a procedure which gives all children a chance at selecting the preferred computer times. Since students occasionally miss their designated time due to illness or unexpected interruptions, a few slots are left open on Fridays.

Depending on class size and length of the school day, this system should permit each student to have two half-hour slots per week with any extra time given to pairs of students. (In larger classes, two children might share the second block of time.)

A small clock by the computer lets the students know when their time is up. That user is then responsible for reminding the next student scheduled to use the computer. This helps minimize "clock-watching" by the next student and increases concentration on the current task. Students move quietly to and from the computer without disturbing the rest of the class or the teacher.

WILL KEEPING TRACK OF MISSED ASSIGNMENTS MEAN MORE RECORD-KEEPING?

A partner system frees the teacher from keeping track of who missed what during computer time. While one student works at the computer, his/her partner keeps track of assignments, turns in finished work and collects passed-out materials. If a laminated chart listing partners is used, they can be changed periodically. Students at Paxson also made construction paper folders in which to keep all papers and notes for their partners. It should be noted that partners are responsible for explaining minor tasks but not important concepts. The introduction of new ideas normally requires full class participation. Partners cannot share computer time, and a student at the computer cannot go to his/her partner for help. The partner system has an added benefit—it works equally well when students are absent from school.

WHERE CAN THE STUDENT AT THE COMPUTER GO FOR HELP WITHOUT DISTURBING THE TEACHER?

Resource charts, including a quick reference chart listing primitives with any abbreviations, are posted in the computer corner. This minimizes the need for help and encourages students to work independently at the computer. Charts serve as references for colors, keyboard characters, editing functions and hardware. An in-depth listing of already-introduced primitives is kept in a notebook on the computer table. This listing shows the correct way to type each command and explains in easy-to-understand language what the primitives can do.

Finally, some Logo screen images that students have created are put on 9x12 tagboard and displayed. This is one way for students to share ideas.

Selected peer helpers are an invaluable source of help. Peer "helpers of the day" should have mastered key concepts and be able to communicate well.

A Logo procedure TO HELP can be entered in the computer and changed to fit the current Logo assignment. When the student types HELP, the text-screen will print directions or explanations without losing the current screen image. (A sample help procedure is listed at the end of this article.)

HOW CAN I KEEP TRACK OF WHAT MY STUDENTS ARE DOING WITH LOGO?

Students need not keep written journals. Handwriting for young children is laborious and is a skill practiced in other learning areas. Students can be taught at the beginning to use diskettes to save and access their work. Their work is then readily available for the teacher to review whenever convenient. The teacher can also send special messages to the student. Paxson students know that they have a message from the teacher when they see a procedure entitled TEACHER in their file. By typing the procedure name TEACHER, the message appears on the screen. (See sample at the end of article.)

ARE RULES NEEDED FOR THE COMPUTER ENVIRONMENT?

A Logo environment provides an excellent arena for students to practice the self-direction that is needed as they become older and assume increased control over their daily lives. Students can exercise responsibility and develop a strong, positive self-image when their teacher encourages cooperation and sharing, has realistic expectations of each student, and allows students to set their own goals.

Since grading and testing are not integral parts of learning Logo, students are free to share and to use each other as resources. As an opening part of our Logo sessions at Paxson, students are given the opportunity to share their discoveries and problems. Such student interaction solves many problems and promotes skill in communication and self-guidance. Teacher participation in these conversations is usually needed only to clarify unknown primitives or provide detailed explanations of concepts that are unclear. Because there is no grading, the Logo experience is not a harsh, competitive activity, and students are eager to share and assist each other in problem solving.

Sharing diskettes introduces the idea of personal ownership of computer work and respect for the rights and property of others. Establishing an ethical code on the accessing of computer records should begin early in a student's computer experience.

```
TO HELP
 CLEARTEXT TEXTSCREEN
 PRINT [DO YOU NEED HELP WITH:]
 PRINT [ ]
 PRINT [THIS WEEK'S ACTIVITY?]
 PRINT [.....PRESS "1"]
 PRINT [ ]
 PRINT [THE NEW PRIMITIVE?]
 PRINT [.....PRESS "2"]
 PRINT [ ]
 PRINT [OR PRESS "Q" TO QUIT.]
 MAKE "Q RC
  IF :Q = "1 PG1 STOP
  IF :Q = "2 PG2 STOP
  IF :Q = "Q CLEARTEXT STOP
 CURSOR 10 15
 PRINT [PLEASE USE 1, 2 OR Q]
 PRINT [.......PRESS RETURN AND TRY AGAIN.]
 MAKE "Q RC
 HELP
END
```

```
TO TEACHER
 PRINT [YOUR PICTURE IS SPECIAL! I THINK IT]
 PRINT [ ] PRINT [WOULD MAKE A GOOD SCREEN STORY. IF]
 PRINT [ ] PRINT [YOU ARE INTERESTED IN WRITING SOME]
 PRINT [ ] PRINT [PARAGRAPHS TO GO WITH IT, I'LL SHOW]
 PRINT [ ] PRINT [YOU HOW TO MAKE IT INTO A COMPUTER]
 PRINT [ ] PRINT ["BOOK". SEE ME AT LUNCH IF YOU WANT]
 PRINT [ ] PRINT [TO TALK ABOUT IT.]
END
```

END■

[Shirley Torgerson, Paxson Elementary School, South Higgins and Evan Ave., Missoula, MT 59801.]

[1]Phillips, Wayne R. "How to Manage Effectively with Twenty-five Students and One Computer." *The Computing Teacher*, March 1983, p. 32.

[2]Bloom, Benjamin S. *Stability and Change in Human Characteristics*. NY: John Wiley & Sons, 1964, p. 237.

ICCE Has New Address

ICCE has moved into the Center for Advanced Technology in Education at the University of Oregon. The new Center (CATE) also houses the Career Information System, ERIC Clearinghouse on Education Management, the computer graphics section of the School of Architecture, and computer labs and classes for the College of Education.

ICCE's new mailing address is:

ICCE
University of Oregon
1787 Agate St.
Eugene, OR 97403

The phone number is the same: 503/686-4414.

advertisement

Introducing

MECC LOGO PRODUCTS

EZ LOGO — *Preschool — Grade 3* — *Price: $48.00*

With **EZ Logo**'s simple commands, children who are just beginning to read and compute can successfully use turtle graphics. **EZ Logo** is used in conjunction with Apple Logo software.

APPLE LOGO IN THE CLASSROOM — *Grades 5-7* — *Price: $28.00*

The manual, divided into ten modules, tracks the student from learning the language to using Logo as a problem-solving tool. The manual is designed to accompany the Logo language software published by Apple Computer, Inc.

INTRODUCTION TO LOGO FOR TEACHERS — *Training manual and diskette* — *Price: $45.00*

This guide provides step-by-step session plans and participant materials for a twelve-hour training workshop for teachers in the use of Logo in the classroom.

For Catalog Write or Call: (612) 481-3500

Minnesota Educational Computing Consortium
3490 Lexington Avenue North
St. Paul, Minnesota 55112

mecc

KEPLER

by
Jim McCauley

Five bright pairs of eyes watch me from the other side of the coffee table. These students, who have been coming to my apartment on afternoons and weekends to stretch their thinking skills with Logo, have created many different things: turtle graphics pictures, games, sentence and poetry generators and all manner of strange and partially-functioning systems. They've also learned some of the central concepts in modular design, functional programming and creative thinking. Now, in the last few weeks before I leave for the summer, they want something more, something special.

Carlos, the most serious-minded, asks, "Have you thought about it?"

"It" is an idea for a truly difficult programming project. They had asked me to come up with a real brain-buster for them to solve in the five weeks that we have left together.

"I have, and I still feel the same way. It's your project, so it's got to be your own idea." The looks I get in return make it plain that this is not the answer they were looking for.

"Look," I conciliate, "let's see if we can come up with an idea together. We've brainstormed before and some good projects have come out of it."

Annie gazes out the window on the bright spring afternoon as if to say, "It's too nice a day to be creative."

Gloria is impatient. "Oh, come on, you guys! Somebody here must have had some ideas this week."

All are silent. Then Peter, the smallest of this group of seventh and eighth graders, says quietly, "I don't know if this is an idea for a project, but I found out something interesting this week from a book I've been reading, a book about comets."

All eyes drift over in his direction. Peter is precise and patient, but given to pedantry. After a thankfully brief and reasonably accurate summary of Kepler's discoveries about planetary motion, he comes to the heart of the matter:

"The interesting thing to me is that I always thought that the planets went around the sun in circles, but they don't. They follow orbits that are ellipses."

"Right," says Arthur, a portly boy given by turns to sarcasm and delight. "What's an ellipse?"

My mind is racing ahead. Good grief, they may decide to write a set of procedures to illustrate satellite orbits. While I enjoy reading the popular literature in science, I haven't seen the inside of a physics text since high school, and I was traumatized by trigonometry and calculus. I won't be able to help at all, and my ignorance will be revealed. I decide to save face by intervening, since Peter is having a hard time explaining ellipses.

We go over to the cork board on my kitchen wall, and I take the group through the time-honored way of introducing ellipses. First I place a loop of string around two thumbtacks punched into a piece of paper on the board. Then I run a pencil around the inside of the loop and draw a nice ellipse. The group shows some interest as I move the thumbtacks further apart or closer together and the shape of the ellipse changes. I am about to launch into the usual mind-numbing treatise on foci and eccentricity when Peter pipes up:

"I don't see what this has to do with what the planets and comets do as they go around the sun. I mean, the shape of the orbits is the same as you have there, but that doesn't explain why the orbits look like that or why the planets behave the way they do."

My bluff has been called. The other children turn to Peter in some admiration, then back to me. "Well?" their eyes say.

"Peter is right, of course. There are laws that govern planetary motion, but I don't understand them well enough to explain them to you."

"Maybe you don't have to." Arthur grins. "Are they written down somewhere?"

"Any encyclopedia should have at least a summary under planets, gravity, physics or Newton. Newton deduced the laws from Kepler's observations." At least I can help them find resources.

"Then it's easy. Arthur and I will go down to the library, copy all the pages on orbits and stuff and bring them back here. Somewhere in there will be some formulas that we can translate into Logo, and then we'll have it." Peter is an optimist.

Carlos says, "It sounds too easy to be interesting."

"I'm not sure it will be that easy at all," I say. "Maybe Peter and Arthur should go on to the library while we try to figure out the laws for ourselves."

In a few moments the Laurel and Hardy pair are out the door, and the four of us who remain sit and stare at one another for a few minutes.

My enthusiasm is beginning to rise. All of our really good projects have begun with Peter and Arthur (the two in the group who have been identified as gifted and talented) feeling overconfident and the rest feeling completely lost. This, however, is the first time that I have felt as much at sea as Carlos, Annie and Gloria. Musty memories of half-recalled physics lessons are swimming in my head when practical Gloria comes to our rescue.

"Let's think about what it's going to look like on the screen," she muses. "We've got to have at least two things on the screen, right? A planet and a satellite. Maybe the planet should be in the middle, and the turtle will draw it first. Then the turtle can be the satellite. It can go around the planet, drawing a line to show what the orbit looks like."

Gloria's presumption is brilliant. Of course the turtle should be the satellite! Once the planet is drawn, the turtle will have an object to relate to, a reference point so it can demonstrate the rules we will generate for its behavior.

Carlos says, "You know, this is like that hunting turtle program we wrote, where the turtle chased the fly we made up out of a dot that jumped around at random, except that in this program, the fly is a planet that stays in the same place all the time, and the turtle never catches it."

Idea-bombs are going off in my head. This is always the most difficult time for me: How much guidance should I give them, and how much should I leave for them to discover on their own? Which ideas contribute to a solution, and which will lead us astray? Carlos' comment about the unmoving planet reminds me that even though the sun seems fixed as the center of the solar system, it has its own motion relative to the rest of the galaxy. Is this fact worth mentioning, or is it extraneous? I elect to let it pass.

Gloria begins wondering out loud about what rules make planets behave the way they do, but Carlos argues that we should wait until Peter and Arthur get back from the library.

Annie, who has been quietly doodling on a pad for a long time, disagrees. "I think we should begin making our own guesses about what the rules are. Mr. McCauley, you must know something about them."

"Yes, I guess I do." I have been thinking furiously about the few models of planetary motion that have stuck in my head. They are images, mostly—planetarium projectors, those wonderful clockwork models of planetary motion known as orreries, and curved-space models that use curious tables with funnel-shaped tops into which one pitches large ball bearings which then follow elliptical paths. I decide to go back to first principles.

"If we do as Gloria suggests, the turtle must be given some rules to follow, and those rules must be in the form of a list of instructions. The problem I'm having is that we're going to have to ask the turtle to do two things at once."

"What do you mean?" Annie asks.

"Well, an object that is far away from anything else in space and isn't moving will tend to stay right where it is. If it's already moving, it will tend to keep going in the direction toward and the speed at which it's moving. That's called inertia. If the object is close to something massive like a star or a planet, it will tend to fall toward the massive thing at an ever-increasing rate. That's the law of gravity." I pause to let these concepts sink in. "The problem we have is to figure out what to do when the turtle-satellite is close to a planet but is also moving on its own course. The two forces of gravity and inertia have to be combined, and it might take some advanced math to figure out. I may have to do some homework of my own to write some procedures that you can use as tools for solving this problem."

Because this group prefers to solve all its own problems, this is depressing news. The feeling is compounded a few minutes later when the two researchers, Peter and Arthur, come back from the library with dozens of pages copied from *Encyclopedia Britannica* which they admit neither of them can comprehend.

Still, the group decides to stick with the problem. It's intractable enough to be interesting, and they begin to see how the turtle might behave.

* * *

During the next week I try to think the problem through on my own, but by Wednesday I realize that this is a mistake. First, I have my regular teaching duties to attend to. Next, I must prepare for my university summer courses. Finally, I realize that the students and I have become a team of thinkers. This is not just my challenge—it belongs to all of us.

I do make some progress, though. I plan a series of activities to help them understand the inverse-square law.

This in itself has involved some interesting decision-making on my part as a teacher. I have figured out enough of the problem to realize that at some point we will need a procedure that returns a number to represent the distance that the turtle-satellite will fall under the influence of gravity at various distances from the planet. Having determined this, I must decide whether I should write this procedure for the students or teach them enough about the inverse-square law for them to write it for themselves. I decide to take the latter course.

When Peter, Arthur, Annie, Carlos and Gloria arrive on Sunday, we all pile into my car and go into the country. I have brought along some graph paper and the sound-level meter that I use for tape-recording. When we find a suitably deserted spot, Arthur stays in the car and leans on the horn while the rest of us use the sound-level meter to track the rate at which the horn's volume falls off as we move away from the car. We then record it on the graph paper. We manage to produce a nice steeply-declining curve (see Figure 1), but why the curve looks that way is as clear as mud.

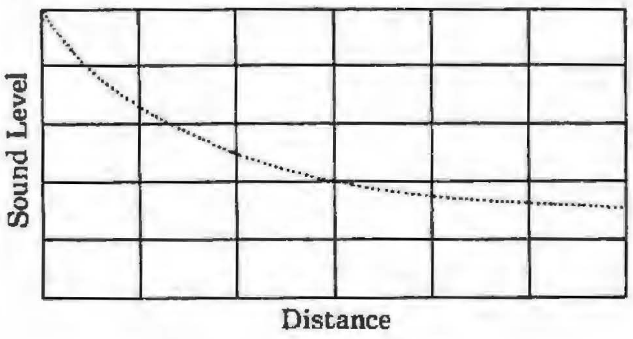


Figure 1.
Horn Experiment

We return to my apartment to try out some other experiments that I hope will clarify the concept. After darkening a room as much as possible with heavy blankets over the windows, we put an unshaded lamp in one corner. Each student looks at the lamp first from the far corner of the room and then from as close as s/he can get before the light becomes too bright to look at. Each of the students fills in a chart like that in Figure 2. The resulting curves correspond well with each other, but why the data follows that curve is still something of a mystery.

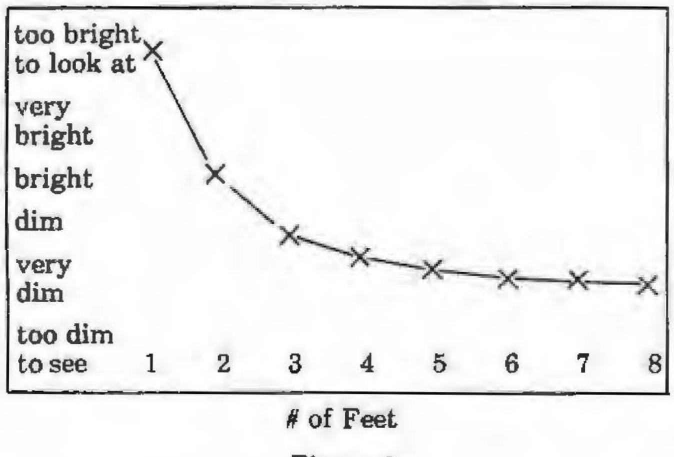


Figure 2.
Light Experiment

Finally, I show the students a balloon which I have inflated to a diameter of four inches, a unit of distance that I have decided to call a "blob." (I have prepared cardboard calipers to help with the measurement.) I have drawn a one-inch square on the balloon (tricky, but not impossible), and inside the square a grid of sixty-four dots. As the group watches, I blow the balloon up further to a diameter of eight inches (two blobs). Now, I place a cardboard template with a one-inch square cut out of its center in the middle of the now-larger square on the balloon and count the number of dots inside the cut-out square (see Figure 3). There are exactly sixteen dots inside the cutout.

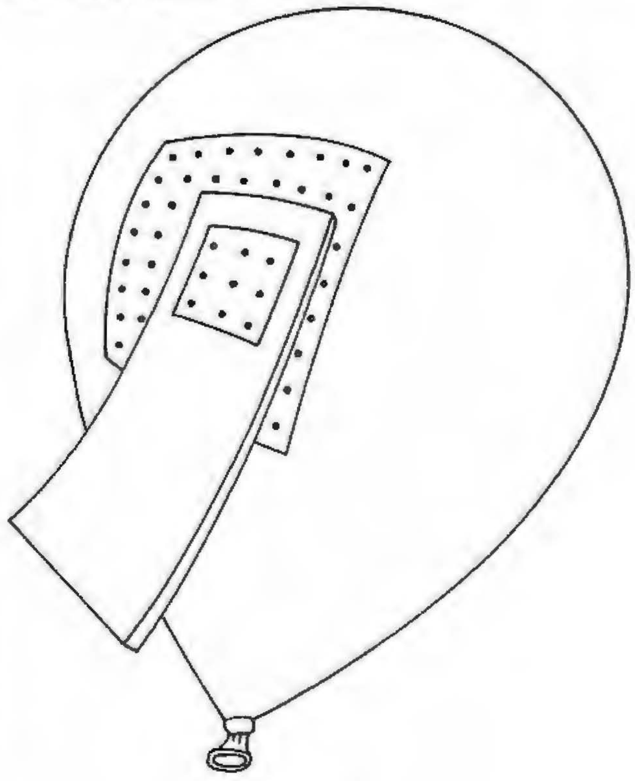


Figure 3.
Template

"Now let's perform a thought experiment. What will happen if we repeat this process, blowing the balloon out to three blobs (with a diameter of twelve inches)? How large will the square on the balloon be?"

Peter, who is a crackerjack at this sort of thing, answers.

"Easy. The first square was one inch on each side when the balloon was four inches in diameter. When you blew it up to eight inches, the square expanded to two inches on each side, but the area of the square went from one square inch to four square inches. If you blow it up to twelve inches, the sides of the large square will expand to three inches and the area will increase to . . . nine square inches."

"Very good! Now, suppose we put the one-inch square cutout over the center of the square on the balloon. How many dots will be inside it?"

Seeing blank looks all around, I go ahead and do it. There are anywhere from six to eight dots inside the cutout, depending on how the cutout is positioned.

"That's interesting," I comment. "Tell me, what's sixty-four divided by nine?"

"7.11, with the one repeating forever," says Annie, who can do this in her head.

"So, we have an average of seven dots inside the cutout, since we get six, seven or eight dots, depending on how we position the cutout. Is 7.11 pretty close to seven?"

All agree that it is. We draw up a chart (see Figure 4), and the Great Light dawns, first for Peter and Annie, and then for the others.

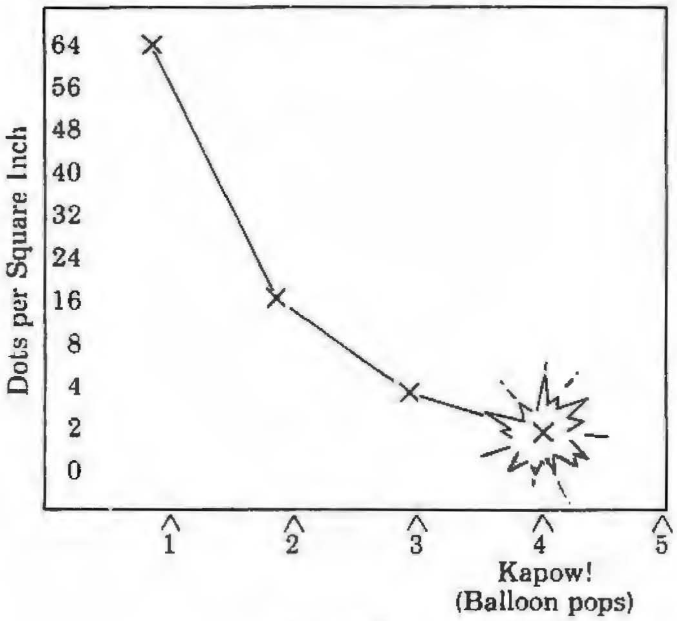


Figure 4.
Balloon Experiment

"Hey!" exclaims Carlos. "The curve looks just like the ones we got for the horn and light experiments."

"Right! Suppose we blow up the balloon to four blobs (sixteen inches) and look at another one-inch square sample of the dotted square. How many dots, on average, will be inside the cutout?"

Gloria shouts, "Four! Because the big square is now four inches on a side, it will have an area of sixteen square inches. And each square inch will have an average of four dots, because four times sixteen is sixty-four—the total number of dots!" She is really beside herself with excitement.

We try to verify this last conclusion, but our balloon pops!

"Okay, calm down," I admonish them gently. "Let's see if we can understand why all the curves look the same. When the car horn first sounded, the sound energy escaped from it in a sort of spherical shell that expanded as time passed—the same way that our balloon expanded as we inflated it. The amount of energy on the surface of the shell is constant as it expands, so if you are very close, the sound is very loud. As you move further away from the horn, your eardrum's share of the shell of sound drops off. But notice that the amount of energy drop is very steep when you are close to the horn and much less steep as you move further away."

"I can see now why that was true for the lamp, too," says Annie. "My eye's share of the light dropped off a lot and moved back a foot when I was close to the light, but it hardly changed at all when I was far away and moved back a foot."

"Exactly right. Each unit of light energy—physicists call these units photons—is like a dot on the surface of an expanding balloon. These dots tend to get scarcer from your eye's point of view as the shell of light expands, but the rate at which they get scarcer decreases. The mathematical expression of this fact is called the inverse-square law, and I think you're ready to understand it now."

I point to our last chart. "If you divide the total number of dots in the dotted square by the increasing area of the square at each "blob," you will get the number of dots per square inch at each distance.

"A scientist would say that the number of dots varies inversely with the square of the distance. Thus, "inverse-square" means that you have to divide the number of total dots (sixty-four) by the distance in "blobs" squared. "Squared" means that you have to multiply the distance by itself."

The students carry out a few calculations on the balloon-experiment data to check what I am saying, and the figures come out right.

"What happens when you get so far away that you only have a fraction of a dot?" asks Peter, who has a gift for posing difficult questions.

"Well, if the dots are photons, you enter the wonderful world of quantum mechanics, which is a bit beyond the scope of our discussion today. Talk to me later, and I'll give you the names of some books to read."

Peter puts on the expression that says he's bitten off more than he can chew but decides to take me up on the physics reading list later.

"Now," I continue, "you're probably wondering what this has to do with our problem." Nods of agreement all around. "Well, gravity follows the inverse-square law. If a satellite is far away from a planet, it will tend to fall toward the planet very slowly as each second passes. But as it gets closer, the planet's gravitational field has a greater and greater effect on it, so the satellite will tend to fall faster and faster."

"Hold it!" says Peter. "I know that things tend to pick up speed as they fall, but you're saying that gravitational pull increases as something gets closer to the earth. That means that the increase in speed itself would increase as the satellite gets closer."

"That's right. Imagine you are out in space on a really high tower thousands of miles above the earth and you drop something. In its first second of fall, it would travel a shorter distance than if you dropped the same object while you were standing on the surface of the Earth, because gravity gets progressively weaker as you move further out into space."

"Wait a minute," says Arthur. "Suppose the planet isn't the Earth, but is a big, heavy one like Jupiter. What happens then?"

"Big planets have correspondingly bigger gravitational fields. Jupiter's is truly stupendous in size. Interestingly, though, very small but very heavy objects, like neutron stars and black holes, have fantastically powerful gravitational fields that are very small in size, while very large objects that are not very dense—interstellar gas clouds, for example—have very large but weak gravitational influence." I have been reading a lot of astronomy to get ready for this lesson.

We begin to talk about what the Logo code for this is going to look like, but the ideas are too new for us to make any headway. We will try writing it next weekend, and we will also try to figure out how to get the turtle-satellite to respond to the two forces of gravity and inertia at the same time.

On his way out the door, Peter comments, "You know, we didn't use the computer even once today."

* * *

Arthur arrives very early for our next meeting, excited and out of breath.

"I've got it! I know how to write the procedure that tells the satellite how far to fall."

"Really? Let's see what it looks like."

Arthur is the only one who has a computer at home. His parents bought it for him the day after he showed them some of his Logo programs we had worked on one afternoon.

"Peter and I figured it out last night. You won't believe it when you see it."

He loads the procedure and prints it on the screen:

```
TO FALL
OUTPUT :MASS / SQ DISTANCE :POS1 :POS2
END
```

He's right—I don't believe it.

"The SQ and DISTANCE procedures are right out of the Logo manual. It was so easy once we figured it out!"

"Great," I say, still mystified. "Now you can explain it to me."

"It's really neat; it sort of does everything at once. The dividing part—SQ DISTANCE :POS1 :POS2—is the inverse-square thing that we learned last week. What makes it inverse is that it's dividing something, just like you told us. The square part of it is SQ, which just multiplies the number that follows by itself. The number that SQ multiplies is DISTANCE, which tells how far the satellite is from the planet."

"Okay," I say, not at all certain. "Now tell me about :MASS."

"Oh, that's the really neat part. :MASS is a number that tells how heavy the planet is. It's just like the number of dots on the balloon in the balloon experiment. Peter worked it out from stuff he found in his book about comets. The bigger and heavier the planet, the more power in its gravity field, right? If you divide :MASS by this inverse-square part, it works just fine. Heavy planets make things fall faster than light planets."

"How do you know that?"

"Remember what you told us about the high tower out in space? Well, Peter and I wrote a procedure that uses FALL to drop the turtle off a tower above planets of different masses and then prints how far the turtle falls."

As Arthur is demonstrating the TOWER procedure to me, the others come in. Peter is a little angry that Arthur is showing off what they invented together until I suggest that he explain it to the rest of the group.

Everyone else is as puzzled as I was, and Carlos asks, "Why doesn't the turtle fall all the way to the planet? Why does it stop?"

Peter and Arthur stare blankly at one another—apparently this hadn't occurred to them, and they don't have an answer. I too have been confused by something during their explanations; I'm haunted by an old physics formula I never really understood: $a=1/2gt^2$. That is, the speed of a falling object is equal to one-half the gravitational pull times the square of the time that it has been falling. What bothers me about the apparently successful set of procedures the two boys have written is that time is not taken into account. Where is the term for elapsed time in FALL?

Then it hits me. There is no time unit because the units of time in the problem we are considering are purely arbitrary. They can simply be defined as having a value of one, which has no effect on the computation in the procedure. Similarly, Carlos' question is answered. Since the TOWER procedure calls on FALL only once, the turtle falls for only one arbitrary time unit. I explain to the group that if the turtle is to fall all the way to the surface of the planet, TOWER will have to call itself recursively over and over again until the turtle hits the surface.

"Splat!" comments Arthur.

"Ugh," responds Annie, who then turns to other matters. "Okay, so this works. How do we get the turtle to go around in an orbit? We can't have the turtle going splat on the planet all the time!"

Peter and Arthur make a few gross comments about falling turtles making craters all over the planet. They are appreciated by no one else, and we settle down to the problem Annie has posed.

Gloria begins thinking out loud. "Hmm. If the turtle gets a push sideways before it starts falling, it might miss the planet and fall around behind it. Oh, I get it—this is where we have to figure out how to get the turtle to do two things at once. It has to fall toward the planet and follow the path of its inertia too, right, Mr. McCauley?"

Gloria is exactly right, but I haven't got the foggiest idea how to write the Logo code to do it. I have a terrible feeling that I am going to have to drag out some old math textbooks and learn instantly all that stuff about vectors I tried so assiduously to avoid when I was in school. It is Gloria who comes up with the first part of the answer.

"Look, this is really just a turtle graphics problem, right? So let's do what we did a couple of months ago when we got stuck on turtle problems."

"Oh, no!" protests Peter. "You don't mean play turtle? That's dumb!"

"Actually, I think it's a good idea," I say. "Maybe we can think of a way to give the turtle-satellite the right instructions if we practice being the turtle and the forces that are pushing it around in space. I've got some ideas about how we can organize it."

I pin up a large sheet of paper on the wall and begin to draw (see Figure 5).

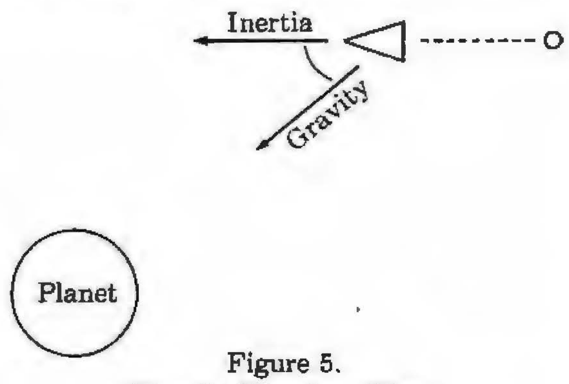


Figure 5.
Turtle-Playing Plan

"Suppose one of us is a planet here in the center. One of us could be the turtle-satellite over here. Then we need two forces, gravity and inertia, to pull the satellite around the planet."

Annie and Gloria quickly volunteer to act as the forces. Arthur asks to be the planet, and Peter is stuck being the satellite. Carlos will help me with some measurements; this sort of thing has always been beneath his dignity.

We adjourn to the parking lot, which is nearly deserted on this beautiful spring Sunday. Arthur, as the planet, positions himself near the center. I give the girls a stack of coasters. They will each take one of poor Peter's arms, dropping one coaster at his heels to mark their starting position. Gloria, as gravity, will take a position directly between Peter and Arthur, since she will be pulling Peter toward the planet. Annie will point off in another direction, representing the satellite's inertial path. Then they will each pull Peter three feet in their respective directions and drop another coaster. Gloria will reset herself on a line between Peter and Arthur, and Annie will reset herself so that she is directly on a line drawn through the two coasters, since her inertial direction will have been influenced by the pull of gravity. Furthermore, the distance that each force will pull the satellite on the next "orbit segment" will be changed. If Gloria is closer to Arthur, she will pull for a slightly longer distance; if further away, she will make a shorter move. Annie will move a distance that is equal to the distance between the last two coasters, since that is in effect the new velocity of the satellite.

We try a few rounds of this, and it actually seems to be working, but it is very slow, since Carlos and I must run around with the tape measure, figuring out the gravitational distance and the new inertial distance between the coasters. Finally Arthur suggests that we use only approximate distances; he will tell Gloria to pull harder when she is closer and less hard as she is moving away, and Annie can keep track of how fast she is moving. Peter has some misgivings about this, which are fully borne out by the resulting chaos. The path they take, laughing and shouting around the lot, does bear a remote resemblance to an ellipse, but Peter's arms are nearly pulled out of their sockets.

"I'm sorry, Mr. McCauley, but there's no way that the turtle can do two things at once if he's going to get pulled apart like this." Peter falls melodramatically onto the lawn.

"Maybe the turtle doesn't have to do two things at once," says Carlos quietly.

"What do you mean?"

"Come inside and I'll show you on Mr. McCauley's drawing."

Carlos takes up the marker and begins sketching as he speaks. (I find myself very excited—Carlos has always been the quiet one.)

"We've been thinking that the turtle has to do two things at once, but it really doesn't. It can do one thing at a time, and it will come out the same.

"Say the turtle starts here. As a satellite, it wants to go in two directions at once, down toward the planet because of gravity, but also out this way because of inertia. Well, why can't it do one thing first and then the other? Look, if it falls toward the planet using FALL, then turns back to the direction it was headed in the first place and travels the length of the inertia, it comes out the same as if it had done them both at once." (See Figure 6.)

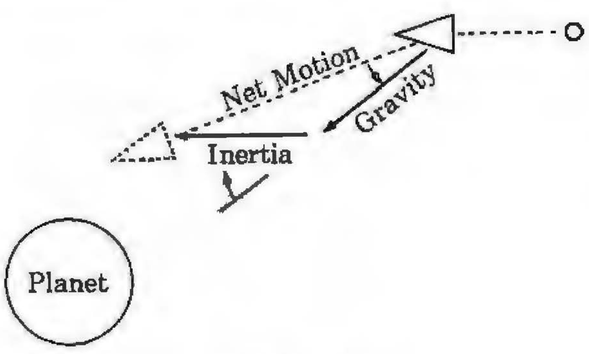


Figure 6.
Carlos' Plan

My first reaction is complete disbelief, followed by stunned surprise and then simple rapture. I had been certain that I would have to teach them something about trigonometric functions, but Carlos has found a way to do the job without them. Real-time vector plotting! I love it.

(Later it will occur to me that we are using a trigonometric function, but in disguise. To turn the turtle in the direction of the planet, we must employ a Logo primitive called TOWARDS, a variation on the arc-tangent.)

Once the others understand what Carlos has in mind, there is a debate about which force should act first. Arthur, Peter and Gloria think that it should be inertia first, gravity second. Annie is convinced that it should be the other way around, and Carlos is sure that it doesn't matter. We decide to go with the majority.

Everyone wants to start writing code immediately, but I insist that we plan what the procedures will look like and which procedures will call which others. After some discussion, we decide the design will fall into three parts: (1) a section that gets information about the planet and the satellite from the user, (2) a section that sets up the planet and satellite on the screen, and (3) a tail-recursive (self-calling) procedure that draws the turtle-satellite's path.

The sequence of actions in this last looping procedure will be as follows:

1. Have the turtle "remember" its INITIAL position.
2. Pick up the pen and hide the turtle.
3. Go forward for the length of the inertia.
4. Set the heading towards the planet.
5. Go forward for the length of FALL.
6. Have the turtle "remember" its FINAL position.
7. Set the turtle's position to INITIAL.
8. Set the turtle's heading towards FINAL.
9. Put the pen down and show the turtle.
10. Set the turtle's position to FINAL.
11. Go to Step 1, resetting the inertia to the distance between INITIAL and FINAL.

Everyone is exhausted, and we decide to call it a day.

* * *

Our last session together is beset by bugs, bugs, interminable bugs. To have so many is usually a sign of bad design and unclear thinking. We've learned a lot from debugging code in the past, but today the bugs are just getting in the way. Now our patience is flagging and tempers are growing short. If the problem is to be solved at all, it must be solved today; tomorrow I leave for my summer university studies.

The easy stuff—the data gathering and the screen setup—is finished. Now we are stuck on a truly serious bug—the turtle is only halfway behaving as it should; it steadfastly refuses to travel in a stable ellipse. Instead, it careens around the planet in what looks like a hyperbola or spirals in very close to the planet and then goes completely crazy, shooting off-screen at insane speeds.

We wonder if our fundamental assumptions might be wrong. Perhaps there is some other property of orbital mechanics that I have neglected to share with the group.

Carlos is at the keyboard, gazing vacantly at a point three feet behind the screen. Gloria is staring out the window. Arthur is trying to cheer up a disconsolate Peter by making strange faces. It isn't working. And I'm starting to think about an early dinner and all the packing I have to do.

Annie is staring at a hole in the chart on the wall, still covered with our sketches from last week. Suddenly she launches herself off the floor like a rocket with a shriek that causes us all to snap our heads about in her direction.

"I knew I was right! I knew it!"

Carlos is the first to react. "Aw, come on. I told you it doesn't matter if it does the inertia first or the gravity-fall first. It all comes out the same."

"No! No, it doesn't. Listen to me! Look up here! Watch!"

She picks up the marker. "Look. Here's what's happening now. The turtle travels its inertial distance, like so (see Figure 7); then it turns toward the planet. That turn isn't the same as the turn it would make if it fell toward the planet first, see? Think about it: the turn the turtle makes after it travels its inertial distance is always sharper than the turn it would make if it set its heading toward the planet before it moves forward. Besides, if the turtle goes inertia-first and that carries it closer to the planet, the distance that it falls will be greater than the distance it would fall if it fell first. But the distance would be less if the inertia carried it further away. My whole point is that they wouldn't be the same."

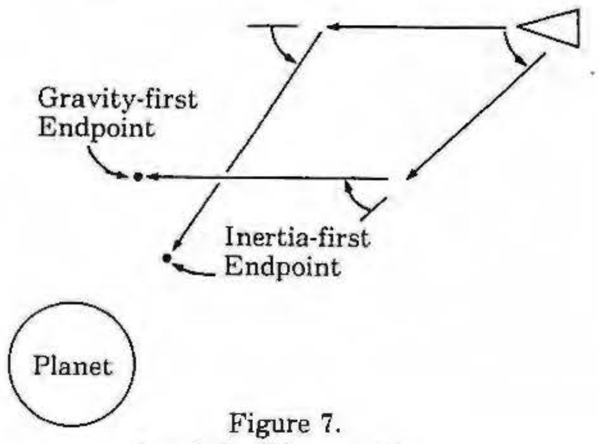


Figure 7.
Annie's Observation

"Well, since both the turn and the falling distance change, maybe the changes offset one another, and it all comes out the same anyway," Arthur responds.

Annie is adamant. "No, I think they add up to something really different. After all, look at the way the turtle's behaving now!"

There are a few moments of silence, then a mixed chorus of sentiments. Peter and Arthur have come around to Carlos' idea that it doesn't make any difference. Gloria just keeps staring out the window.

Carlos is unsure. He turns to me. "What do you think, Mr. McCauley?"

"Well, you were sure that it wouldn't make any difference, so why not try it?"

"But what if it messes it all up?"

"Before you make any changes in the code, save what you have on the disk. If our change doesn't work, you can always load up what you've saved and hit it again."

Carlos looks around and is greeted by shrugs that say, "Why not?"

"Okay, let's try it. What name should I use to save the stuff we have now?"

"How about BLACKHOLE? It's behaving weird enough to be a black hole."

Carlos complies, and after the old code is saved, he hegins working the ORBIT loop that has caused us so much grief. In this new version of the procedure, the turtle will remember both its INITIAL position and its heading, then FALL towards the planet. Then the turtle will turn back to the original heading and travel forward the inertial distance. From there on the code is the same as before.

When he finishes editing, Carlos calls up the top-level procedure (which the students have called KEPLER in honor of that pioneering astronomer). He responds to the parameter questions posed by the program with some reasonable starting data and sits back, expecting another disaster.

It's very quiet in the little room; everyone is tired. Nothing is said for at least twenty seconds. Then, very quietly:

"Oh, wow."

"Will you look at that?"

"Hey, Gloria. . . ."

Gloria turns from the window and now all of us can see the beautiful, beautiful ellipse that the turtle-satellite is drawing.

Peter tears out of the room and out of the apartment. In a few seconds, I can see him turning cartwheels on the lawn. Annie and Gloria hug one another and dance around the room, squealing with delight. Arthur is in a corner, thumping his head on the wall and yelling at the top of his lungs, "We did it! We did it!"

Only Carlos and I are quiet. He is at the computer. Tears of joy stream silently down his face.

As they do down mine. END■

[For a copy of the code, send a self-addressed stamped envelope to Jim McCauley, Santa Clara County Office of Education, 100 Skyport Dr., San Jose, CA 95115.]

advertisement

Terrapin and Krell Logo Users...

Logo is now easier and more exciting than ever!!!

NOW YOUR LOGO CAN...

- **use all 16 APPLE colors!**
- **provide super sound effects!**
- **support multiple turtles!**
- **support full use of 2 disk drives!**
- **provide easy color animation!**
- **offer new microworlds of movement, color, and sound to explore!**
- **help new learners work independently in 1 minute or LESS!**

HELPTURTLE, the newest Logo enhancement disk, lets you do all this and more. Your HELPTURTLE disk comes packed with high motivation microworlds for easy, exciting learning.

Less than one minute of instruction is necessary to get your students actively and independently exploring the TRAINING TURTLE microworld. Children direct the TRAINING TURTLE with SINGLE-KEYSTROKE commands from a clear COMMAND MENU that stays on the screen! Since the TRAINING TURTLE can't get lost behind text and can't generate errors, it's the perfect way to meet the turtle. Age 6 and up.

Imagine four turtles dancing in colored lights! Imagine the creation of a masterpiece with each performance. HELPTURTLE gives you this with the KALIED microworlds of motion and symmetry. Menu driven with HELP screen, makes it easy to select one, two, or four turtles. KALIED can run BY ITSELF, INTERACTIVELY from the keyboard, or through your students' OWN PROCEDURES.

The COLOR microworld is the fastest and easiest way to make large colored shapes using all 16 APPLE colors. Animation is easy in a way never before possible with Logo. HELP screens are ALWAYS available without losing your picture-in-progress. Disk access for permanently saving or retrieving your 16-color pictures is easy, fast, and uses very little disk space.

Use TURTLESONG to add great sound effects or music in any Logo procedure. Simple commands generate your choice of music, glissandos, or arcade sound effects. If your students can write procedures, they can fill the air with TURTLESONG.

DOUBLEDRIVE, alone, will save you hours of tedium in managing your students' or your own Logo files. Logo disk commands are abbreviated to make your work even easier. For example, ERP for ERASEPICT. DOUBLEDRIVE's full use of both drives can end the Logo disk shuffle for you.

HELPTURTLE is fully copyable for backup protection. It comes with a clear instruction manual and demonstration procedures. Available for Terrapin and Krell logo on APPLE II or APPLE //e.

Don't teach another logo lesson without Helpturtle. Order today and receive a free copy of LOGO TEACHING TIPS. You'll be glad you did.

LITTLE RED SCHOOL DISK, Dept. CT-1,
431 Washington Ave., Bethlehem, PA 18017
RUSH my free copy of LOGO TEACHING TIPS with ____HELPTURTLE disks at $39.95 each

___Terrapin ___Krell ___APPLE II ___APPLE //e

NAME (print) ____________

ADDRESS ____________

CITY/STATE/ZIP ____________

PA residents add 6% sales tax

JOIN THE GREAT LOGO ADVENTURE

EXPLORE NEW HORIZONS. JOIN US FOR EXCITING EXPLORATIONS OF LOGO, PILOT, BASIC AND THE MANY WORLDS OF THE PERSONAL COMPUTER.

THROUGH OUR MONTHLY MAGAZINE, YOU'LL SHARE IDEAS, PROGRAMS, FASCINATING CHALLENGES AND CONTESTS, WITH YOUNG PEOPLE THROUGHOUT THE WORLD. LEARN ABOUT ROBOTS AND MICROELECTRONICS FROM MRTLE, OUR MULTIFUNCTIONAL ROBOTIC TASK AND LABOR EXPEDITER, MODEL X409. SHE NEVER LEARNED TO SPEAK "COMPUTERESE."

SHARE YOUR PROGRAMS THROUGH THE YPLA SOFTWARE EXCHANGE, 12 TO 20 PROGRAMS FREE IN RETURN FOR JUST ONE OF YOURS.

DISCOVER LOGO ON AND OFF THE COMPUTER, LOTS OF IDEAS AND ACTIVITIES TO HELP MAKE THE COMPUTER A TRULY CREATIVE LEARNING CENTER FOR ALL THE YOUNG PEOPLES OF THE WORLD.

WE'RE THE YOUNG PEOPLES' LOGO ASSOCIATION. SO, C'MON!

JOIN THE GREAT LOGO ADVENTURE.
IT'S YOUR FUTURE!

YOUNG PEOPLES' LOGO ASSOCIATION

YOUNG PEOPLES' LOGO ASSOCIATION
P.O. BOX 855067 RICHARDSON, TEXAS 75085

( ) YES, I WANT TO JOIN THE LOGO ADVENTURE
( ) ADULT MEMBER ($25) ( ) 18 & UNDER ($9)
$15 FOR INTERNATIONAL AIR MAIL

NAME ______

ADDRESS ______

CITY ______ STATE ______ ZIP ______

Supporting Young Children's Logo Programming

by
Douglas H. Clements

[Editor's Note: The author wishes to acknowledge the enthusiastic cooperation of the staff and students of Evanmire Elementary School in Hudson, Ohio.]

"My first graders might draw with the Logo turtle, but they can do that with a pencil too! And I can't believe they can do *real* programming." This teacher's comment, expressing a common concern, has some truth in it. Without specially-designed assistance, using Logo may be either a frustrating experience or little more than graphic fun for many young children—more expensive, but not more beneficial than crayons. This falls far short of Papert's (1980) vision of children creating new commands to teach the computer, breaking prohlems down into "mind-sized bits" or procedures and thinking about their own thinking. If they do *not* do these things, young children may indeed be better off with just physical objects—crayons or blocks (cf. Barnes & Hill, 1983).

Is it Difficult for Young Children to Learn to Program?

Even with a "friendly" and accessible language like Logo, research indicates that full procedural programming is a complex task for children as old as eight or nine years (Pea, Hawkins & Sheingold, 1983). However, if given *support* in their efforts, even preschool children have been reported to create graphics programs. Perlman (1976) constructed a system of special terminals, a button hox and a slot machine, which allow children to choose from a limited number of pictured commands to program the turtle to draw pictures. Such an elaborate system is not readily available to teachers, however, and children might find it difficult to make the transition from this system to Logo programming.

The Demands of Programming: Providing Support

Is there an inexpensive way to support true procedural programming for kindergarten and primary grade children which allows them to make a gradual transition to the full Logo language? Yes! This article describes a series of programs written in Logo which have been used successfully to support young children's initial programming efforts. These programs give children what good teachers often give—just enough, but not too much, support. They provided the "scaffolding" which allowed children to reach what would have been beyond their grasp. As children's abilities developed, parts of this scaffolding were gradually removed.

What are the demands of programming, even Logo programming, that are obstacles between what children *can* do and what they *want* to do (their goals)? These demands include psychomotor skills such as typing, memory skills such as spelling, and mathematical skills such as estimating distances and angles. Perhaps more difficult are the demands of thinking as programmers think: planning, ordering and remembering a sequence of instructions to be given to the turtle, *without* seeing how it moved in response to each.

"Creative analysis is encouraged in the Logo programming environment. As one girl mumbled to the turtle, "One of us doesn't understand this!"

In a project at Evamire Elementary School in Hudson, Ohio, three levels of support programs were designed to remove some of these obstacles. Level 1 greatly reduced mechanical obstacles such as typing, spelling and estimating distances as well as the obstacles of planning a sequence of instructions. It permitted children to use single keystrokes when moving the turtle, defining procedures ("teaching the turtle new words") and combining these procedures. Level 2 removed only the difficult demands of abstract planning and ordering instructions. It allowed children to write and define new programs in the full Logo language while the turtle simultaneously carried out each command. Level 3 provided only a set of procedures which children could use as "black boxes" in their programming.

Level 1. Nine children without previous computer experience were randomly selected from a middle-class first grade. They were introduced to the computer through the single keystroke program. The group was given a brief explanation of the commands, along with a color-coded chart which pictured the key, its effect and its place on the keyboard. After some time for exploration, children were engaged in problem solving in teams of four. After the team thought of a picture, one child would tell another child who "played turtle" how to walk the picture while the third child typed the commands into the computer. The remaining child checked the whole process for accuracy. The available commands included:

F	(FORWARD 10)	E	(erases last command)
B	(BACK 10)		
1-9	(FORWARD that number)	W	(wipes the screen clean)
R	(RIGHT 30)	S	(saves your picture as a procedure)
L	(LEFT 30)		
T	(Tiny Turn, i.e., RIGHT 3)	P	(asks what picture you want the turtle to draw)
U	(PENUP)		
D	(PENDOWN)	N	(names the picture you have)
C	(asks what pen color you want)		

Unwanted primitive (e.g., R or F) commands are erased directly, without making the child wait for the picture to be redrawn. That is, the turtle can just back up and erase an unwanted line, or turn right to erase an L command. Additionally, pressing "?" (or "/") displays a "help" screen. All other keys (except for CTRL-RESET) were disabled.

A child who drew a square and a triangle with the F and R commands and saved each with the S keystroke might combine them into a house. Typing P, she would be asked,

```
WHAT PICTURE DO YOU WANT TO SHOW?
YOU HAVE  SQUARE  TRI
```

She might type SQUARE and, after the turtle drew this, move to the top of the square, turn right and press P and then TRI. Her house would be complete. She is provided useful information at every step. For example, if she typed in the name of a picture that was not defined, the program would not "crash" but would merely respond YOU DON'T HAVE A PICTURE CALLED SQUARE. Likewise, if she tried to label a new procedure with an old name such as "flower," the program would print YOU ALREADY HAVE A PICTURE CALLED FLOWER.

The Level 1 program, which is an extension of several so-called INSTANT programs, allows children to define procedures and use them to build other procedures. (Teachers wishing to use such programs can find suggestions in "The Logo Center" column of *The Computing Teacher*, October 1983.) Such programs are probably necessary in order to fulfill Papert's vision of children "breaking a problem down" into manageable pieces, using these pieces to build larger wholes and thinking about their own thinking. Abelson (1982) also discusses this concept.

Level 2. After children gained some proficiency at this first level, they were introduced to the second. Here they used the usual Logo commands but in addition utilized several powerful programs provided by the teacher. Typing TEACH, children could define a procedure without exiting the turtle's graphic screen. Like the first method, the turtle carried out the commands one by one as the children entered them, permitting them to use their intuitive visual strengths in creating their programs. If children typed E, the turtle erased the last command entered (as with the previous program, this was usually done immediately, without redrawing). If a command was unknown to the turtle, the program informed the children and did not enter it into the procedure being defined. For instance, if they typed FORWARD50, the program responded: DID YOU FORGET A SPACE? THERE IS NO PROCEDURE NAMED FORWARD50. Similarly, if they typed TEACH again with the TEACH procedure, the program responded YOU ARE ALREADY TEACHING ME SOMETHING! The teacher helped the children read the messages the first few times; children were so motivated to read these messages themselves that they learned to do so quickly.

If a command completely "confused" the turtle, the program would stop; the children then typed RETEACH to begin again right where they left off—minus the offending command. Typing END instructed the computer to define the procedure. Another program, CHANGE, allowed children to alter previously defined procedures. [The appendix contains a listing of the TEACH program along with comments. It is written in MIT Logo but can be easily adapted to other versions, some of which, like Apple Logo, have other error-trapping commands that could be used. The program can be extended to include inputs to the procedures, once children are ready for this new step.]

Along with these programs, children were encouraged to plan their projects in a top-down fashion in the following manner. First, they drew a sketch of what they wanted the turtle to draw. Then they decided how to break down this picture into manageable pieces and used tracing paper to trace each piece onto a *separate* piece of paper, labeling each piece. Two girls traced and named these parts of their GIRAFFE picture: GRASS, LEG, BODY, HEAD, TAIL, SPOTS and BIRD. Using TEACH, they wrote a procedure for each piece and finally wrote a superprocedure which combined them. Figure 1 illustrates their program, GIRAFFE.

To help plan each separate procedure, children were encouraged to draw over the pieces with a pencil which had a paper turtle mounted on it. They would move and turn the pencil while describing their actions to a friend, who would write them down. The teacher encouraged children to bring their thinking to an explicit level of awareness by asking questions such as, "What does the turtle have to know to draw this piece?" or "What did you tell the turtle to do? What *did* it do? What did you *want* it to do? How could you change . . . ?" and so forth.

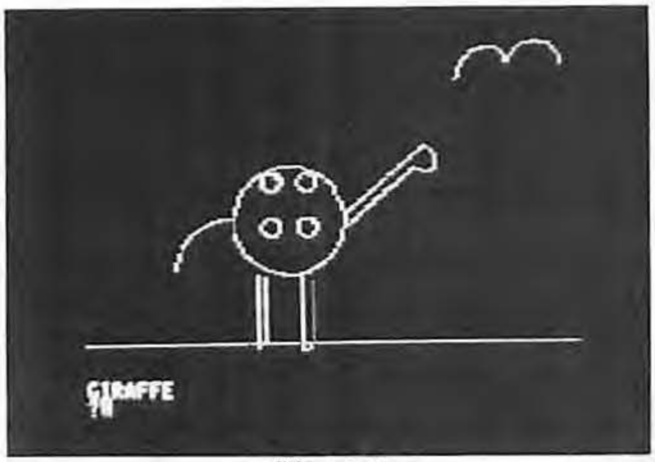


Figure 1.

Level 3. At this level, children used the Logo editor (or TEACH, depending on their preference) along with special procedures supplied to them. These included CIR.RT (draws a circle curving to the right; the child must specify a radius), CIR.LT, CUR.LT (a quarter circle to the left), CUR.RT, MOVE.UP (moves the turtle up without drawing; the child must specify a distance), MOVE.DOWN, MOVE.RT, MOVE.LT, POLY and POLYFILL (draws a polygon and fills it with color by drawing ever-smaller concentric polygons; the child must specify the number of sides and the length of each side). Although all of the children already figured out how to draw curves at each of the two previous levels, these programs allowed them to use procedures with a variable input as a "black box" to draw curves and circles or to move quickly around the screen. It was found that children did not have to be prompted to enter an input. They simply typed MOVE.UP 50, and the turtle would move 50 steps straight up, without drawing a line and without changing its orientation. These procedures, incidentally, were also available to the children at the previous level on the basis of need.

How Well Did Children Work With These Programs?

The results of three months (two sessions a week) of work with these programs by first grade children were encouraging. The most obvious outcomes were the programs written. (See Figures 2 and 3, along with Figure 1.)

Just as interesting are the results of several assessments. The children's performance on a measure

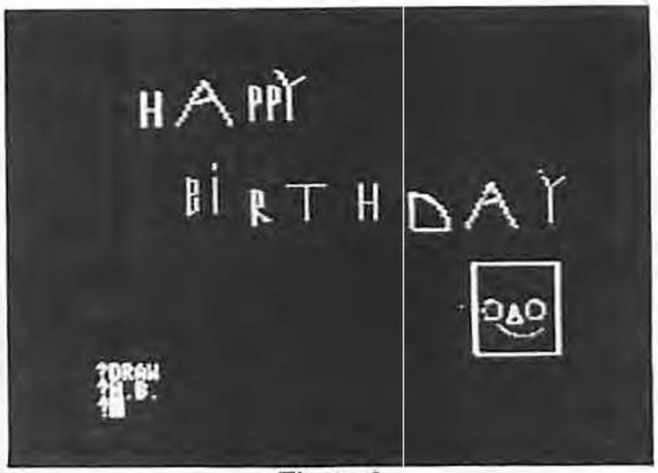


Figure 2.

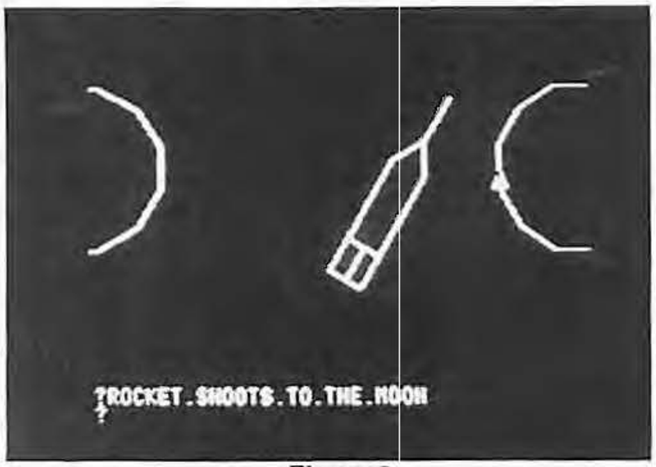


Figure 3.

of learning style changed significantly—they became more reflective and made fewer errors compared to non-programming children. After a procedure failed to work satisfactorily, one trio of Logo programmers expressed the need to take time and think: "We've *got* to get organized!" They scored higher on a test of creative thinking than (a) a control group and (b) than they had scored on a pretest given before the Logo experience. They performed dramatically better than the control group on a task in which they were required to discover that directions were incomplete. In this task, children are told how to play a game, but an essential step is omitted; the children must realize that they do not understand and inform the teacher.

Such creative analysis is encouraged in a Logo programming environment. As one girl mumbled to the turtle, "*One* of us doesn't understand this!" Their excitement and involvement in the programming never abated. The obvious social (and emotional!) involvement may have contributed to these positive results (see Figure 4).

We learned several lessons as *teachers. Next year* we will allow children more time to "overlearn" the methods at each level before moving to the next higher level. We learned that a very sensitive

Figure 4.

balance must be maintained among brief teacher-directed lessons (on specifics of the language), group problem-solving discussions, and student-planned and executed activities. At first our directed lessons were too long—children tended to copy exactly what we had demonstrated. We then virtually eliminated these lessons but found that children were not learning new commands. Finally, we settled on a pattern of (a) a directed lesson on some new concept for 3-5 minutes, (b) group problem solving based on the new concept for 5-10 minutes, and (c) self-directed work for 30 minutes. We also learned that, given the proper support, children's capacity for creative and logical thinking consistently surpassed our expectations.

Appendix

TO TEACH
  ; BY DOUGLAS CLEMENTS
  ; KENT STATE UNIVERSITY	

DRAW	Clears the screen.
PRINT1 [NAME OF PROCEDURE >] BELL	Asks for the name of the procedure.
MAKE "PROCEDURE FIRST REQUEST	This line causes the computer to wait for the user to type in the name.
MAKE "HISTORY FPUT :PROCEDURE [ ]	Starts a list of all the commands, naming it "HISTORY."
TEACH2 END	
TO TEACH2 PRINT1 [OK! >] BELL	Informs the child all is well with a message and a tone; PRINT1 is used because we do not want a carriage return.
MAKE "LINE REQUEST	Waits for the next line of the program.

IF :LINE = [END] DEFINE :PROCEDURE
    FPUT [ ] BUTFIRST :HISTORY (PRINT
    :PROCEDURE "DEFINED) STOP
	If the child types "END", the program defines the new procedure, using the list of commands.

IF :LINE = [E] RETEACH STOP	If the child types "E", this causes the computer to run a procedure to erase the last command.
IF BADLINE? :LINE TEACH2 STOP	Causes the computer to run a procedure which checks if the command typed in was a Logo command.
UPDATE	Runs a procedure which updates the list of commands.
RUN :LINE	Runs the command so that the child can see the effect.
TEACH2 END	Generates another copy of TEACH2.

```
TO RETEACH
  IF BUTFIRST :HISTORY = [ ] PRINT [THERE'S
    NOTHING TO TAKE AWAY] TEACH2
    STOP
```

If the list of commands is empty, the computer tells the child there is nothing more to erase.

```
  MAKE "PROCEDURE FIRST :HISTORY
  MAKE "ERRORLINE LAST :HISTORY
  MAKE "HISTORY BUTLAST :HISTORY
  MAKE "FIRSTWORD FIRST :ERRORLINE
```

These lines give names to
—the procedure
—the last, erroneous line
—the new list of commands (omitting the last line)
—and the first word in the last line.

```
  IF NOT BUTFIRST :ERRORLINE = [ ] MAKE
    "SECOND FIRST BUTFIRST :ERRORLINE
    ELSE MAKE "SECOND [ ]
```

If the last line (ERRORLINE) is more than one word, then the name "SECOND" is assigned to the second word; otherwise, SECOND is an "empty list."

```
  IF NOT :SECOND = [ ] MAKE "THIRD
    BUTFIRST BUTFIRST :ERRORLINE ELSE MAKE
    "THIRD [ ]
```

Similarly for the third word.

```
  PRINT1 :ERRORLINE
  PRINT [?  I ERASED THAT.]
```

The child is shown the command that has been ignored or erased.

```
  (PRINT [NAME OF PROCEDURE IS > ]
    :PROCEDURE )

  IF (ALLOF MEMBER? :FIRSTWORD :COMMAND
    NUMBER? :SECOND :THIRD = [ ] )
    UNDO ELSE DRAW RUN.AND.PRINT BUTFIRST
    :HISTORY
```

This lengthy line asks: Is the first word a member of the list of commands, *and* is the second a number, *and* is the third word non-existent? If *all* of these are true, use the UNDO erasing procedure; *otherwise*, use the run and print method, which redraws the picture, omitting the last command.

```
TEACH2
END
```

```
TO BADLINE? :LINE
```

This procedure checks whether the line is one that cannot be run for some reason.

```
  IF :LINE = [ ] OUTPUT "TRUE
  IF FIRST :LINE = :PROCEDURE UPDATE
    OUTPUT "TRUE
```

If the command is the same as the procedure being developed, add the command to the list, but do not try to run it yet, because the procedure is not yet defined.

```
  IF MEMBER? FIRST :LINE [TEACH RETEACH
    CHANGE] PRINT [YOU ARE ALREADY
    TEACHING ME SOMETHING!] OUTPUT "TRUE
```

Children sometimes forget they are TEACHING the computer something already.

```
  IF NUMBER? FIRST :LINE PRINT1 [YOU DON'T
    SAY WHAT TO DO WITH ' '] PRINT :LINE
    OUTPUT "TRUE
```

Children may forget the command and type in 30 instead of FD 30.

```
  IF ALLOF MEMBER? FIRST FIRST :LINE
    [F B R L] NUMBER? LAST FIRST :LINE
    PRINT [DID YOU FORGET A SPACE?]
```

Or they may type FD30, so this checks if the command starts with F, B, R or L *and* ends with a number.

```
  IF ALLOF MEMBER? FIRST :LINE :COMMAND
    BUTFIRST :LINE=[ ] PRINT [ DID YOU
    FORGET A NUMBER? ] OUTPUT "TRUE
```

Or they may forget the number, e.g., FD.

```
  IF TEXT FIRST :LINE = [ ] PRINT1 [THERE IS
    NO PROCEDURE NAMED ' '] PRINT :LINE
    OUTPUT "TRUE
```

Or they may type any other word that is not a procedure, such as BAC or DKSK.

```
  OUTPUT "FALSE
```

Otherwise, it is probable the line was *not* a "badline."

```
END
```

```
TO UPDATE
  MAKE "HISTORY LPUT :LINE :HISTORY
END
```

Adds the new line to the list of commands called HISTORY.

```
UNDO
  MAKE "FIRSTLETTER FIRST :FIRSTWORD
  MAKE "ORIGINAL.PENCOLOR LAST
    TURTLESTATE
  PENCOLOR LAST BUTLAST TURTLESTATE
  IF :FIRSTLETTER = "F BACK :SECOND
  IF :FIRSTLETTER = "B FORWARD :SECOND
  IF :FIRSTLETTER = "R LEFT :SECOND
  IF :FIRSTLETTER = "L RIGHT :SECOND
  PENCOLOR :ORIGINAL.PENCOLOR

  PRINT BUTFIRST :HISTORY

END
```

This procedure allows the turtle to erase without redrawing the whole picture. It "reverses" the last command, drawing in the same color as the background color.

The original pencolor is reset.

The child is shown what commands are left in the procedure.

```
TO RUN.AND.PRINT :LIST
  IF :LIST = [ ] PRINT [ ] STOP
  (PRINT1 FIRST :LIST ['    '] )

  IF NOT :PROCEDURE = FIRST FIRST :LIST
    RUN FIRST :LIST
  RUN.AND.PRINT. BUTFIRST :LIST
END
```

When the picture must be redrawn, this procedure does the job, as it also prints each command.

```
TO MEMBER? :WORD :LIST
  IF :LIST = [ ] OUTPUT "FALSE
  IF :WORD = ( (FIRST :LIST ) OUTPUT "TRUE
  OUTPUT MEMBER? :WORD BUTFIRST :LIST
END
```

Checks if the word is a member of the list.

```
TO CHANGE
  DRAW

  PRINT1 [CHANGE WHAT?>]
  MAKE "PROCEDURE FIRST REQUEST
  MAKE "HISTORY FPUT :PROCEDURE BUTFIRST
    TEXT :PROCEDURE
  (PRINT [NAME OF PROCEDURE >]
    :PROCEDURE )
  RUN.AND.PRINT BUTFIRST :HISTORY
  TEACH2
END
```

This is typed in if a child wants to edit an already defined procedure. It asks which procedure to change, then "allows" the child to edit it.

```
TO BELL
  PRINT1 CHAR 7
END

"COMMAND IS [RT LT FD BK RIGHT LEFT
    FORWARD BACK]
```

END■

[Ed. note: The author will provide a more detailed description of the support programs to interested readers. Dr. Douglas H. Clements, Assistant Professor of Childhood Education, Kent State University, 300 White Hall, Kent, OH 44242.]

References

Abelson, Harold. *Logo for the Apple II.* Peterborough, NH: BYTE/McGraw-Hill, 1982.

Barnes, B.J., and Hill, Shirley. "Should Young Children Work with Microcomputers—Logo before Lego?" *The Computing Teacher,* Vol. 10 #9 (May 1983), pp. 11-14.

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

Pea, Roy; Hawkins, Jan; and Sheingold, Karen. Developmental Studies on Learning Logo Computer Programming. *Abstracts from the Biennial Meeting of the Society for Research in Child Development,* 4 (April 1983), p. 207. (Summary)

Perlman, Radia. *Using Computer Technology to Provide a Creative Learning Environment for Preschool Children.* Logo Memo #24. Cambridge, MA: Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1976.

Where Is Logo?

by
Tom Lough

Since its recent release to the public, Logo has grown in popularity. This growth is easily discernible. Here are some obvious indicators.

Versions of Logo have been produced initially for the microcomputers most widely used in education. Logo books became available from a variety of publishers. Nearly every educational computing magazine has added a regular Logo column or section. Newsletters dedicated to Logo have appeared. Computer camps have added Logo classes to their programs. Logo presentations have found their way into the schedules of nearly every educational computing conference.

These parameters of Logo's growth are apparent. However, they provide at best only an indirect indication of Logo's growth in the classrooms. They also do not give an indication of where Logo is being used as an educational tool.

How widespread is the use of Logo in our schools? What is the geographical distribution of these schools?

I feel these are important questions. The answers would give a picture of current Logo activity in the nation's classrooms and provide information critical to future teacher training and school budget planning.

As editor of *The National Logo Exchange* (NLX), a newsletter for Logo teachers, I realized that I had in my possession data which could suggest an initial response to these questions. In this article, I will present two sets of data which indicate the present geographic distribution of Logo classroom activity and which provide a basis for an estimate of school Logo use nationwide.

Data Description

Sets of data were generated from responses to two offerings made to Logo teachers by the NLX. These responses were sorted by state to obtain an indication of the relative Logo activity in various parts of the country. Each of the two sets of data contain approximately 1000 entries. Although these data are not direct observations of use, they are strong indicators of involvement of school personnel in the use of Logo.

Data generated during the period studied represent the Logo versions available at that time. These included Apple (LCSI) Logo, MIT Logo (Terrapin, Krell) for the Apple, TI Logo, TRS-80 Color Logo, and CyberLogo.

Here are brief descriptions of the two sets of data and how they were generated.

Group I: NLX ABCs

As a result of a challenge by Robert Tinker of Technical Education Research Centers (TERC) (see *Hands On!,* Spring 1982, page 5), the NLX developed a set of Logo procedures to draw all letters of the alphabet in any arbitrary size. These procedures not only fit into the memory of the Apple computer, but allowed enough extra workspace for the development of utility procedures to manipulate the letters.

This set of procedures was made available to all Logo teachers through a series of announcements in the *NLX*, *The Computing Teacher*, and other leading monthly educational computing periodicals. The set of procedures was offered free of charge. Additionally, since the NLX ABCs were designed only for the floating point Logos of the Apple computer (LCSI and MIT), users of the other (integer based) versions available were not as motivated to respond to the offer. (A few did, however, just to investigate a starting point for their own work.)

Respondents to this offer were primarily elementary and middle school teachers. The responses were received during the period from December 6, 1982 to September 2, 1983.

Group II: NLX Subscribers

The overwhelming majority of NLX subscriptions were obtained as a result of news notices and references in Logo articles which appeared in most educational computing periodicals throughout late 1982 and early 1983. Additional responses were generated by personal recommendations from subscribers and from the NLX ABC offerings. No paid advertisements were placed; hence, no responses are based on them. In all instances, subscription payments of at least $25 were made either by individuals or by sponsoring institutions. Thus, a degree of financial and professional commitment was indicated.

The NLX subscriber distribution represents a mixture of Logo teachers in elementary, middle and high schools, with an additional similar group of education professors at the university or college level. The subscriber group covers the period from July 28, 1982 to September 2, 1983.

Overlap

Although the motivation for each offering was different, some overlap between the two populations was to be expected. Announcements of both the NLX subscriptions and the ABCs were made in the same general set of publications. Additionally, a number of ABC recipients became NLX subscribers. The observed overlap was just under 20%.

Data Analysis

As with all statistical data, there are several ways to interpret what is presented here. The first analysis was a computation of the relative number of responses by state. Table 1 gives an alphabetical summary of all the data. Columns 2 and 4 provide a state ranking of the ABC and NLX subscription activity as percentages. It should come as no surprise that the large populations of New York and California account for a significant portion of the responses. Note that the widespread Logo activity of Massachusetts is indicated clearly.

In spite of the small overlap in the two populations, the percentages for each state are the same magnitude in each distribution. The correlation coefficient between them was calculated to be 0.96, giving a reasonably consistent indication of the Logo activity represented.

Because the population was a confounding variable, it seems of more interest to examine the responses relative to per capita distribution. A radically different picture is presented by Columns 1 and 3, in which the number of responses of each state is divided by the 1980 census population. Once more, there appears to be a general correspondence between the two groups. The correlation coefficient was calculated to be 0.86.

Table 1.
Alphabetical Summary of Logo Activity Reported

State	ABC per Cap	ABC %	NLX per Cap	NLX %
AK	10.00 (7)	0.36	35.00 (1)	2.05
AL	2.31	0.81	0.26	0.15
AR	0.87	0.18	0.44	0.15
AZ	3.68	0.90	1.47	0.59
CA	4.86	10.41 (2)	2.62	9.09 (2)
CO	9.34 (9)	2.44	7.61 (6)	3.23 (9)
CT	5.47	1.54	4.82	2.20
DC	7.81	0.45	4.69	0.44
DE	0.00	0.00	3.39	0.29
FL	1.74	1.54	1.64	2.35
GA	1.65	0.81	1.10	0.88
HI	12.37 (4)	1.09	14.43 (2)	2.05
IA	5.15	1.36	3.78	1.61
ID	7.45	0.63	4.26	0.59
IL	5.16	5.34 (5)	3.15	5.28 (4)
IN	3.83	1.90	2.37	1.91
KS	6.36	1.36	4.66	1.61
KY	2.46	0.81	0.82	0.44
LA	0.95	0.36	0.48	0.29
MA	11.15 (5)	5.79 (3)	8.36 (4)	7.04 (3)
MD	4.74	1.81	3.32	2.05
ME	5.31	0.54	4.42	0.73
MI	2.81	2.35	2.59	3.52 (7)
MN	6.86	2.53	4.41	2.64
MO	3.86	1.72	2.44	1.76
MS	0.00	0.00	0.00	0.00
MT	16.46 (1)	1.18	3.80	0.44
NC	3.40	1.81	1.53	1.32
ND	6.15	0.36	0.00	0.00
NE	9.55 (8)	1.36	3.82	0.88
NH	4.35	0.36	1.09	0.15
NJ	8.28	5.52 (4)	2.85	3.08
NM	4.62	0.54	2.31	0.44
NV	3.75	0.27	1.25	0.15
NY	7.97	12.67 (1)	4.90 (10)	12.61 (1)
OH	4.54	4.43 (6)	1.94	3.08
OK	0.99	0.27	0.66	0.29
OR	13.69 (2)	3.26 (9)	8.75 (5)	3.37 (8)
PA	3.37	3.62 (8)	1.69	2.93
RI	7.38	0.63	5.27 (9)	0.73
SC	2.88	0.81	0.64	0.29
SD	7.25	0.45	2.90	0.29
TN	1.31	0.54	1.09	0.73
TX	2.39	3.08	2.11	4.40 (6)
UT	6.16	0.81	2.74	0.59
VA	4.49	2.17	5.99 (7)	4.69 (5)
VT	7.84	0.36	9.80 (3)	0.73
WA	8.47 (10)	3.17 (10)	5.33 (8)	3.23 (10)
WI	10.19 (6)	4.34 (7)	2.76	1.91
WV	2.05	0.36	1.54	0.44
WY	12.77 (3)	0.54	4.26	0.29

(Ranking of the top ten in parentheses)

Several states with relatively small populations (Hawaii—965,000 and Alaska—402,000, for example) show a high Logo activity on a per capita basis.

States with larger populations no longer appear at the top of the list. New York is barely in the top ten, and California now appears in the middle of the per capita distribution.

The depth of the Logo activity in Massachusetts is evidenced by its staying power among the top 5 states on a per capita basis. Likewise, Oregon remains at or near the top of all four lists.

Concluding Comments

It is conservatively estimated that the two samples presented here (on the order of 2000 total responses) represent between 5% and 10% of the Logo-using educators of the United States. The proportions of this population likely follow the geographical distribution of the reported samples.

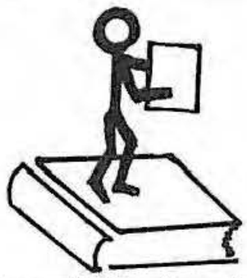
As Logo use in the schools continues to spread, the total picture presented here will change. Additionally, the effects of Logo encouragement and endorsement by state departments of education will be felt.

Finally, Logo versions for several other popular microcomputers have appeared or have been announced for release. Among these are Commodore Logo, Atari Logo, and at least four Logo versions for the IBM personal computer (Waterloo Logo, IBM (LCSI) Logo, PC Logo and DR Logo). As these and other Logo versions find their way into the nation's classrooms, we can expect Logo activity to increase rapidly.

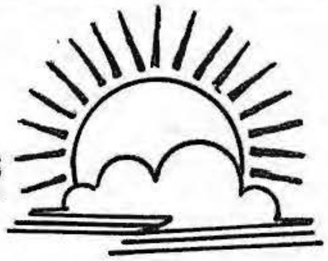
In the meantime, it is hoped that the data and tables reported in this article will have given a rough indication of the educational Logo activity in the United States, its geographical distribution, and a basis for an estimate of Logo use in the nation's schools. END■

[Tom Lough, Editor, The National Logo Exchange, *P.O. Box 5341, Charlottesville, VA 22905.]*

advertisement



OAKLAND SCHOOLS



LEARNING ASSESSMENT CLINIC'S COMMUNICATION ENHANCEMENT CENTER

Proudly announces the availability of two educational products for use with children and adults with communication and motor disabilitie

PICTURE DICTIONARY

Developed by: Ina J. Kirstein
Illustrated by: Carol Bernstein

contains...

- over 500 simple line drawings divided into 20 categories
- alphabetical and categorical indexes
- line drawings presented in three sizes -- 2 inch, 1 inch and ½ inch
- sample communication boards and grids
- compatible with commercially available microcomputer interfaces (Zygo 16, 100, Express 1 & 3)

** Price $35.00

SUNNY DAYS MICROCOMPUTER READING GAMES DISK

Developed by: Nathaniel A. Peters
Designed by: Christine Cafmeyer and James Cafmeyer

- for use on the Apple II Microcomputer
- a game oriented approach to teaching word recognition and comprehension skills
- music, colorful graphics, sound effects, individualized speed of display
- uses keyboard, spacebar, game control paddles or commercially available adapted switches for input
- allows total independence for the student
- uses a pre-entered set of 100 core vocabulary words or is easily modified by the teacher to individualized vocabulary

** Price $30.00

For further information please contact Oakland Schools, Communication Enhancement Center, 2100 Pontiac Lake Road, Pontiac, MI 48054.
** Price based on limited supply and subject to change.

A Cure for Recursion

by
Tom Lough

"At first, Logo seemed almost too simple to bother with. But, the more I worked with it, the more substance I found to it."

Comments such as this are heard often from teachers as they acquire experience in learning and teaching with Logo. The same type of comments are heard about specific features of Logo also.

Take recursion, for example. It has a deceptively shallow and simple appearance. However, there is an incredible capability hidden within which has confounded and frustrated many. This article attempts to help teachers understand the various aspects of recursion in Logo, and it provides some specific suggestions on how to help students learn these aspects effectively.

A companion article by Tim Riordon in this issue offers additional ideas and many more projects.

Prerecursion Thoughts

Often, a second look at Logo procedure definition helps the later understanding of the recursion process. In particular, the mechanism which operates on embedded procedures contains a concept critical to recursion. Let's look at an example.

```
TO BOX
REPEAT 4 [ FD 20 RT 90 ]
END
```

It is fairly easy to see that a square of size 20 is drawn when BOX is called.

```
TO FIGURE
FD 30 BOX FD 40 RT 90 BOX
END
```

Now BOX is embedded in the procedure FIGURE. What happens when FIGURE is called?

First is FD 30. Then FIGURE calls BOX, which draws its square of size 20. But, what is FIGURE doing while BOX is drawing? Of course! It is waiting for BOX to finish. Logo can do only one thing at a time (like many of us!). After BOX has finished the square, FIGURE continues with its next instructions, FD 40 RT 90. BOX is called once more, causing FIGURE to wait. When this second square is drawn, FIGURE resumes. At this point, however, there are no more instructions in the procedure FIGURE, so it ENDs.

This example may also seem simple. But the idea of one procedure pausing while another performs is central to a full understanding of the recursive process.

This general rule may help explain the process:

When one procedure calls another, the calling procedure pauses until the called procedure ends. Then, and only then, the calling procedure resumes.

In the context of the above example, FIGURE called BOX and paused until BOX finished. Then FIGURE resumed.

As you teach procedures, it is important to encourage and allow your students to explore fully the effects of embedded procedures. The computer experience is important, of course; however, activities away from the computer—in which each student is assigned either a Logo command or a procedure name to act out—help to establish this process and precedence in accordance with the general rule above.

"As you teach procedures, it is important to encourage and allow your students to explore fully the effects of embedded procedures. The computer experience is important, of course; however, activities away from the computer in which each student is assigned either a Logo command or a procedure name to act out help to establish this process and precedence in accordance with the general rule above."

With this idea in mind, let's begin our adventure into recursion.

Testing the Waters

Look carefully at this procedure.

```
TO SQUARE
FD 20 RT 90 SQUARE
END
```

What happens when SQUARE is called? Let me suggest an answer along the same lines as the example with FIGURE. After all, it really is the same process, isn't it? SQUARE executes FD 20 RT 90, and then calls another procedure, in a manner similar to that of FIGURE.

However, there is one difference. The name of the procedure which is called has the same name as the calling procedure! Notice that I did not suggest that the procedure called itself. Instead, the procedure called another procedure with the same name and instructions. This point is small, but important to a full understanding of recursion.

Who among us has not tried dialing the number of the phone you were using just to see what would happen? In most (?) cases, you get a busy signal from trying to call yourself. In Logo, that does not happen! Instead, when the calling procedure calls a procedure with the same name, it sees the same set of instructions as defined in the procedure, but as a separate procedure from itself (like telephoning yourself and having yourself answer from another universe)!

Let's continue with our examination of the SQUARE procedure. From the general rule we know that when a procedure calls another, it pauses until the called procedure finishes. Logo keeps track of each separate SQUARE procedure called. Let's do the same by denoting the first SQUARE procedure as firstSQUARE, the second as secondSQUARE, and so on. Each SQUARE has the same instructions as the other.

FirstSQUARE executes FD 20 RT 90, calls secondSQUARE, and waits for it to finish. SecondSQUARE performs FD 20 RT 90, calls thirdSQUARE, and waits for it to finish. ThirdSQUARE . . . you get the idea. Can you figure out what the turtle is drawing? It is a square of size 20 in which the turtle goes over the lines forever (until it is interrupted or it encounters an error condition). This is called tail recursion.

The term tail recursion refers to the position of the crucial line in the Logo procedure, just before the END. This line contains the name of the procedure in which it appears, and causes another set of the procedure's instructions to be called after a complete set of them has been executed.

Here is a way to experience tail recursion in a physical way. Carefully remove a speck of the reflecting material from the middle of the back of a small inexpensive mirror. To prevent flaking, place some cellophane tape over the spot. Put your eye up to the hole and look through it with the mirror facing another mirror. The infinity of images you see are produced by self reflections, akin to the phenomena of tail recursion. Each reflection is distinctly discernable and is a copy of the original mirror.

Testing the Brakes

It is one thing to get a tail recursion started in Logo. But how to stop it? There are a variety of ways. Perhaps the most intrusive and direct way is to interrupt. CONTROL G will interrupt in several of the most popular Logo versions. This stops the turtle in its tracks!

If there were some way for fourthSQUARE in our example to stop on its own, then thirdSQUARE, which was waiting in line, could finish its commands. (In this case, of course, the next command would be END.) Once thirdSQUARE finished, secondSQUARE would end also, followed by firstSQUARE. This would produce a square and stop the turtle after one repetition.

The Logo TEST command can also be used within a recursive procedure to stop the action. For example, here is a way to draw a square and use TEST to stop when the fourth side is completed.

```
TO DRAW.SQUARE
SETHEADING 0
SQUARE
END

TO SQUARE
FD 30 RT 90
TEST HEADING = 0
IFTRUE STOP
SQUARE
END
```

DRAW.SQUARE sets the heading of the turtle to 0 (to face the top of the screen), calls SQUARE, and waits for it to finish. Since SQUARE is tail recursive, let's trace the development in detail. FirstSQUARE executes RT 90 FD 20, as in the earlier example. Then it tests to see if the heading is 0. In this case, it is 90, so the result of the test is FALSE. The IFTRUE instruction is not carried out. FirstSQUARE calls secondSQUARE, and waits for it to finish. SecondSQUARE draws, tests the heading (180), calls thirdSQUARE and waits. ThirdSQUARE also draws, tests the heading (270), calls fourthSQUARE and waits. FourthSQUARE draws and tests the heading. It is now 0 because of the RT 90 in fourthSQUARE. The result of the TEST is TRUE, so the IFTRUE instruction (STOP) is now carried out. FourthSQUARE STOPs. Since thirdSQUARE, secondSQUARE, firstSQUARE and DRAW.SQUARE are still waiting, they each can resume in turn.

But the next instruction in each case is END. So, they each END in turn, and Logo returns control to the user. Whew!

This was a lot of detail. But if you can follow the last paragraph you are well prepared for the more sophisticated aspects of recursion.

Here is a tail recursive activity for you to trace out. Can you follow the sequence of execution?

```
TO DRAW.CIRCLE
MAKE "AZIMUTH HEADING
CIRCLE
END
```

```
TO CIRCLE
FD 1 RT 1
TEST HEADING = :AZIMUTH
IFTRUE STOP
CIRCLE
END
```

More Tail Recursion Activities

Incremented variables make tail recursion a lot of fun For example, try the following procedure:

```
TO FIGURE :X
FD :X RT 60
FIGURE :X + 10
END
```

This procedure has the same general organization as previous ones. Let's use the same technique to begin tracing out the steps. You will find that the general rule will still serve quite well.

Let's start with an input of 20. FIGURE 20 executes FD 20 RT 60, calls FIGURE 30 and waits. FIGURE 30 executes FD 30 RT 60, calls FIGURE 40 and waits. Once more we have an endlessly repeating cycle. But this time something is growing! The size of the FD input is getting larger with each successive call. Can you guess what the figure will look like?

What if we want to stop it? We could use TEST on the HEADING as before. But since we have a growing variable, we could just as easily TEST for a maximum size. Here is one way to do that. (Note that in this particular example, the TEST comes before the drawing commands. They could also be placed after them.)

```
TO FIGURE :X
TEXT :X > 100
IFTRUE STOP
FD :X RT 60
FIGURE :X + 10
END
```

Can you deduce what will happen? For an initial input of 80, FIGURE 80 tests, draws and calls FIGURE 90, which tests, draws and calls FIGURE 100, which tests (100 is *not* greater than 100!), draws and calls FIGURE 110. The value of the input variable :X is now greater than 100, so the TEST is TRUE. The IFTRUE instruction STOPs the procedure. FIGURE 100, FIGURE 90 and FIGURE 80, in turn, continue to their next instruction (END) and the entire action comes to a close.

Tail Recursion Exploration Ideas

Students need plenty of associated activities in order to understand the use of the general rule in tail recursion. Here is one which I especially like.

- Create a procedure with tail recursion. Write the procedure name and commands on a large card and give it to a student. The student assumes the role of the first procedure and carries out the instructions down to the recursive call. (The commands might be drawing on the board, calling out numbers, turning, walking, and the like.) The first student initials the card and passes it to a second student, who assumes the role of the next procedure. The process continues until the teacher interrupts, at which point the card is passed back up the line to each student in turn, so that it ends up with the first student. (This activity also could be carried out using the TEST commands.)
- Recursion could be used to model the classic situation of placing one grain of wheat on the first square of a chess board, two grains on the second, four on the third, and so on. Students could carry out part of the exercise physically. Then, using Logo, TEST and tail recursion, they could calculate the number of grains on any square.

```
TO GROW :GRAINS :SQUARE.NUMBER
TEST :SQUARE.NUMBER > 64
IFTRUE PRINT :GRAINS STOP
GROW :GRAINS*2 :SQUARE.NUMBER + 1
END
```

GROW 1 1 will get things started.

- Students could explore other projects and challenges such as more elaborate spirals, procedures to fill shapes and calculations of factorials. The OUTPUT command could be used instead of PRINT for some applications. Interactive procedures are also possible with recursion, and hold the potential for sophisticated development.

Varsity Recursion

Now it is time to take a look at a deeper part of recursion. I hope that you will find it equally simple. After all, we will be doing the same thing we have done to this point—using the general rule to trace out the development.

Let's use this procedure as a review.

```
TO WHIRL :X
TEST :X < 10
IFTRUE STOP
FD :X RT 90
WHIRL :X - 10
END
```

If we enter WHIRL 30, it will test, draw and call WHIRL 20, which will test, draw and call WHIRL 10, which will test, draw and call WHIRL 0. The TEST will be TRUE since 0 is less than 10. The IFTRUE command (STOP) will be executed. Meanwhile, WHIRL 10, WHIRL 20 and WHIRL 30 are waiting in line for the WHIRL procedure they called to finish. Thus, when WHIRL 0 stops, each of the waiting procedures carries out the next command (END) in turn. Finally, all action ceases.

Now, let's make a small change in the *WHIRL* procedure.

```
TO WHIRL :X
TEST :X < 10
IFTRUE STOP
WHIRL :X - 10
FD :X RT 90
END
```

Can you see the difference? The WHIRL recursive command has traded places with the movement commands. What will happen now? If you can trace through this successfully, you are beginning to see the potential and the power of non-tail (or embedded) recursion.

Use the general rule here, just as you have before. WHIRL 30 tests and calls WHIRL 20, which tests and calls WHIRL 10, which tests and calls WHIRL 0. Note that no drawing has taken place yet, because all the procedures are waiting. The TEST in WHIRL 0 is TRUE, so WHIRL 0 STOPs. Now WHIRL 10 can continue.

Note, however, that WHIRL 10's next commands are FD 10 RT 90, and *then* END! Next, WHIRL 20 executes FD 20 RT 90 END. Finally, WHIRL 30 contributes FD 30 RT 90 END.

What was the result of the drawing? The tail recursion drew a spiral from the outside in. The embedded recursion drew a spiral from the *inside out*! Try a larger input number such as WHIRL 100.

With embedded recursion, Logo has the ability to figure out a situation all the way to the end, start there and work back to the beginning.

This could be illustrated to your students with the same card activity as before. Use a procedure similar to the one below. Have students say their numbers out loud when they come to the PRINT command.

```
TO NUMBER :X
IF :X > 10 STOP
PRINT :X
NUMBER :X + 1
END
```

NUMBER 1 would result in a counting up, 1,2,3, 4,5,6,7,8,9,10.

Now, change the lines to read as follows.

```
TO NUMBER :X
IF :X > 10 STOP
NUMBER :X + 1
PRINT :X
END
```

If the students follow the instructions closely, NUMBER 1 will result in a countdown, 10,9,8,7,6,5, 4,3,2,1. Can you predict what the outcome of NUMBER 4 will be?

Recursion Blowout

I hope that you have enjoyed learning a little more about recursion. I would like to leave this article unfinished. Logo is like that. There doesn't seem to be an end. So I will make the following remarks, and stop writing for now.

Recursion is not limited to just one call within a procedure! Consider the following example.

```
TO NUMBER :X
IF :X > 10 STOP
NUMBER :X + 1
PRINT :X
NUMBER :X + 1
END
```

Can you predict the outcome for NUMBER 7? Believe it or not, the general rule still works! Good luck! END■

[Tom Lough, The National Logo Exchange, *P.O. Box 5341, Charlottesville, VA 22905.]*

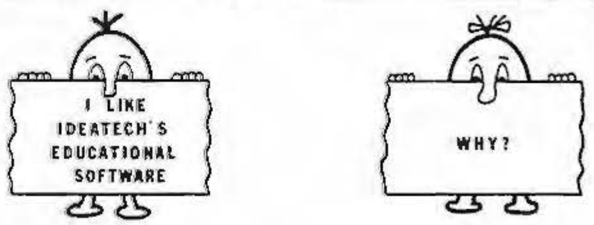
The Role of the Computer in Education IV

March 8-10, 1984
Arlington Heights, Illinois

Hosted by Micro-Ideas, the conference includes day-long seminars as well as shorter sessions.

For further information write Micro-Ideas, 1335 N. Waukegan Rd., Glenview, IL 60025, Attn: Rick Nelson, 312/998-5065.

advertisement



FEATURES:

Apple II, II+ & IIe
Back-up disk included
User's manual included
Classroom tested
Menu driven

IDEATECH COMPANY
P.O. BOX 62451 DEPT. G
SUNNYVALE, CA 94088

K-12 LANGUAGE ARTS, MATH & SCIENCE · FREE CATALOG

Helping Students with Recursion: Teaching Strategies

by
Tim Riordon

This paper has been inspired by the work of three other people. In a paper presented at the Fifth Annual Conference of the Cognitive Science Society, Kurland and Pea presented some disturbing findings about children working with the concept of recursion in Logo; namely, that students with a year of Logo programming are not understanding recursion. Kurland and Pea believe that their student subjects were misguided because they had constructed an incorrect "looping model" for recursion. A looping model works successfully in the context of tail recursion but fails when students encounter embedded recursion. These findings caused me to begin thinking about how to provide students with better mental models for recursion. As I began thinking about this problem, I found myself recalling the ideas that D. R. Hofstadter presented in a public lecture entitled "Recursion in Nature and Art." I wish to acknowledge Douglas Hofstadter as the inspiration and source of many of the ideas presented below.

"If students are to understand the concept of recursion, then that understanding should be grounded in a rich background of non-computer recursive experience."

The first mistake many of us make in presenting the concept of recursion is that we attempt to introduce the idea in a computer context. If students are to understand the concept of recursion, then that understanding should be grounded in a rich background of non-computer recursive experience. Students learn a turtle geometry, built upon the idea of defining everything in terms of moving and turning relative to one's present location and heading. Their learning is made possible and facilitated because of their rich background in moving and turning their bodies. I believe that my own past attempts to teach recursion to children were as misguided as it would be to try to teach turtle geometry to someone who lacked a background in seeing and feeling movement and turning in space. The first task in teaching recursion is to help students recall or to provide them with recursive experiences. The activities which follow attempt to do that.

A second mistake made by many of us who have focused upon recursion in a computer context is that our talk, our metaphors and our diagrams too often contribute to students arriving at a "looping" model of recursion. The following lesson plans avoid that mistake.

A third problem is that we teachers are often trapped into teaching a concept which we do not ourselves fully grasp. I believe that this article, in conjunction with Dan Watt's *Learning With Logo* and Hal Abelson's *LOGO for the Apple II,* will help teachers to better understand the powerful computer idea of recursion. So, another purpose of this article is to help teachers acquire a better understanding of this important concept.

These ideas are presented in a lesson plan format in the hope that teachers will actually use the suggested materials and activities.

Part I: Introducing Recursion with Non-Computer Recursive Experiences

TEACHER BACKGROUND

Take a moment to read through Student Activities 1-5, and then return to this background section and consider the following comments:

- The purpose of these activities is to provide students with recursive experiences—experiences that are associated with the word "recursion" itself. By giving students this word, you give them a way to talk about other experiences which they will encounter.
- These activities were selected because they expose students to *objects or processes which contain copies of themselves.* This is the aspect of recursion which we need to reinforce. This is the mental model we want students to employ instead of a "recursion is looping" model. In Logo, recursion involves procedures that in some sense "contain" themselves.

- It is important for students to experience recursion in both an auditory and visual way. Activities 2, 3 and 5 should not be skipped.
- It is important to have students try to actively create recursive productions. Activities 4 and 5 accomplish this.

MATERIALS

- 2 good-sized inexpensive mirrors, 12" x 20". One mirror should have a stand.
- Record player
- 2 tape recorders, one with variable speed control
- Recording of "Windmills of the Mind," a popular song recorded by a number of artists
- Video camera and monitor
- Morton salt container or Land of Lakes butter carton
- Transparencies I-IV (see p. 42)
- Cassette with the recording of "Teacher's Example of Recursion"

In order to create "Teacher's Example of Recursion," you'll need three students, the two tape recorders, 2 cassette tapes (labeled TAPE A and TAPE B), and Transparency III to serve as a script. Assign parts to each student.

1. Turn on the variable speed recorder and record at normal speed the students saying their lines on TAPE A. Encourage students to enunciate clearly and not to rush the tempo. Just skip over the "hole" in the script pattern this first time. Rewind this tape and get it ready to play. It helps if you have a silent microphone switch. Set the playing speed to high (about 25% faster than normal).
2. Now get the second recorder ready to record. This recorder doesn't need to be a variable speed recorder. Turn on the second recorder and have the students begin recording the same lines on TAPE B, but when you come to the "hole" in the script pattern (right after the line "*It sounds like this*"), switch on the first tape recorder. The first copy from TAPE A will then play back at a faster tempo and higher pitch. This is being placed into the "hole" in the script and is now being dubbed onto TAPE B. When this higher-pitched playback is finished, switch it off and have the student continue recording the last line of script onto TAPE B. This is the line that comes after the "hole." Then stop this recorder and rewind it. Play it back to see if you captured the copy of the conversation within the conversation.
3. Rewind the tape you just recorded (TAPE B). Now take that tape and place it in the variable speed recorder. Leave it set to play back at a faster speed. Have the switch ready so you can turn on this recording. Put the TAPE A in the regular tape recorder, erase it and get it set to record.
4. Turn on the regular recorder and begin recording the script on TAPE A. When you come to the "hole" in the script, switch on the variable speed recorder so that TAPE B plays back at a faster speed. Now TAPE B is being dubbed onto TAPE A in the "hole" in the script. This time two higher pitched copies of the script will be heard. When TAPE B is finished playing back, cue your student to say the final line and then cease recording on TAPE A. Rewind TAPE A and play it back. You will now hear a copy within a copy within a copy of the pattern script. Rewind this and place it in the variable speed recorder ready to play back once again.
5. Take TAPE B, erase it, put it in the regular recorder and prepare the recorder to record. Repeat the process until you have a tape that contains 4 or 5 copies within copies.

Don't be discouraged by the poor quality of recording you obtain. The value of this is not in the final product—it's the process that students go through to create the recursive example.

Douglas Hofstadter inspired this idea by playing a tape recording of a recursive ad for his book *Godel, Escher, Bach: an Eternal Golden Braid* (Basic Books, 1979).

ACTIVITIES

Activity 1: Visual Example

Have students take turns putting their heads between two good-sized mirrors which are placed so that they face each other. Allow them to adjust one of the mirrors. Ask them to comment upon what they see (an infinity of reflected images). Encourage students to verbalize the process that produces the infinity of images. (One mirror shows the other mirror that is showing the first mirror showing the other mirror. . . .) Don't over-emphasize the verbalizing aspect; your goal is to produce a right-brain understanding, not a left-brain verbalization. Once all students have experienced this, tell them that this experience has something to do with an idea called recursion. If they ask you, "What's recursion?" tell them that you aren't ready to explain it yet. If they really press you with, "Hey, you're a teacher and you're 'sposed to explain things," tell them that recursion is a process in which something contains a copy of itself. Leave the mirrors in the classroom for students to return to. If your students are older and know something about light and images, you might ask them whether there really are an infinity of images. (There aren't.)

Activity 2: Video Recorder, Microphone and Speaker

Bring a video camera and a monitor into your classroom. Encourage students to point the camera at the video screen. Ask them what they see. Encourage them to turn the camera and to hold it at various angles. Ask them to verbalize what's going on. Again, tell them that this process has something to do with an idea called recursion. There is something that contains a copy of itself. Most video camera setups have microphones and speakers. Turn up the volume on the amplifier and bring the microphone near the speaker. When you get the piercing hum, you can explain that this also has something to do with recursion. Help them understand what caused the hum. (Unfortunately, it's called a "feedback loop." This is unfortunate, because "loop" is the wrong word to use when talking about recursion.)

Activity 3: Musical Example

Play the song, "Windmills of the Mind." Tell students that this has something to do with recursion. Ask them to state the ways in which this song *contains a copy of itself.* Have them sing a round such as "Frere Jacques" or "Row, Row, Row Your Boat." Ask them to discuss whether this is a recursive experience. It isn't important to arrive at a definitive answer to these questions. Sometimes whether something is said to be recursive or not depends upon how the thing is being looked at or considered. Your purpose here is to start students thinking—not to have them *finish* thinking.

Activity 4: Recursive Drawings

Show students the picture on the Morton salt container. It shows a girl pouring a box of salt. The box of salt she is pouring has a picture on it. It is a picture of a girl pouring a box of salt which has a picture on it. The picture is that of a girl pouring . . . etc. Tell students that this has something to do with recursion, because there is something which contains a copy of itself. (Land of Lakes butter also uses a recursive design.)

Next, place Transparency II on top of Transparency I and show it to the class. Ask students whether this picture is recursive. (It is, because it is something that contains a copy of itself.) Remove Transparency II so that only Transparency I is projected. Do your students see that Transparency I is a non-recursive picture?

Have students pair up, and give out drawing paper. Each pair of students is to design a recursive drawing. Display these so everyone can view the pictures. A nice follow-up to this activity would be to show the film *Maurits Escher, Painter of Fantasies* (Coronet Films).

Activity 5: Recursive Scripts

Play the tape recording called "Teacher's Example of Recursion." (Directions for creating this tape are given in the Materials section above.) After students have listened to it, show Transparency III with Transparency IV placed on top. Play the tape again, and this time point to the lines of the script. Tell students that this is a recursive tape recording because the recording contains copies of itself.

Remove Transparency IV so that only III is projecting. Explain to students that what they are now viewing isn't recursive. But if we were to place inside this script another copy of this script, then the result would be recursive. Take Transparency IV, lay it over III and demonstrate. Remove IV again.

Tell students that Transparency III is a pattern or template with a hole in it. We become involved in a recursive process if we attempt to fill in the hole with another copy of the original pattern (template). But there is still a hole to fill. If we put another copy of the original into that hole, there is still another hole to fill. This process could go on forever!

Now ask the three students who made the tape recording to explain to their classmates how the recording was produced. (See above.) These students will serve as helpers for other recording teams wishing to create recursive recordings. Point out that they can create any pattern script they wish as long as they leave a "hole" in it. This hole is to be filled by another copy of the original pattern—which itself has a hole in it that has been filled.

[Editor's Note: Part II and III of this article will appear in the next two issues of TCT. Readers who wish full-sized, black on white masters for the entire set of 16 transparencies may order them by taking the following steps:

1. *Put 37¢ postage and your mailing address on a 9x12 clasp envelope.*
2. *Send the envelope and $2.00 to Tim Riordon, Alpine Educational Computing Services, 1633 High St., Eugene, OR 97401.*

TCT invites readers to submit other examples of recursive experiences.]

New Computer Books for Your Classroom

from dilithium Press

An Apple® For Kids
A PET® For Kids

Sharon Boren

A fresh, instructive, fun approach to teaching kids programming and computer operation! Written by a teacher who wanted to teach enthusiastic students about computers, these books are a delight to see and use. They are full of illustrations (a circus theme) and examples that appeal to kids and motivate them to higher learning levels.

Sharon uses a developmental approach to teaching good programming skills with a focus on problem solving, improved thinking skills and creativity.

The books are designed for children ages 8 to 13, and parts can be adapted for use by younger children as well.

An activity workbook is also available for both books. It includes over 75 tear-out activity worksheets, many of which can be done as seatwork (which helps solve the problem of 1 computer and 25 kids).

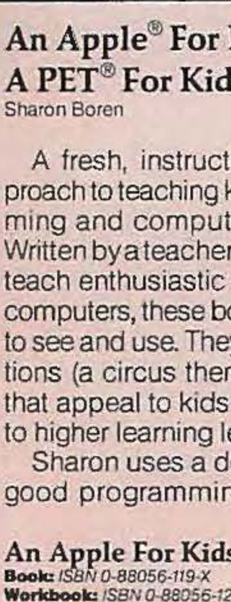
The teachers manual includes complete lesson plans, extra information that is useful, and answers to all student worksheets. When you order 25 activity workbooks, you receive a teachers manual free!

An Apple For Kids

Book: *ISBN 0-88056-119-X*	*148 pages/140 illustrations*	*$7.95*
Workbook: *ISBN 0-88056-120-3*	*175 pages*	*$5.95*
Teachers Manual: *ISBN 0-88056-118-1*	*90 pages*	*$14.95*

A PET For Kids (Examples work on the COMMODORE 64 and VIC 20)

Book: *ISBN 0-88056-106-8*	*148 pages/140 illustrations*	*$7.95*
Workbook: *ISBN 0-88056-122-X*	*175 pages*	*$5.95*
Teachers Manual: *ISBN 0-88056-121-1*	*90 pages*	*$14.95*



Computers, Teaching and Learning

Jerry W. Willis, D. LaMont Johnson & Paul N. Dixon

If you're interested in using computers in your classroom, this book covers the important topics of concern to you as an educator.

As teachers and educational administrators, you've been put in a peculiar position by the popularity and increasing interest in microcomputers these days. Not only do you have to know all about microcomputers and how to run them, but you must also decide which ones to buy, what software to buy, and which programming languages to teach.

This book gives you direction and support in this new teaching area. It deals with the practical issues (selecting hardware and software), and the problems (piracy, teacher training and the crisis of computer literacy).

ISBN 0-88056-065-7 *200 pages/100 illustrations* *$9.95*

Also available from dilithium Press

Computers For Everybody, 3rd Edition
Jerry Willis and Merl Miller

In a clear, understandable way, this new edition explains how a computer can be used in your home, office or at school.
ISBN 0-88056-131-9 *200 pages* *$7.95*

Brainfood: Our catalog listing over 130 microcomputer books.

dilithium Press books are available at your local bookstore or computer store. You can also call us to charge your order on VISA or MC—**(800) 547-1842** outside of Oregon, or 646-2713 in Oregon.

Instant BASIC— 2nd Astounding! Edition
Jerald R. Brown

Here is an active, easy, painless way to learn BASIC!
ISBN 0-918398-57-6 *200 pages* *$12.95*

dilithium Press

We're the number one publisher of easy-to-read computer books.

SEND TO: dilithium Press, P.O. Box E, Beaverton, OR 97075

Please send me the book(s) I have checked. I understand that if I'm not fully satisfied, I can return the book(s) within 10 days for full and prompt refund.

___ Computers, Teaching and Learning $9.95

___ An Apple For Kids $7.95

___ An Apple in the Classroom: Activity Workbook $5.95

___ An Apple in the Classroom: Teachers Manual $14.95

___ A PET For Kids $7.95

___ A PET in the Classroom: Activity Workbook $5.95

___ A PET in the Classroom: Teachers Manual $14.95

☐ Please send Brainfood, your free catalog.

☐ Check enclosed $________ Payable to dilithium Press

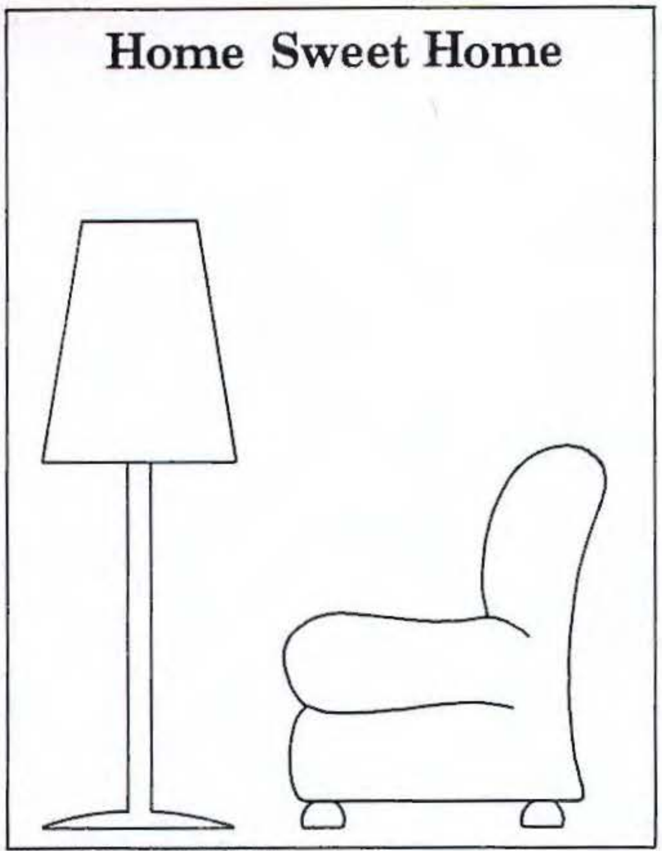
☐ Please charge my ☐ VISA ☐ MasterCard #________ Exp. Date________

Signature________

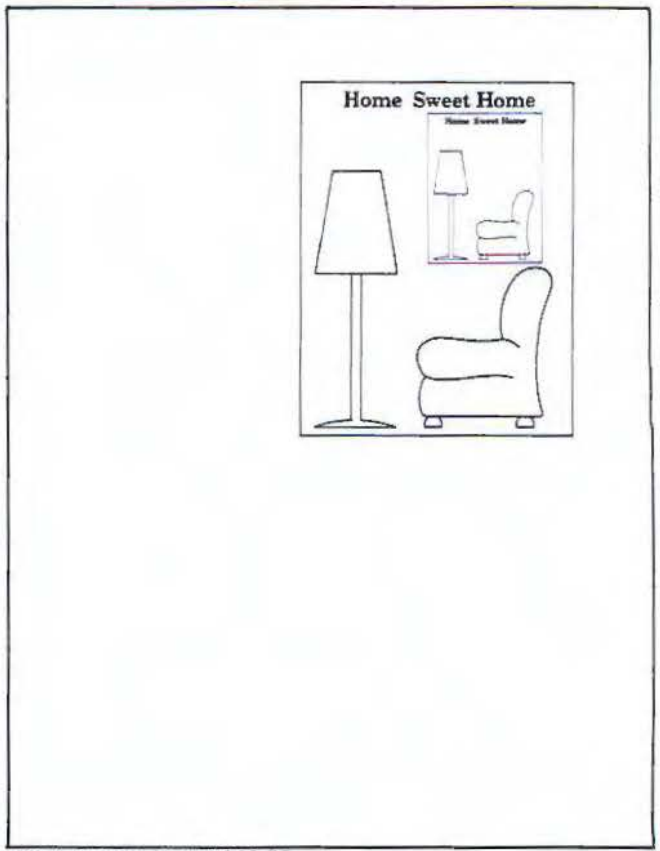
Name________

Address________

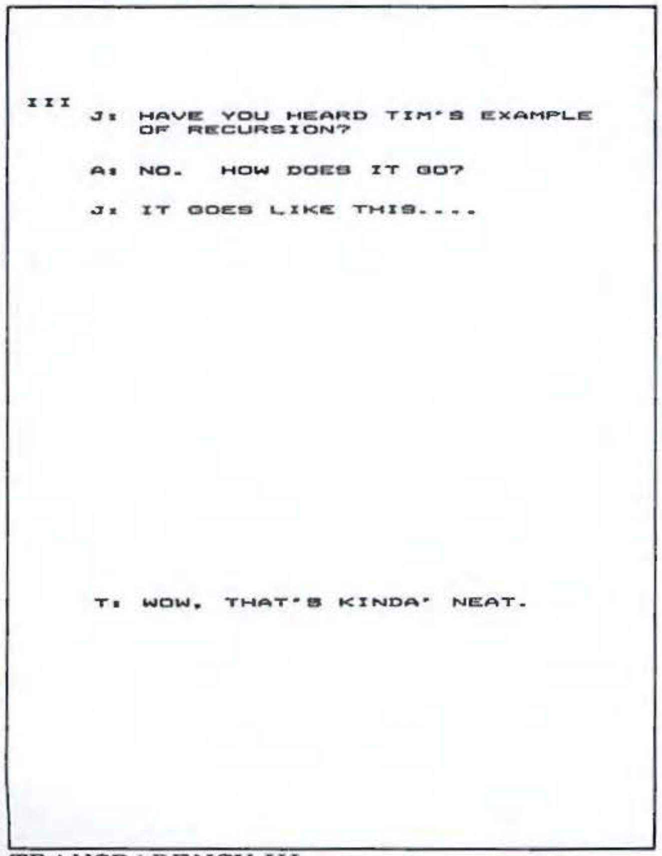
City, State, Zip________ 5



TRANSPARENCY I

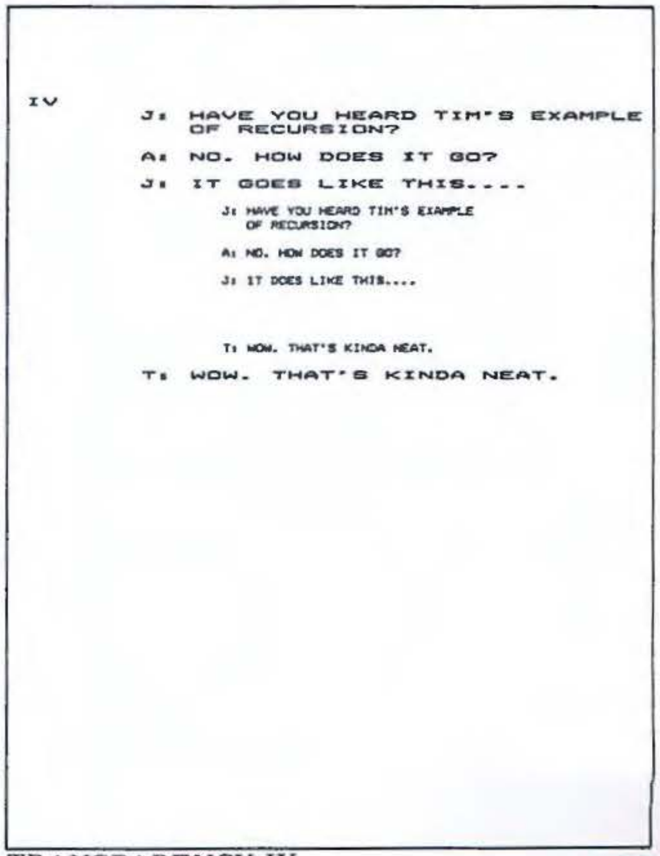


TRANSPARENCY II



```
III
      J: HAVE YOU HEARD TIM'S EXAMPLE
         OF RECURSION?

      A: NO.   HOW DOES IT GO?

      J: IT GOES LIKE THIS....




      T: WOW, THAT'S KINDA' NEAT.
```

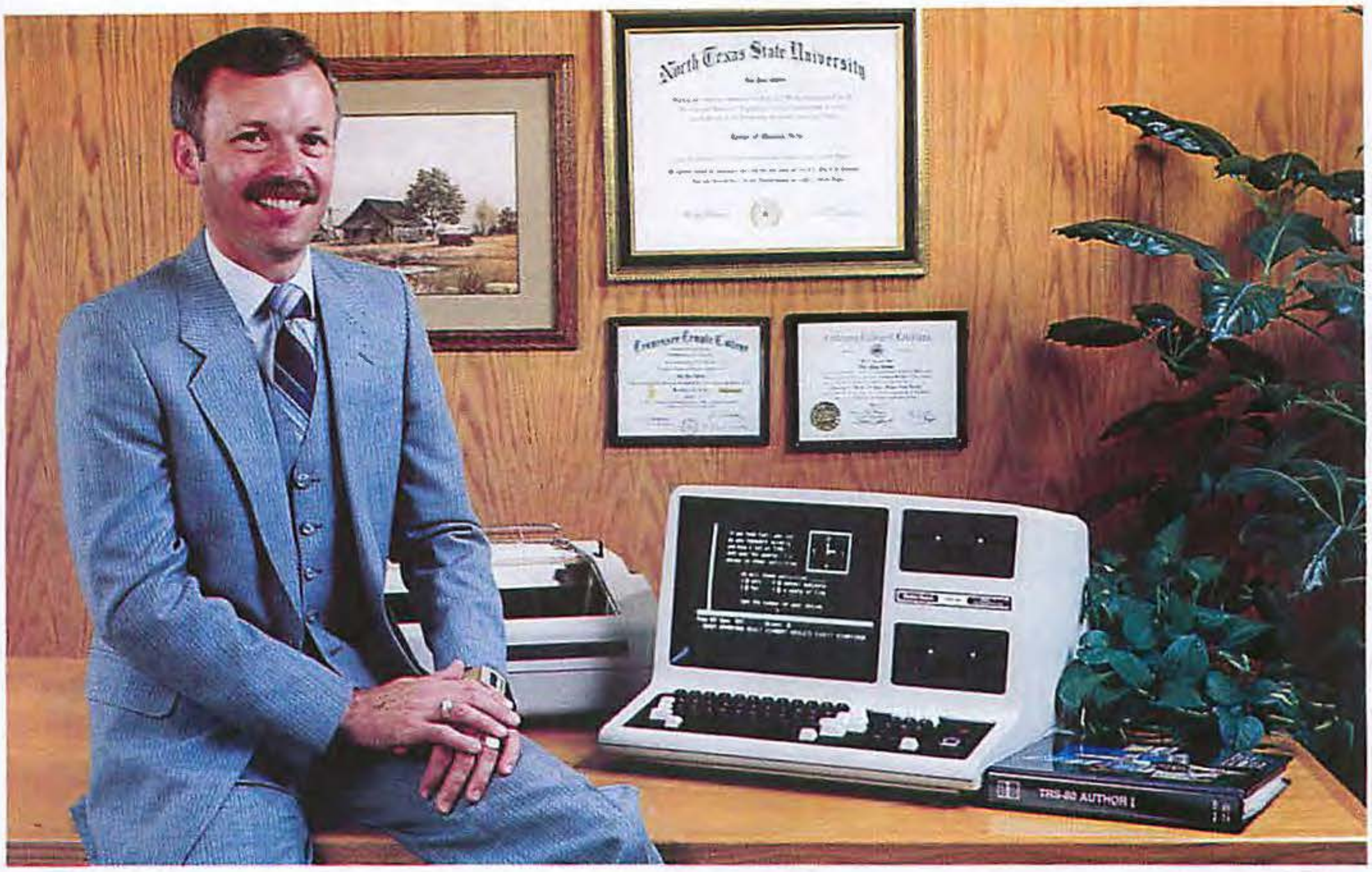
TRANSPARENCY III



```
IV
          J: HAVE YOU HEARD TIM'S EXAMPLE
             OF RECURSION?
          A: NO. HOW DOES IT GO?
          J: IT GOES LIKE THIS....
                  J: HAVE YOU HEARD TIM'S EXAMPLE
                     OF RECURSION?
                  A: NO. HOW DOES IT GO?
                  J: IT GOES LIKE THIS....

                  T: WOW. THAT'S KINDA NEAT.
          T: WOW. THAT'S KINDA NEAT.
```

TRANSPARENCY IV

END■



"The key to courseware development is a flexible, easy-to-use authoring tool."

—Dr. Dan Gibbs, Consultant
Radio Shack Education Division

"Radio Shack's TRS-80® AUTHOR I fills these requirements—and more!" AUTHOR I enables any teacher, regardless of computer experience, to easily create effective computer-assisted instruction materials.

"Courseware developed by teachers is more effective than courseware developed by non-educators." With AUTHOR I, teachers can maintain focus on the important subject matter, not a cumbersome computer language.

"AUTHOR I is a screen-oriented authoring system containing four types of 'pages' or displays." The developer simply "plugs in" various types of information—instructional text, question text, positive and negative feedback messages, glossary definitions and much more.

"The powerful text/graphics editor allows incredible word-processing-like flexibility." An AUTHOR I lesson can be edited and updated as many times as necessary, whether during the initial development process or after the lesson is used by students. And the branching capability allows students to take different paths based on individual response.

"AUTHOR I is the right price for any school district." TRS-80 AUTHOR I (Cat. No. 26-1727) is only $149.95. It requires a 32K TRS-80 Model I or Model III disk system (or 64K Model 4 in the Model III mode). Find out more at a Radio Shack Computer Center, Expanded Computer Department, participating store or dealer—or talk with your Regional Educational Coordinator.

For the name of the full-time Educational Coordinator in your area, call Radio Shack's Education Division at 800-433-5682, toll-free. In Texas, call 800-772-8538.

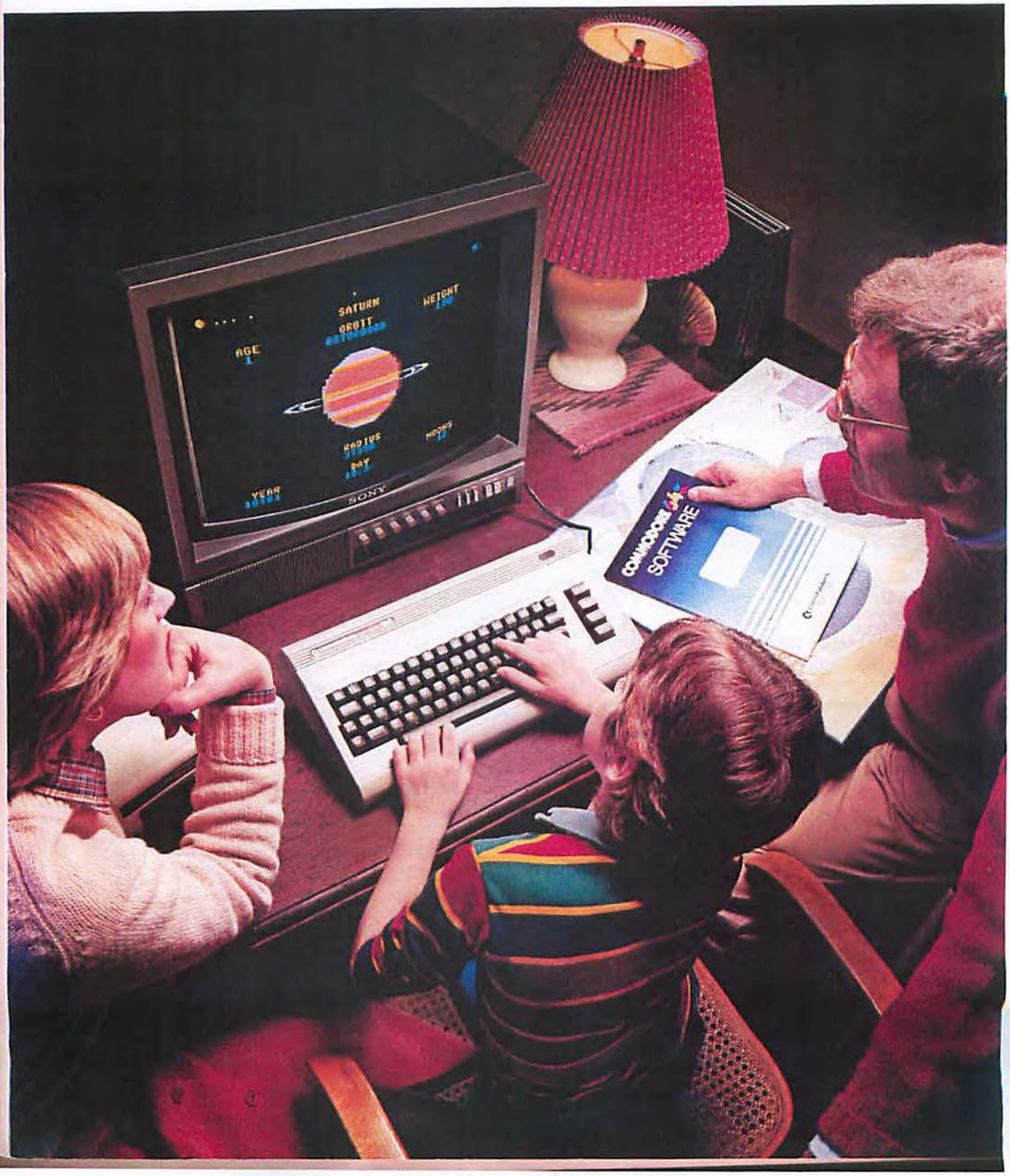
Radio Shack®
The Name in Classroom Computing™
A DIVISION OF TANDY CORPORATION

For more information about Radio Shack educational products and services, mail this coupon to:
Radio Shack, Dept. 84-A-396
300 One Tandy Center, Fort Worth, TX 76102

NAME ____________________
SCHOOL ____________________
ADDRESS ____________________
CITY ____________ STATE ______ ZIP ______
TELEPHONE ____________________

Price applies at participating Radio Shack stores and dealers. Special order may be required.

Commodore Con
The Best Ed



uters and Software ation Aid Yet.

From its beginning, Commodore has had a commitment to help educators and parents get the most for their computer dollars.

The Commodore PET system has, and continues to be, the core product for computer education in many school districts. The Commodore SuperPET with 5 high-level languages, is also in widespread use in schools and colleges.

Our new Commodore 64™ is the computer for both school and home. For example, the C=64 gives you a powerful 64K memory. That's as much memory as either the Apple® IIe or the IBM® Personal Computer. **But at far less than half the cost.** You also get a 9 scale music synthesizer, high resolution color graphics, and a wide variety of educational software. Now teachers and parents can work together to provide quality education for students.

Carnegie-Mellon Univ. may be first to make computers mandatory.

of Carnegie-Mellon, study electronic com- ils coming home late

DATA PROCESSING
STRUCTURED
CONSULTANTS/TRAINERS

NETWORK CONTROL TECH.

PROGRAMMER/OPERATOR

Computers reshape work habits at college

The University of Illinois in Urbana-Champaign, for example, has established 46 computer centers where dents can go to do

For Home and School—We've just released numerous educational software programs into the public domain. These programs, written by educators, include courses in Business, Computer Science, English, French, Geography, History, Mathematics. The list goes on and on.

We're also working with major educational publishers to develop new software. For example, a significant portion of the well-regarded MECC courseware has been completely adapted for the Commodore 64. The Edufun™ series from Milliken will be available for home and school use in the near future, and over thirty early learning programs from Midwest Software will help children master the basics.

In addition, we've developed a complete set of software tools to make our educational computers even more useful. Take Logo and PILOT, for example. These popular languages have been completely adapted for the Commodore 64.

Our Educational Resource Centers, 250 strong, continue to provide teacher support in computer use in the classroom, and the number is growing!

COMMODORE'S COMMITMENT & APPROACH BECOMES STRONGER AS THE DOLLARS GET TIGHTER.

GROW WITH US. For further information about software or the Education Resource Centers, contact Commodore or your nearest Commodore Education Dealer.

C= commodore
COMPUTERS
Committed to Excellence in Education

Commodore Business Machines—P.O. Box 500M, Conshohocken, PA 19428.
Canada—3370 Pharmacy Avenue, Agincourt, Ont., Can. M1W2K4

Recursion, Recursion

by
Rick Billstein and Margaret L. Moore

Logo is often referred to as a language in which users can discover powerful ideas on their own by experimenting in a Logo environment. Recursion is one of these powerful ideas. It can be simply described as the **ability of a procedure to call a copy of itself.** The simplicity of the description disguises the power of recursion. It is a concept that can be easily misunderstood.

Test your understanding of recursion by predicting the result of the following program. If your prediction does not match the figure on your output screen, read on.

```
TO PREDICT
FD 40
RT 60
TEST HEADING = 0
IFFALSE PREDICT
FD 40
LT 60
END
```

In a procedural language such as Logo, one procedure can call upon another procedure. Consider the two procedures called ONE and TWO.

```
TO ONE                 TO TWO
PRINT [THIS IS ONE]    PRINT [THIS IS TWO]
TWO                    PRINT [ENDING TWO]
PRINT [ENDING ONE]     END
END
```

Predict the results of the command ONE and then check to see if your prediction matches these results:

```
THIS IS ONE
THIS IS TWO
ENDING TWO
ENDING ONE
```

You were probably able to predict this result. A telescoping model which describes procedure ONE helps to describe the action.

```
COMMAND: ONE                                RESULTS:
┌──────────────────────────────────────┐
│ ONE                                  │
│  PRINT [THIS IS ONE]                 │   → THIS IS ONE
│  ┌───────────────────────────────┐   │
│  │ TWO                           │   │
│  │                               │   │
│  │   PRINT [THIS IS TWO]         │   │   → THIS IS TWO
│  │   PRINT [ENDING TWO]          │   │   → ENDING TWO
│  │                               │   │
│  │ END                           │   │
│  └───────────────────────────────┘   │
│  PRINT [ENDING ONE]                  │   → ENDING ONE
│                                      │
│ END                                  │
└──────────────────────────────────────┘
```

Notice that when procedure ONE is called, it proceeds line by line until it encounters the call for procedure TWO. At this point all the lines of procedure TWO are "inserted" and procedure ONE is not continued until procedure TWO is completed.

This model shows how one procedure calls another. What happens when a procedure calls itself? In other words, what happens on a recursive call? Predict the result of the command HEXAGONS 40.

```
    MIT Logo                        Apple Logo
TO HEXAGONS :N                  TO HEXAGONS :N
IF :N=0 STOP                    IF :N=0 [ STOP ]
REPEAT 6[ FD :N RT 60 ]         REPEAT 6[ FD :N RT 60 ]
HEXAGONS :N-20                  HEXAGONS :N-20
END                             END
```

HEXAGONS is an example of tail-end recursion, because it calls a copy of itself at the **tail end** of the procedure, just before the END statement. This code looks very much like a loop on paper. It also acts like a loop on the screen, because the Logo interpreter treats tail-end recursion differently from non-tail-end (embedded) recursion. (A Logo procedure with tail-end recursion but no STOP will tie up the computer indefinitely. A Logo procedure with embedded recursion will fill the computer's memory rather quickly.) The HEXAGONS procedure simply says, "Go back and do it again until it's time to stop, and when it's time to stop, execute the END statement."

Now change HEXAGONS so that it is no longer an example of tail-end recursion.

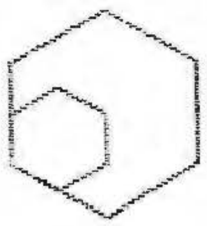
```
    MIT Logo                        Apple Logo
TO HEXAGONS :N                  TO HEXAGONS :N
IF :N=0 STOP                    IF :N=0 [ STOP ]
REPEAT 6[ FD :N RT 60 ]         REPEAT 6[ FD :N RT 60 ]
HEXAGONS :N-20                  HEXAGONS :N-20
PRINT [ NOT-TAIL-END ]          PRINT [ NOT-TAIL-END ]
END                             END
```

It is important to predict the results of the command HEXAGONS 40 and then test the procedure. How many times will the phrase NOT-TAIL-END be printed? Once? Wrong!

Let's use the telescoping model to examine non-tail-end recursion:

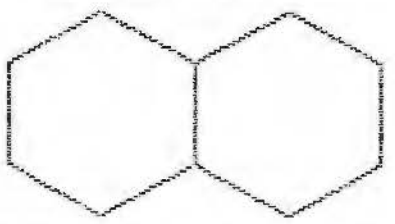
Command: HEXAGONS 40 RESULTS:

```
HEXAGONS 40
  IF :N=0 STOP
  REPEAT 6[ FD 40 RT 60 ]
  HEXAGONS 20
    IF :N=0 STOP
    REPEAT 6[ FD 20 RT 60 ]
    HEXAGONS 0
      IF :N=0 STOP                Stops!
    END
    PRINT [ NOT-TAIL-END ]        → NOT-TAIL-END
  END
  PRINT [ NOT-TAIL-END ]          → NOT-TAIL-END
END
```



From this model we can see how recursion works. When a procedure calls itself, execution of the calling procedure is *temporarily suspended* and control is passed to a copy of the calling procedure where :N is 20. Execution of the copy of the calling procedure is temporarily suspended and control is passed to another copy of the calling procedure where :N is 0. The STOP is executed in this procedure and control is returned to the calling procedure where :N is 20. This procedure *resumes where it left off* and executes the appropriate PRINT. When the END is executed, control is returned to its calling procedure where :N is 40. This procedure resumes where it left off and executes the appropriate PRINT.

Recall the procedure PREDICT. Can you use this telescoping model to explain these results?



The telescoping model demonstrates that embedded recursion definitely is not iteration or simple looping. If examples of tail-end recursion only are presented, students may think of recursion as looping. Certainly, many materials currently available contribute to this thinking.

So why bother with the distinction? Because recursion is a "powerful idea." Recursion is a compact way of thinking that enables us to describe and solve some complicated problems in terms of simpler versions of themselves. Here are some procedures to use to investigate the power of recursion. Can you predict the results?

Example One: Use the command CIRCLE 10

```
TO CIRCLE :S
IF :S=0 STOP
REPEAT 36[ FD :S RT 10 ]
CIRCLE :S-1
REPEAT 36[ FD :S LT 10 ]
END
```

Example Two: Use command HOUSES 30

```
TO HOUSES :S
  IF :S< 1 STOP
  SQUARE :S
  PU RT 90 FD :S * 2 LT 90 PD
  HOUSES :S-10
  PU LT 90 FD :S * 2 RT 90 PD
  FD :S
  RT 30
  TRIANGLE :S
  LT 30
  BK :S
END

TO SQUARE :S
REPEAT 4[ FD :S RT 90 ]
END

TO TRIANGLE :S
REPEAT 3[ FD :S RT 120 ]
END
```

Example Three: Use command REFLECT.POLYGON 8

```
TO REFLECT.POLYGON :S
IF :S < 3 STOP
REPEAT :S[ FD 30 RT 360/:S ]
REFLECT.POLYGON :S-1
REPEAT :S[ FD 30 LT 360/:S ]
END
```

Many of us adults will need to trace through these procedures carefully more than once to feel confident about what they do. Some children are learning to understand and create such procedures intuitively. Every time a group of people learns to think in a new way, the results are powerful—and incalculable.

END■

[Rick Billstein, Dept. of Mathematical Sciences, Univ. of Montana, Missoula, MT 59812; Margaret L. Moore, Assistant Professor, Dept. of Science and Mathematics Education, Oregon State University, Corvallis, OR 97331.]

☑ Check with Houghton Mifflin for quality software.

Houghton Mifflin has over 100 years of experience in developing content and refining teaching strategies. We've merged these ideas with technological expertise to offer you and your students the best instructional software and textbooks available. At Houghton Mifflin, whatever the medium, the message is quality.

Check the areas that interest you ... and we'll send you the information.

Subject	Level	Purpose
☐ Reading	☐ K–3	☐ Management
☐ Language Arts	☐ 4–6	☐ Reinforcement
☐ Mathematics	☐ 7–12	☐ Enrichment
☐ Business Education	☐ Adult	☐ Instruction
☐ Computer Education		

Name ____________________

Position ____________________

School ____________________

School Address ____________________

City ____________________

State ________ County ________ Zip ______

Houghton Mifflin Instructional Computing

Please mail this coupon to: Houghton Mifflin, Dept. J, One Beacon St., Boston, MA 02108

Houghton Mifflin

Atlanta, GA 30360/Dallas, TX 75234/Geneva, IL 60134/Hopewell, NJ 08525/Palo Alto, CA 94304/Toronto, ONT L3R 1B2/Boston, MA 02108

A Recursion Excursion with a Surprising Discovery

by
Margaret L. Moore

[Editor's Note: This article discusses recursion in Logo. All examples have been developed with Terrapin/Krell Logo format. They can easily be converted to Apple Logo.]

Explorations with Logo procedures frequently begin with a SQUARE.

```
TO SQUARE
  REPEAT 4 [FORWARD 50 RIGHT 90]
  END
```

And, with the SQUARE procedure there must be a TRIANGLE.

```
TO TRIANGLE
  REPEAT 3 [FORWARD 50 RIGHT 120]
  END
```

Surely you have the urge to say, "Oh no, here comes HOUSE again!"

Certainly there must be something else that can be done with a square and a triangle. One of the most intriguing aspects of Logo is that it encourages the question, "Can you do it another way?" This question is a form of lateral thinking which produces surprising results in the classroom. It challenges students to find different solutions, and encourages exploration and search for a more complete understanding of the figure and programming in Logo.

I find that the notion of "more than one way" is strange to many students. They mistrust me, certain that I am still looking for *the* way and frustrated because they haven't yet discovered it. It is difficult to convince students that you really are encouraging them to find other ways! But once they begin trying to find other solutions, the atmosphere in the classroom is rewarding.

So, can you make the figures in SQUARE and TRIANGLE another way? Try it. Here are some travels of my students.

How about a procedure that can do both?

```
TO POLYGON :SIDES
  REPEAT :SIDES [FORWARD 50 RIGHT
    360/:SIDES]
  END
```

The turtle draws a triangle with POLYGON 3 and a square with POLYGON 4.

What about a recursive solution?

```
TO SQUARE
  FORWARD 50
  RIGHT 90
  SQUARE
  END

TO TRIANGLE
  FORWARD 50
  RIGHT 120
  TRIANGLE
  END
```

Do these procedures teach the turtle to draw a square and a triangle? My students say yes, "... but the poor turtle keeps drawing the figure over and over and over."

Students know I always have more questions and rarely give a solution. If they are concerned about the turtle being overworked, they must find a method of stopping the extra work. Their question is typically, "Does Logo have an IF (something is done) THEN STOP statement?" My answer, as always, is, "I don't know; you tell me." They try!

```
TO SQUARE
  IF DONE THEN STOP
  FORWARD 50
  RIGHT 90
  SQUARE
  END
```

When this procedure is called, the turtle politely explains that it doesn't know the word DONE. That must mean it knows IF. Someone must decide what it means to be DONE. The turtle is DONE IF the four sides of the square have been constructed. The students need a way to count the sides. They try variable inputs:

```
TO SQUARE :COUNT
  IF :COUNT=5 THEN STOP
  FORWARD 50
  RIGHT 90
  SQUARE :COUNT
  END
```

However, this procedure doesn't change the :COUNT. They ask, "Can we add in a recursive call like SQUARE :COUNT?" I answer, "I don't know, try it!"

```
TO SQUARE :COUNT
  IF :COUNT=5 THEN STOP
  FORWARD 50
  RIGHT 90
  SQUARE (:COUNT+1)
  END
```

The command SQUARE 1 works!

A solution . . . and other questions.(There are so many possible questions that it is hard to choose one!) Can you make it count backwards and still draw a square? What happens if you call it with SQUARE 5?

When students pick their favorite solutions for teaching the turtle to draw a square and a triangle, I try another twist with the original recursive procedures.

```
TO SQUARE                TO SQUARE
  FORWARD 50               FORWARD 50
  RIGHT 90        →        RIGHT 87
  SQUARE                   SQUARE
  END                      END
TO TRIANGLE              TO TRIANGLE
  FORWARD 50               FORWARD 50
  RIGHT 120       →        RIGHT 118
  TRIANGLE                 TRIANGLE
  END                      END
```

What happens if these changes are made? The challenge is to draw the figure yourself *before* you have the turtle draw it.

These procedures are different, but most students don't find them terribly exciting. Okay, make the FORWARD length variable. Can *you* tell what happens with SQUARE 30 and TRIANGLE 70?

```
TO SQUARE :LENGTH        TO TRIANGLE :LENGTH
  FORWARD :LENGTH          FORWARD :LENGTH
  RIGHT 87                 RIGHT 118
  SQUARE :LENGTH           TRIANGLE :LENGTH
  END                      END
```

These procedures are still boring. Make the :LENGTH change. Can *you* figure out what happens with SQUARE 30 and TRIANGLE 70 now?

```
TO SQUARE :LENGTH
  FORWARD :LENGTH
  RIGHT 87
  SQUARE (:LENGTH-2)
  END
TO TRIANGLE :LENGTH
  FORWARD :LENGTH
  RIGHT 118
  TRIANGLE (:LENGTH+5)
  END
```

Now it's exciting! But the turtle is being overworked again! What is happening in SQUARE 30 when the call hits SQUARE 0? SQUARE −2? SQUARE −4? . . . Students discover that FORWARD −2 is the same as BACK 2. This is an interesting and usually unexpected result for most students' first experiences with recursion! But by now they can supply some stopping action for these recursive calls. One of many possible solutions is:

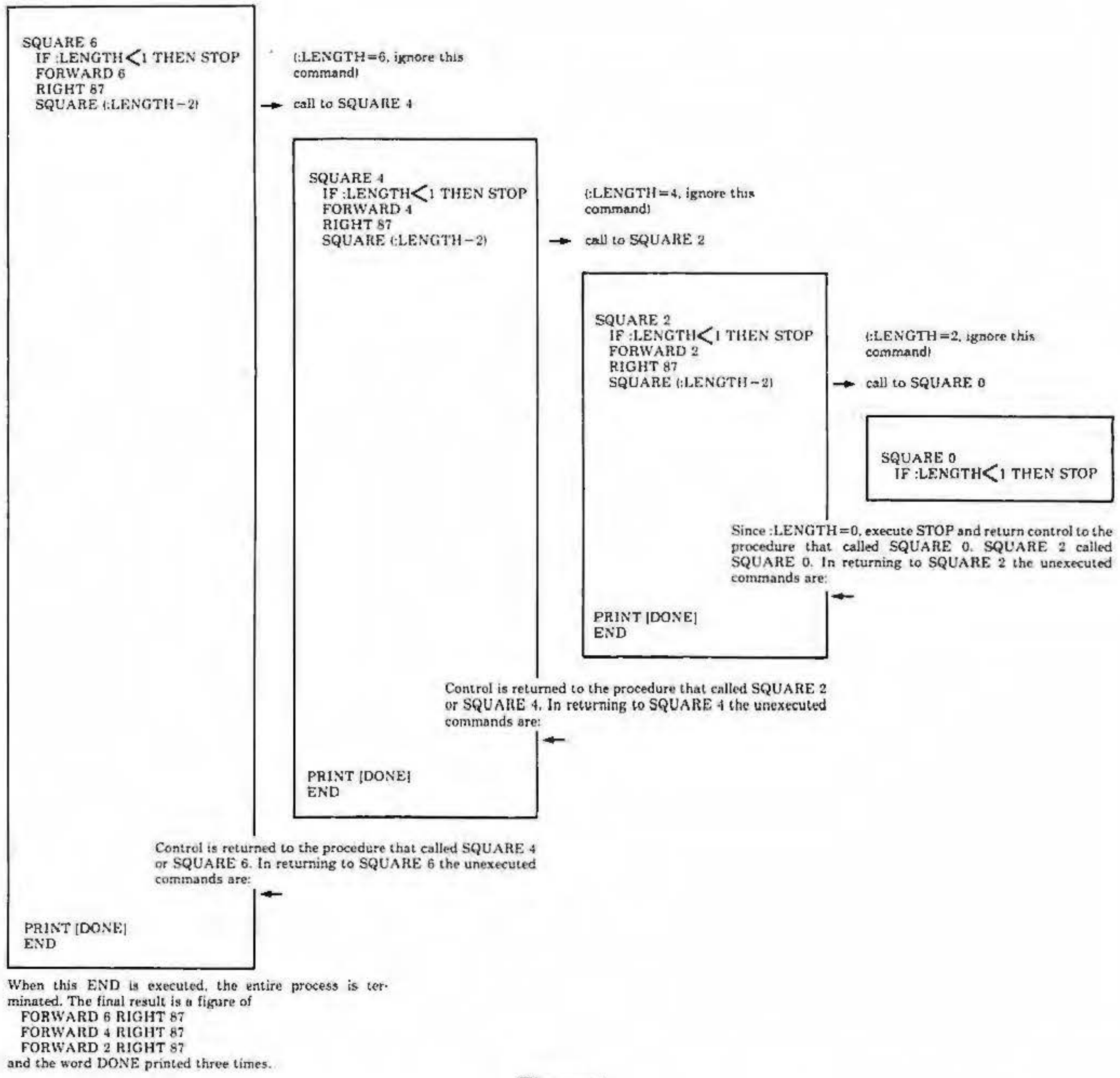
```
TO SQUARE :LENGTH
  IF :LENGTH <1 THEN STOP
  FORWARD :LENGTH
  RIGHT 87
  SQUARE (:LENGTH-2)
  END
TO TRIANGLE :LENGTH
  IF :LENGTH > 100 THEN STOP
  FORWARD :LENGTH
  RIGHT 118
  TRIANGLE (:LENGTH+5)
  END
```

You must know by now that there are still questions to be asked. What happens if one simple command is added to each procedure *after* the recursive call?

```
TO SQUARE :LENGTH
  IF :LENGTH <1 THEN STOP
  FORWARD :LENGTH
  RIGHT 87
  SQUARE (:LENGTH  -2)
  PRINT [DONE]
  END
TO TRIANGLE :LENGTH
  IF :LENGTH >100 THEN STOP
  FORWARD :LENGTH
  RIGHT 118
  TRIANGLE (:LENGTH  +5)
  PRINT [DONE]
  END
```

What does the turtle do if the SQUARE procedure is called with SQUARE 30? Guessing *before* trying the command is essential at this point. Guesses are a commitment to understanding recursion. Until this example, all the recursive procedures demonstrate TAIL-END recursion. The procedures have the appearance of looping. Students have the mistaken impression at this point that recursion and looping are identical structures. The addition of *one* command after the recursive call effectively demonstrates that *recursion is not looping.* If the command is SQUARE 30, why is DONE printed 15 times after :LENGTH becomes less than 1? In the procedure the command PRINT [DONE] appears only once *after* the recursive call, SQUARE (:LENGTH −2)! The answer to the question requires a better understanding of recursion.

Investigate with a simpler call, SQUARE 6. Figure 1 describes what happens.



```
SQUARE 6
  IF :LENGTH<1 THEN STOP          (:LENGTH=6, ignore this
                                  command)
  FORWARD 6
  RIGHT 87
  SQUARE (:LENGTH-2)          →   call to SQUARE 4

    SQUARE 4
      IF :LENGTH<1 THEN STOP      (:LENGTH=4, ignore this
                                  command)
      FORWARD 4
      RIGHT 87
      SQUARE (:LENGTH-2)      →   call to SQUARE 2

        SQUARE 2
          IF :LENGTH<1 THEN STOP  (:LENGTH=2, ignore this
                                  command)
          FORWARD 2
          RIGHT 87
          SQUARE (:LENGTH-2)  →   call to SQUARE 0

            SQUARE 0
              IF :LENGTH<1 THEN STOP
```

Since :LENGTH=0, execute STOP and return control to the procedure that called SQUARE 0. SQUARE 2 called SQUARE 0. In returning to SQUARE 2 the unexecuted commands are:

```
        PRINT [DONE]
        END
```

Control is returned to the procedure that called SQUARE 2 or SQUARE 4. In returning to SQUARE 4 the unexecuted commands are:

```
    PRINT [DONE]
    END
```

Control is returned to the procedure that called SQUARE 4 or SQUARE 6. In returning to SQUARE 6 the unexecuted commands are:

```
PRINT [DONE]
END
```

When this END is executed, the entire process is terminated. The final result is a figure of

```
FORWARD 6 RIGHT 87
FORWARD 4 RIGHT 87
FORWARD 2 RIGHT 87
```

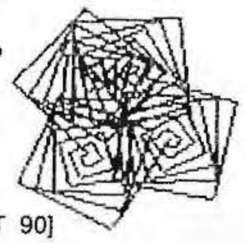
and the word DONE printed three times.

Figure 1.

If the command has been SQUARE 30, the recursive calls would have continued until SQUARE 0 caused the STOP to be executed. The STOP only affects the current procedure, SQUARE 0. Then the process becomes one of backing out, completing each calling procedure and returning control to the previous calling procedure. With SQUARE 30, the backing out process causes DONE to be printed 15 times. It's your turn to explain what happens with TRIANGLE 70.

Once when I was teaching this concept, the students expressed the feeling that while the idea was new, they wondered what good it was. I suggested that they might try to rotate the original design created with SQUARE 50 (without the PRINT [DONE]). They developed this approach:

```
TO SQUARE :LENGTH
  IF :LENGTH <1 THEN STOP
  FORWARD :LENGTH
  RIGHT 87
  SQUARE (:LENGTH - 2)
  END
TO ROTATE.SQUARE
  REPEAT 4[SQUARE 50 RIGHT 90]
  END
```



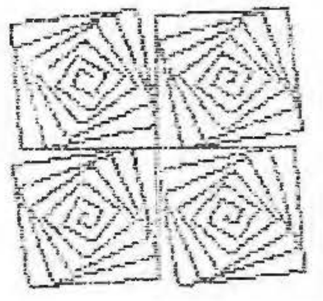
They brought the solution to me wondering, "What good is recursion if you get lost and can't

BACK OUT." The words "BACK OUT" suddenly sparked *my* curiosity. If recursion was all I had told them, then it must be able to be used to "BACK OUT." I was so intrigued with my discovery that I took the keyboard from the students and began to EDIT SQUARE. I wondered what it meant to "BACK OUT" of the square-like figure. It occurred to me that backing out meant to go in the opposite direction, doing the opposite things. Using this idea, I edited SQUARE.

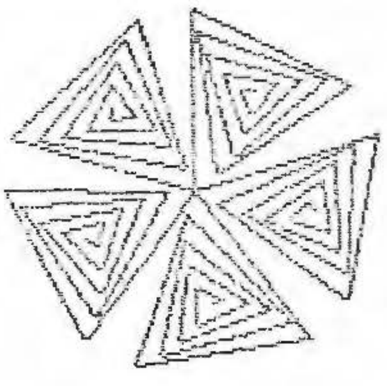
```
TO SQUARE :LENGTH
  IF :LENGTH < 1 THEN STOP
  FORWARD :LENGTH
  RIGHT 87
  SQUARE (:LENGTH - 2)
  END
```

```
TO SQUARE :LENGTH
  IF :LENGTH < 1 THEN STOP
  FORWARD :LENGTH
  RIGHT 87
  SQUARE (:LENGTH - 2)
➡ LEFT 87
  BACK :LENGTH
  END
```

The critical discovery was that each calling procedure *remembered* what value it was using for :LENGTH. SQUARE 30 used :LENGTH=30; SQUARE 20 used :LENGTH=20; SQUARE 10 used :LENGTH=10. In backing out of the procedure calls, the turtle was backing out of the design. With this addition to SQUARE, ROTATE.SQUARE became:

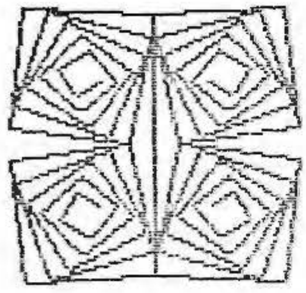


Applying the same ideas to TRIANGLE, this design can be created. Can you figure out how?



One last challenge (the students in my classes know that there is no such thing):

If you can turn **right** in making SQUARE, why can't you turn **left**? You probably thought that turning left was much too simple a solution for the problem at the beginning of the article (how many different ways can you make a square?). But what if you do left turns in some recursions with SQUARE, say SQUARE.LEFT, and right turns in others, say SQUARE.RIGHT? Can you create this design?



Although my students were impressed with the whole new field of designing with recursion, they recognized that asking questions and looking for other solutions was a significant part of the learning process with Logo. The spark of an idea I received from their comments and questions startled them. They seemed amazed to see me struggle for a solution. Perhaps they learned a more important idea than recursion—they observed a teacher wrestling with a problem and recognized similarities with their own struggles. They saw that the process was perhaps more exciting than the end product—a most surprising and rewarding discovery. With a questioning framework there is no limit to the explorations, challenges and excursions with Logo and recursion. END■

[Margaret L. Moore, Assistant Professor, Dept. of Science and Mathematics Education, Oregon State University, Corvallis, OR 97331.]

Computers in the NEW Curriculum

February 29-March 3, 1984 Austin, Texas

This Texas Computer Education Association State Conference features Arthur Luehrmann, who will keynote on Thursday morning, and Jim Poirot, who will speak in a featured session on teacher training on Friday morning. Pre-conference workshops will be held on Wednesday.

For further information, contact TCEA-State Conference, P.O. Box 2573, Austin, TX 78768.

Problem Spaces in a Project-Oriented Logo Environment

by
Glen Bull and Steve Tipps

[Editor's Note: This article will also appear in the December and January issues of The National Logo Exchange. *This article may be reprinted provided full credit is given the authors and TCT.]*

A recent review in *Consumer Reports* characterizes Logo as a language that has gained a wide following among educators as a way to teach programming concepts to children before they have sufficient reading, writing, and math skills to tackle the BASIC language.[1]

Programming concepts *are* readily learned through Logo, but Logo is more than an efficient way to introduce programming. Such misunderstandings about Logo are not uncommon. Many introductory texts discuss Logo solely as a "language for children" or as a "way to teach computer literacy." These statements unduly restrict Logo's intent, both as a language and as an educational philosophy.

"The challenge to the Logo community is to provide projects which involve learning about content or subject matter without resorting to direct instruction with the computer as a teaching machine. Logo should make the computer a learning machine."

Using Logo also develops problem-solving skills; the structure of the language encourages a modular approach to problems. Features such as a resident screen editor facilitate a strategy of successive approximations to a solution, since procedures can be altered in seconds.

Logo can also be a vehicle for learning content. Papert presents the imagery of learning math in "Mathland" in the same way that it is possible to learn French through a stay in France. Acquiring programming and problem-solving skills, and developing concepts in subjects such as math and English are all important parts of a Logo environment.

Of these three, the potential of Logo as a meaningful way of learning content has been neglected. Using Logo for learning subjects ranging from beginning reading to college physics have been imagined, but such applications, though being explored, are not yet in widespread use. Classroom activities which use Logo to explore and develop concepts in subject areas are needed as alternatives to traditional didactic teaching styles. Otherwise, the Logo environment is incomplete.

PROJECT-ORIENTED LEARNING

Consider the following proposition:

"Effective Logo learning is project oriented."

In a project, the content to be learned is embedded in a goal meaningful to the learner. The need for personally satisfying goals can be found in initial use of the REPEAT statement. The first need for repetition can be induced by a desire to avoid repeated retyping of an instruction, but children soon discover great joy in the unpredictable effects of randomly-typed commands repeated a large number of times. These seemingly random instructions often produce beautiful and symmetrical designs. Some exploration of this type must be encouraged. However, failure to go on to purposeful activity will produce a row of children gazing in stupefied boredom at 5,000 shifts in the background color of the screen.

To return to Papert's imagery, French is most effectively learned when it is embedded in another context and is not in itself the primary goal. Yet even the presence of the French environment is not enough—Americans in Paris have survived on hamburgers from American hamburger chains.

Subjects such as mathematics and English are also learned best when embedded in a meaningful project. To the student using Logo, the subject matter content learned should simply be something which must be understood in order to do something else which the student wants to accomplish.

Finding projects and applications which address specific content areas seems to be a prime difficulty in implementing Logo-based school programs. The

[1]"The Computer as Teacher." *Consumer Reports.* Vol. 48 (November 1983), p. 617.

problem stems from perceptions of limitations and insufficient illustrations of classroom use rather than the language itself (any memory limitations aside). The relative newness of Logo on widely available microcomputers means that most teachers and students have not pushed the boundaries of Logo beyond turtle graphics. The notion that teacher selection of specific content invalidates the discovery principle is also an inhibiting factor.

Removing these barriers permits a broader view of Logo and makes opportunities for learning content areas with Logo more apparent.

PROBLEM SPACES

Two dimensions are important in defining how Logo might be used in schools:

- Who provides the idea for the project?
- Who solves the problem and who programs the solution?

A two-dimensional matrix defines four possible problem spaces in a Logo environment:

	Source of the Project	
Implementation (Solution)	LEARNER	FACILITATOR
LEARNER	I	III
FACILITATOR	II	IV

I. Learner-Generated/Learner-Implemented Problem Spaces

In a Type I project, the learner generates both the problem and the solution. This is often considered to be the most natural, "Logo-like" environment.

Even in Type I Problem Spaces the teacher has considerable control over the content to be learned. If the teacher supplies the set of primitives dealing with turtle graphics, the learner is apt to develop projects which involve geometry, physics and trigonometry. If the teacher supplies primitives which can be used to take apart and combine words and sentences, the learner may embark upon poetry generation and other projects which involve a natural language.

If the teacher supplies tailor-made procedures, the learner is apt to explore that particular set of ideas. A BOUNCE procedure modeling reflection of light rays or billiard balls would encourage exploration of vectors.

The underlying structure of Logo encourages strategies such as patterning, modularity, comparison and approximation. At the same time, the learner discovers the possibilities and limitations of the rules which govern the natural microworld of the turtle. This world offers opportunities for learning a new vocabulary, enhancing thinking skills and discovering a consistent rule set.

Ideally, each new project inspires further learning. Each bug, error or "what if" demands satisfaction and solution. The need for REPEAT begins with pattern recognition. The rationale for procedures is developed when Logo users need an easy way to recall previous events and change them. Implementation of REPEAT statements or of the editor cannot be transmitted telepathically. Providing information about the vocabulary and structure of Logo is the facilitator's responsibility. A facilitator must at times instruct.

The natural learning space of exploration in Logo is not without difficulties for both learners and facilitators. This environment of exploration demands a non-authoritative role for teachers. When a learner asks, "Can I accomplish this?", the answer may be, "How do you think that might be done?" or "Show what you want the turtle to do." These replies are invitations for learners to extend their skills. Some people discover new applications for known skills as they describe their intended goal. Others seem to refuse the challenge of thinking about their own question. They are discouraged because Logo is not automatic—like a video game.

II. Learner-Generated/Teacher-Facilitated Problem Spaces

In a Type II Logo project, the learner may generate an application which requires skills beyond those possessed. The teacher can choose to direct the learner to an area which is more consonant with that learner's existing level of computing skills. However, there can be value in working out the description (and, if possible, a solution) of the problem in English, even if someone else codes the program. A system analyst, for example, may work out the flowchart or pseudocode for a problem that is coded by a team of programmers. In a Type II project, the learner fills the role of system analyst, and the teacher serves as programmer by providing the actual coding.

Beyond the skills required for development of the logic underlying the solution, two other benefits may result from a Type II project. The learner may have the opportunity to observe firsthand the false starts and problems encountered as the teacher attempts to code the problem. When a learner asks how to do something which is not immediately apparent to the teacher, the question may become the basis for a cooperative project. This type of collaboration is not possible when the learner senses that the teacher already knows the answer.

The second benefit is the possible introduction of a new command or concept in a meaningful context. The teacher's version of the program can serve as a model or exemplar which the learner can use to generate solutions to similar problems.

III. Teacher-Generated/Learner-Implemented Problem Spaces

Some children invent their own projects, while others languish for lack of direction. One reason for tiring of Logo is a low skill level. Another is lack of interesting examples. Potential solutions for ennui in Logoland are ideas which build skills or provide models for exploration. The teacher generates a project for student solution in a Type III problem space.

A Type III project can become a static model. Asking students to type in a POLYSPI procedure which they don't understand is not a valid Type III project. The problem is not with POLYSPI, but with the short-circuiting of the learning in POLYSPI through imposition of a solution. Teachers grounded in the purpose of Logo do not require a rigid scope and sequence chart with fixed activities; besides, students already receive enough practice in doing things they don't understand.

Teachers do need many open-ended ideas and projects. Characteristic of a good Type III project is a clear objective which can be reached in a variety of ways. A didactic approach can involve a specific suggestion:

"Place a square inside a circle."

A less structured approach could include the suggestion for a range of projects appropriate for the learner, using one procedure as an example:

"This procedure places a diamond in a circle. Can you create other shapes which fit inside? What other shapes could fit inside each other?

Long-range cooperative projects may be even less structured. The term *problem schemes* has been invented to describe long-range classroom projects which emphasize applications of Logo in content areas. A class project to create a variety of increasingly sophisticated programs to generate random sentences according to linguistic rules is an example of a problem scheme.

In this instance the teacher has deliberately chosen a context which requires deliberation on the nature of English syntax. The incongruities generated will inevitably create questions about the ways in which words work together. Such projects have an ideal Logo flavor, because the ideas sparked can lead in a variety of directions. This situation may be as threatening as a non-authoritative role for some teachers. The teacher directing the project must have a firm grasp of the potential subject matter embedded in the project.

Teachers need explanations of how content embedded in a project can be developed in a classroom setting. Discussions of subject matter are missing; the emphasis is on programming concepts required to conduct a project. For example, ANIMAL is a well-known program in which the computer attempts to guess an animal by asking questions such as:

- Does the animal have four legs?
- Does it bark?

The program makes a final guess about the animal, asking: Is it a dog? Or is it an elephant?

This is a potential problem scheme for the classroom. When developing this project, a class will discover why some nouns have the article "a" before them, while others have the article "an." Members of the class can become linguists, attempting to discover patterns which govern sentence structure. Questions arise: "Why do vowels exist?" "What *is* a vowel?" In this type of project, the class confronts natural rules found in the English language, rather than disassociated rules memorized by rote from a workbook.

The greatest potential of Logo in a classroom is its possible use for elimination of disassociated learning. This potential will be neglected as long as extensive illustrations of how Logo can be used to learn specific content are not available.

IV. Teacher-Generated/Teacher-Implemented Problem Spaces

The teacher supplies both the idea and the implementation of the project in a Type IV situation. At first glance, Type IV problems seem to be at odds with Logo philosophy, but problems of this sort may be worthwhile to illustrate a new capability of the language or to seed new student-generated problems.

If a student has only seen turtle graphics, a program which uses list processing to manipulate English or to graph a linear equation may be a revelation. Music generators can provide "hooks" for students to create procedures for their own projects. Tom Lough's *(The National Logo Exchange)* alphabet procedures use the turtle to draw letters which can be used in student projects.

Templates provided by teachers also may be a Type IV project. A poetry formatter which asks for words to complete a haiku or cinquain form might entice students to explore those forms. The questions, asked in a fill-in-the-blanks way, make apparent the form of the poetry.

When teachers first learn Logo, they gravitate toward another kind of Type IV project—drill-and-practice or tutorial programs, which are easy to write in Logo. Teachers develop their own skills in programming Logo with such projects, but using them in class to teach content is inherently anti-Logo and has nothing to do with Logo for student use.

A CALL FOR TYPE III PROJECTS

Valid projects from all four problem spaces should be used in a complete Logo environment. Logo's potential is reduced if it is restricted to one type of problem. Type I problems permit free exploration and play. Type II problems invite teacher-learner collaboration. Type IV projects expand the use of Logo in areas which exceed the learner's programming skills. Type III projects are valuable because they are particularly suited to the use of Logo in learning subject matter.

Logo with Content

Where are the projects in Logo which focus on learning experiences in algebra, social science, industrial arts and other content areas? The Logo community has avoided describing Logo projects of this kind for fear that descriptions might become recipes—explanations that shift into rigid lock-step programs deemed to be "the right way to do Logo" or "the right thing to do with Logo." A written description of a project tends to capture product rather than process. It is a still picture rather than a film.

Some teachers do not recognize that using Logo to teach a subject is even a possibility. A high school math teacher asked, "Can Logo help me teach algebra?" Other teachers realize the potential but find it difficult to develop methods of using Logo to teach specific concepts. How can Logo be used to understand multiplication of polynomials? Traditional methods have focused on mechanical techniques which can be used to obtain the answer without understanding the process.

For example, the solution to the following operation can be determined with mnemonic FOIL (First, Outer, Inner, Last) which specifies all possible paired combinations of the elements:

$$(5X + 3)(2X + 1) \text{ becomes } 10X^2 + 11X + 3$$

The teacher who attempts to present this concept through Logo is first tempted to create a tutorial with Logo. After recognizing the discrepancy between the intent of Logo and this tutorial approach, the second effort may involve the use of Logo to create a graphing tool. This becomes a Type I project for the teacher, but is a Type IV tool for students. From the student's perspective, it doesn't matter whether the graphing tool is written in assembly language, BASIC or Logo. Calling the graphing tool a microworld doesn't alter the situation.

A Proposed Model

What is needed is a project which will allow students to use Logo directly to explore the meaning of polynomial multiplication. One approach involves a geometric method utilizing turtle graphics. A second technique might be based on exploration of the distributive law using list operators. Several stages are likely before a Logo project actually reaches the classroom. Implementing content-laden projects may include:

- Recognition of the potential of using Logo for learning specific concepts
- Identification of a concept
- Determination of avenues of attack with Logo
- Development of the problem scheme
- Use in the classroom

The fourth stage involves the design of an experience in teaching without telling. **A true Logo problem demands inquiry and discovery.**

A model problem scheme in the Type III category (teacher generated/student implemented) should contain a full English description of the problem with a variety of possible directions in which the project might go. Exploration of area is one possible direction for binomial multiplication, but others might be suggested. The teacher needs an understanding of the concepts potentially embedded in the project. Given the nature of Logo, this frequently includes more than one subject area. With the distributive approach to binomial multiplication, compound words and sentences as well as combinatorial theory are outcomes.

The project would not include a full programming implementation to reduce the temptation to use the project in a static, cookbook-like fashion. At most, key programming ideas or program stubs might be suggested. The challenge to the Logo community is to provide projects which involve learning about content or subject matter without resorting to direct instruction with the computer as a teaching machine. Logo should make the computer a learning machine.

Problem schemes in many subject areas are essential if Logo is to go beyond computer literacy. This is a complex task—we still do not have fully realized Logo projects for binomial multiplication and have only arrived at our present understanding after several weeks of thought and sharing with high school mathematics and science teachers with strong Logo backgrounds. No single teacher will have the resources to create a series of projects spanning a school year. The challenge to the Logo community is to jointly develop and share ideas for projects of this kind. END■

[Glenn Bull and Steve Tipps, School of Education, University of Virginia, Charlottesville, VA 22903.]

SOFTWARE REVIEWS

Doris Carey

Apple Logo: Sample Program Disk and Tool Kit for the Apple II Family (1983)

Logo Computer Systems, Inc., 220 Fifth Ave., Suite 1604, New York, NY 10001.

Cost: Variable. The software is available from your Apple dealer. Some dealers will copy the program onto your three blank diskettes; others may charge a minimal fee for a copy of the diskettes (with the fee covering the cost of the diskettes). The documentation can be printed out using Applewriter. (Your Apple dealer may be willing to print it out for you free or for a small fee.) The disks and documentation can be obtained by sending $12.95 to Logo Computer Systems, Inc. For the documentation only, send $3.25. (Both prices include postage and handling.)

Hardware Required: Apple II, II+, IIe

Additional Requirements or Features: Apple Logo Disk

Reviewed by: Larry Francis, 2626 Thompson Creek Road, Applegate, OR 97530.

It would take a large pamphlet or a small book to adequately cover this package of extraordinarily useful programs which Apple and LCSI have generously placed into the public domain. Fortunately, such a thing already exists, both in booklet form and on disk, so that you can get yourself acquainted with the kit and then make any changes you wish, tailoring it precisely to fit your needs. I borrowed the booklet from my Apple dealer and found it so convenient that I've sent $3.25 to LCSI to get a copy of my own. (A book's an awful lot easier to leaf through than a disk.) In this review I'll recap the kit's contents, highlighting TEXTPRO, a toy word processor, and MUSIC—two programs I found immediately and especially useful.

The "Sample Programs" disk contains student-written programs, demonstration graphics programs, language programs, games and simulations. FIRST.DEMO and POLYS introduce the disk and turtle graphics, particularly the exploration of incremental changes in simple recursive procedures. I modified POLYS to include INSPI 10 1 10, one of my favorites from Harold Abelson's excellent book, *Apple Logo* (BYTE/McGraw-Hill, Peterborough, NH, 1982). There follows a series of eight nice student-written procedures of widely differing elegance and complexity. Both teachers and students ought to find the documentation's brief discussion of these projects stimulating and helpful. An additional student project, ANIMATE, is included in the more sophisticated graphics programs that make up the next nine files on the disk. My favorites were GONGRAMS, ROADRUNNER and MONDRIAN. GONGRAMS, also my children's favorite, draws beautifully-colored designs on the screen. There's nothing interactive about it—it's just lovely to watch and enjoy. ROADRUNNER makes a nice use of SETBG 6, draws houses in perspective and landscapes the whole scene with a liberal planting of recursive trees. The turtle is given a cute new role in ROADRUNNER and rises, beeping, to the occasion. In MONDRIAN, the turtle creates randomly generated "paintings" that also depend partly on the value of a variable input by the user.

Next come seven language programs. The first few of these generate phrases, rhymes or sentences from their own lists or new ones input by the user. To speed up SILLY, the first of the series, I inserted a procedure, S, to save the trouble of typing PRINT SILLY each time:

```
TO S
PRINT SILLY
END
```

SILLY was fun, but I wanted to make its linguistic product a little more complex by connecting some verbs and prepositions with a period (e.g., FISHED.FOR, MOONED.AT, etc.).

WORDGAME, the next program, is an excellent example of the way Logo can transform what normally would be a drill and practice situation where the computer programs the child, into an interactive exercise in which the child is responsible for the raw materials of the curriculum while the program provides the structure. Logo's list-processing capabilities make it easy to build language microworlds in which children can acquire a concrete, manipulative feel for linguistic structure by literally playing with their own language. It wouldn't be hard to expand WORDGAME's lists of nouns and verbs to include adverbs, adverbial phrases, and so on.

I liked the third program, LANGUAGE.PLAY so much that I decided to enhance it with a test for singular and plural nouns and verbs so users could enter both nouns and verbs in either singular or plural form and still have the randomly generated sentences come out grammatically correct. Another addition was the separate listing of singular and plural articles. In keeping with Logo's low-threshold, high-ceiling philosophy, I want students to be able to use and enjoy this neat program with only a bare minimum of prerequisites (i.e., without having to worry about understanding and being able to classify nouns, verbs and articles into singular or plural forms). Accordingly, I inserted an a/an discriminating procedure and restricted possible articles to *the*, *a* and *an*. Another problem was teaching Logo how to make plurals correctly, when to add *-s* and when to add *-es*, and how to reduce plural nouns and verbs to singular form, i.e., when to subtract *-s* or *-es*. (For a complete listing of my revised edition of LANGUAGE.PLAY, send a SASE and $2.00 to Larry Francis at the address above.)

RANDOM.POETRY uses lists of nouns and rhyming verbs to create a series of two-word rhymes. One of the rhyming verbs, *alone*, isn't a verb at all, but since these procedures are so easily modified, it's easy to fix. In PIGLATIN, students should be reminded that it's important to mark objects with beginning quotes (for words) or with brackets (for lists), so Logo knows they are objects to be manipulated and not commands.

The "Games and Simulations" section of the disk includes eight files. ANIMAL.GAME is a dandy and appears to be the same as the Young Peoples' Logo Association version, with the welcome addition of an introductory INFO message. It does, however, share a small error-trapping problem with the YPLA version. When ANIMAL.GAME has failed to guess the name of the animal you were thinking of, it asks you the animal's name so it can start a new branch on its tree of knowledge. If you press RETURN without entering anything, ANIMAL.GAME prompts you with a cute "C'mon, what was it really?" So far, so good. But when you finally do identify the animal, Logo responds with an "I DON'T KNOW WHAT TO DO WITH (whatever you entered)" error message. I spent more than several hours trying to untangle things and trap that little error so that I could present it triumphantly here in this review, but I failed miserably. There are still at least a few elements of Logo's flow of control and variable-handling mechanisms that remain mysterious to me. Another program, LUNAR.LANDER, just plain didn't work until I changed FENCE to WINDOW, eliminating the OUT OF BOUNDS problem that kept the program from running. DYNATURTLE, an especially nice simulation of a "Newtonian object," was marvelous. The only problem I had was me—I had a heck of a time steering that Newtonian turtle in the direction I wanted it to go. Somebody ought to add a congratulatory message for those who succeed in landing the turtle in the target area and maybe a procedure that uses the text section of the splitscreen for a running display of the turtle's heading and distance from the target's center.

I've found it very instructive, often fun, and sometimes devilishly frustrating to get into these programs and fiddle with them, dressing some up with a few frills, adding some color to others, maybe even attempting extensive additions or wholescale remodeling. These disks are especially valuable for just this kind of activity. In concept and execution, the programs are generally marvelous and neat. Besides their practical utility, they are gold mines for the curious Logo user and would-be programmer who wants to pick up a few new slick techniques and gambits. In this context, I ought to mention a funny book, *How Tom Beat Captain Najork and His Hired Sportsmen* (by Russell Hoban, illustrated by Quentin Blake) because of the way it celebrates this very notion of "messing around." Even the most casual look at the lives of great figures in every discipline—Einstein, Bourzaki, Dorothy Sayers, Beverly Sills, Chet Atkins, and of course multitudes of others—reveals the importance of being able to be playful with the materials of one's art or craft. The exemplary programs on these two disks create environments in which children can play with math, language and music in intellectually stimulating and academically worthwhile ways. Entirely apart from the rather crass goal we teachers sometimes have of raising children's achievement in school, these microworlds introduce kids to the intrinsic structure and beauty of the subjects themselves.

I really like being able to change these programs to suit my needs or my fancy. Impenetrable ready-made materials, "teacher-proofed" and impossible to modify, besides being inflexible, carry with them the implication that teachers are pretty dumb and not likely to get much smarter very soon. These expectations, subtle or not, are not only bad for our own self-esteem and accordingly inhibit our own personal and professional growth, they foster the most appallingly rigid ideas about curricula, limiting the play of our students' imaginations (not to mention our own) and hampering their individual and collective preparedness for whatever surprises await us *homo sapiens*.

There are a couple of programs on the TOOL KIT disk that I would transfer to the SAMPLE PROGRAMS disk: TEACH and ZOOM. These programs allow the beginning user to explore turtle graphics and even, in TEACH, to define what the user has already drawn as a procedure. The TOOL KIT also contains a tremendous assortment of handy programs, some that facilitate floor turtles, plotters and printers, others that save pictures, make primitives, plot curves, manipulate sets, and insert remarks in Logo programs. There's even a Logo assembler. Out of all this, the two files that my kids and I most enjoyed and that should be most immediately useful to teachers are TEXTPRO, a toy word processor, and a series of files called MUSIC.

My family and I found TEXTPRO easier to use than Bank Street Writer (though certainly not as powerful) and, since it was part of the public domain and we already had Apple Logo, much less expensive. After becoming familiar with word processing using TEXTPRO, our whole family graduated to Applewriter II on which Veronica, now in fourth grade, does much of her homework. As the documentation explains, TEXTPRO can only handle about a screenful and a half of text at any one time. If you enter more text than that, you get a TOO MUCH TEXT message when you exit the editor with CTRL-C and you lose what you've just entered. "There are ways to get around this problem," says the documentation, "but they are tricky to use." Nothing could have been more titilating to this reviewer (who wants to know how to use the tricky ways most of all).

I couldn't resist calling LCSI and was fortunate to get to talk with Mark Jeffery, TEXTPRO's author. Mark was particularly helpful in elucidating this problem, the same one on which the documentation was so coy and obscure. He told me that the difficulty arises because Logo has to remember your original definition as well as the version you've just finished working on in the editor. This may be a common cause of those dreaded OUT OF SPACE messages in complex programs. One good way to get around this, Mark explained, is to go to the top line of your procedure in the editor and then open a new line by pressing CTRL-O. On this new line you should enter the command ER (erase) *name* as in the example below:

```
ER "WHATEVER
TO WHATEVER
```

In this way, you can create your own customized version of T.P.INFO and maybe trick your way out of some of the other knotty workspace problems you encounter in Logo from time to time.

Since our family enjoys playing music as well as listening to it, and since both Veronica and Wayne are becoming fairly proficient on the pennywhistle, a "low-threshold, high-ceiling" instrument if there ever was one, I was especially anxious to try out the MUSIC procedures. Upstairs, next to the computer, we have a little Yamaha electric keyboard for plinking out tunes before writing them into procedures using NOTE. Since we're not especially adept at reading music and since we do have an electric keyboard nearby, we found the COMPOSE procedure rather cumbersome to use. We couldn't get the list of notes named TUNE into the editor where we could use it to define the procedures that would play the song. We found it easier first to use keyboard, paper and pencil to get the tune down and then enter the editor and use NOTE to write the procedures that would play the tune. This leads me to suspect that the ultimate low-threshold Logo music program is still waiting to be written. I'd like to see one along the lines of TEXTPRO that would allow kids to plunk out their tunes, perhaps on the number keys, define it *post facto* as in TEACH, and edit it in procedural rather than list form. But even so, just as they are, without any improvements or embellishments, these MUSIC tools—especially NOTE—have made it possible for us to appreciate and *do* musical composition in a structured, modular way.

Perhaps the most useful thing I learned from this gargantuan package, even more useful than the erase-before-exiting-the-editor gambit, was how to use the *name* STARTUP: by typing MAKE "STARTUP [INFO] and then defining TO INFO such that it prints instructions on the screen, you can write slick, uptown introductions for each of your programs that are displayed automatically on the screen as soon as the program is loaded. You can set things up so the user can get things rolling on his/her own by pressing RETURN or typing a simple command. Kids often like to be left alone to explore a microworld for themselves. But without instructions, exploration can become more than heroically difficult, and some form of hovering adult becomes necessary. I think we often fail to realize how quickly our available assistance becomes obnoxious to potential recipients and how much of their mental workspace gets taken up just wishing we would go away. Liberal use of MAKE "STARTUP [INFO] and clear instructions can deliver kids and teachers from each other's unwanted company, freeing both for more worthy pursuits.

This latest offering from LCSI provides a wealth of exemplary programs for our kids and us to use, emulate and modify; extends Apple Logo's capabilities to include word-processing and music composition; and doesn't cost anything at all (beyond the price of three blank disks and some fan-fold paper). It has to be one of the more stunning recent developments in the Logo craze.

Logo (1983)

Commodore Business Machines Inc., 1200 Wilson Drive, West Chester, PA 19380.
Subject: Computer Programming, Computer Literacy
Target Age Group: K-Adult
Hardware Required: Commodore 64, disk drive
Cost: $99.50 (Discounted as low as $45.67.)
Reviewed by Richard W. Haller, University of Oregon

Most readers of *The Computing Teacher* are already familiar with Logo through reading about it if not through using it. This review does not attempt to critique Logo in general, assuming that the reader is interested in Logo and wishes to know the values and defects, if any, of this particular implementation. I am an experienced programmer with teaching experience at the university level; however, my 14-year-old daughter, Sarah, sat down with the manual and worked through the material as much as possible on her own. Her comments are incorporated into this review. Her previous experience with computers was limited to game programs.

This version of Logo represents an exceptional value, both in comparison to other software available for the Commodore 64 and to versions of Logo available for other microcomputers. It was developed for Commodore by Terrapin, Inc., who earlier developed a highly-regarded version for the Apple II. Indeed, the C-64 version bears a strong family resemblance to the earlier Apple version. The major difference is that the C-64 hardware cost is roughly one-third that of the Apple.

In addition to providing all the Logo primitives and procedures one would expect, C-64 Logo supports convenient use of the 64's unique "sprite" graphics and three-voice sound synthesizer. As a matter of fact, it is much easier to write programs using these features to advantage in Logo than it is in Commodore's own BASIC. Likewise, the DOS command allows the user to issue Disk Operating System commands from within Logo more conveniently than from BASIC. The screen editor embedded in Logo (which is superior to that provided with BASIC), coupled with intelligible error messages and solid debugging capabilities (e.g., TRACE, PAUSE, CONTINUE), make program development a pleasure. C-64 Logo (like Terrapin Logo for Apple) even contains a package of procedures allowing the user to write assembly language subroutines that can be called from a regular Logo program. The overall effect creates an environment where the development of programming skills is permitted and encouraged.

The materials supplied include one copy of the Logo system itself on a single diskette, a second diskette with over fifty demonstration or utility procedures written in Logo and a 350-page tutorial and reference manual. The manual, aimed at secondary school students and above, is excellent. (A *Logo Primer,* aimed at elementary students, is being developed. The motivated student or teacher should be able to successfully master all the material with virtually no previous background in computers and with only occasional assistance. The quicker student may find, as my daughter did, that some of the sections move rather slowly. On the other hand, most students working in the sprite section will probably need some teacher assistance.

Numerous suggestions for interesting projects are provided in addition to examples which are worked out in detail in the text. The teacher will find that most of the work necessary to develop a Logo course has already been done. An added instructional benefit is that both examples and projects are in large part self-grading; that is, since they are programs, either they work or they don't, and the error messages and errant program behavior tend to encourage and facilitate users in correcting their own problems. The major responsibilities of the teacher are to encourage good programming style, resolve ambiguities or gaps in the text and provide an occasional hint.

I have two major criticisms of the C-64 Logo package. First, only one copy of Logo is provided, and it does not seem possible to back it up. While the replacement cost of $5 for a damaged diskette is quite reasonable, the user is without a functioning Logo while waiting for the replacement. I would rather see a back-up copy provided with the package, even at a higher cost, with replacements available as needed. This is particularly true since one is warned not to leave a disk in the drive while powering up or down. In a classroom environment, particularly with younger children, accidents are only a matter of time. Since younger children are one of the targets of Logo, it makes more sense to anticipate and provide for this possibility.

My second criticism is the turtle display. A triangle to represent the turtle's location and orientation is a Logo tradition. Even with the "bottom" side of the triangle drawn twice as thick as the other two, the orientation of the turtle is only clear when it is facing one of the four major compass points (NSEW). Because of the limited dot resolution of the video display (320 horizontal and 200 vertical), the rotated turtle shape is distorted and it is virtually impossible to decide which side is the bottom and therefore which vertex is the "head." I suspect that this problem is not unique to the 64, but will be characteristic of all home computer implementations due to the limited dot resolution. I would prefer a shape for the turtle that is less ambiguous in its orientation when rotated.

I used Commodore's 1701 color monitor. An additional problem I encountered was that many lines did not have consistent colors. Depending on the background and pen colors chosen, lines can change their color with location and orientation to the extent that one would not be willing to bet what color they were intended to be. For example, a single vertical line can have a completely different color than it is supposed to, depending on where it is drawn. (DOUBLECOLOR mode will, however, help this problem at the cost of decreased resolution.) I had the most success with yellow or black on a white background and the least with red on a black background. You will wish to experiment. It is possible to change the defaults for the DRAW and TEXT modes to your own choice, as well as change them dynamically. I found the "prismatic" white letters on the black background while in EDIT mode particularly distracting. I did find a discussion on how to change them. On the other side, rather than being annoyed as I was, my daughter found the unstable color effects in the DRAW mode delightful. However, if you are going to concentrate on geometric figures or graphs of functions, you may wish to use a black and white monitor, or turn the color control on your color monitor all the way down. *[Editor's note: Commodore Business Machines Inc. informs us that the "chroma" effect described by Richard*

Haller can be reduced to a minimum by plugging the monitor into the back *of the computer rather than the front.]*

Some minor criticisms. The material on workspace management could be better organized and given more emphasis. It turns out that it is possible to copy a procedure by editing its name and redefining it. To get the effect of renaming, you copy and then erase the old version. This is not obvious in the manual. Information on how to "clean up" your workspace via the ERASE command, especially its selective use, is listed under "Defining and Editing Procedures" instead of "Filing and Managing Workspace." I would have preferred more emphasis on saving selected procedures. If you forget to surround the SAVE command and its list of selected procedures to be saved with the required parentheses, no error or warning is generated. Instead, the whole workspace is saved. This can lead to mysterious problems in the future if you recall the saved material, thinking it only contains the procedures you meant to save.

The manual states that "sprites can do anything that the turtle can do." This is not completely correct. In particular, unlike the turtle, sprite shapes do not rotate in response to RT or LT commands. A related problem involves the behavior of the STAMPCHAR command. Since this was implemented with the 64's built-in screen memory structure rather than attempting to simulate a true bit-mapped graphics screen, not only do the characters or graphic symbols not change their orientation to match that of the turtle, but they are restricted in their possible locations to one of the 40 by 25 possible character locations in the 64's text screen display. The turtle or sprites can take on any of the 64 locations within the 8 by 8 dot matrix that forms each character location. As a result, moving the turtle and stamping another character will not always work as expected unless one moves in increments of 8. These are perfectly reasonable limitations and are found in other home computer versions of Logo—I just wish they had been explicitly and prominently stated. Adding these capabilities to Logo by creating new Logo procedures would make a nice advanced project. Of course, the ultimate results would be subject to the same distortion problems as the rotated turtle.

Overall, this is an exceptional value. Anyone who is seeking a friendly and stimulating environment in which to learn or teach programming skills would be hard put to find something better. And the price (including hardware) is *very* right.

No Threshold, No Ceiling ... Really!

by
Dr. Whatson and Shurlook 'olmes, YPLA, PC, DR, WMI
as told to
James H. Muller

"Dear Dr. Whatson, isn't it nice to be rid of those dreadful animals. That turtle! What's his name? Logy, or some such abomination. And that dreadful rabbit, Morf. How insufferable.

"I do mean, after all. Everyone knows of the lagomorphs, that lower order of gnawing mammals with two pairs of very nasty incisors; and those horrible ears—always dropping into the tea. I mean, Whatson, just how . . . how colonial!"

"Now 'olmes, tolerance, tolerance. Everyone has to have his/her due. That turtle and rabbit have done an admirable job with what they have had to work with. But now we will have a chance to show what can really be done with the Logo language."

"Ah, yes, the PC. Fascinating device. And it is about time someone gave us a version of Logo that will really accomplish something worthwhile."

"Please, 'olmes. Don't be so chauvinistic. While previous versions have not been able to do as much as the new versions of Logo for the IBM PC, they have much to offer, especially to young people."

"But, dear Whatson, that is the whole problem. Everyone thinks that Logo is just a children's language, a common 'toy.' And it's just because they haven't had the chance to do any really productive work with it. They keep finding themselves 'Out of Space.'"

"Tis true, 'olmes. Logo is far more than just a children's language for drawing pictures and creating word games. Most importantly, it gives you the chance to manipulate numbers, character strings and entire lists with an ease offered by no other language I know of. And look what you can do with recursion! Now three companies have combined the power of Logo with the 16-bit microprocessing of the PC to give Logo programmers the true feeling of 'no threshold and no ceiling.'

"I believe the first company to announce Logo for the IBM PC was Digital Research, Inc. of Pacific Grove, California. They are best known for the development of the CP/M operating system. But now they have come out with an excellent version of

Logo. Of course, it uses CP/M-86 as its operating system, rather than PC/DOS as used by the other two versions.

"The second company to announce Logo for the IBM PC was Harvard Associates of Cambridge, Massachusetts. These people have worked with Logo for quite some time. They're the same people who offer the Tasman Turtle."

"Now, Whatson, you know how I detest those pesky animals running all over the place."

"This isn't an animal, 'olmes. It is a small robot that moves around the floor on command from the computer. It just happens to look like a turtle.

"The third company offering Logo for the IBM PC is Waterloo Microsystems Inc., of Waterloo, Ontario, Canada. Development work has been done at the University of Waterloo. Waterloo Microsystems Inc. also offers a unique operating system and software development system called the Waterloo Port™."

"Whatson, good fellow, we all know Logo is Logo. And the PC is the PC. So why bother explaining what everyone already knows?"

"Some day, 'olmes, I hope you learn more to do with your head than just look through that silly glass. Think, man, think!"

* * *

Dr. Logo and Waterloo Logo are both extensions of Apple Logo. Both use many of the same primitive commands plus a number of new ones which are very useful in editing and debugging procedures and in managing the workspace. PC Logo is a bit closer to the MIT versions of Logo.

Undoubtedly the most significant difference between these versions and those available on other computers is the amount of free workspace available. PC Logo requires only 64K of memory but will address 192K. Dr. Logo and Waterloo Logo both require 128K of memory—however, Dr. Logo can address a megabyte of memory (that's one million bytes!) through a hard-disk system and 192K through floppy disks. That's many times the workspace available in other systems. That also means that truly useful programs can now be written in Logo without running out of space. Interactive business graphic procedures, statistical analysis using recursive procedures and data base management systems using list manipulation are among the possibilities.

To help the user take advantage of this workspace, all three versions offer extensive editing commands. Dr. Logo offers a "Help" function which is particularly useful. When heavily involved in a complex procedure, users can ask for help and have each Dr. Logo primitive listed and defined with a short example of its use. In addition, users can also change the order of procedures, cross-reference procedures and add and delete comments. Another nice feature added to Dr. Logo is the TRACE command. This is particularly useful in debugging complex procedures.

Dr. Logo and Waterloo Logo both allow the use of two displays, a monochrome display for text and a color monitor for graphics. Of course, both the monochrome and color graphics adaptors have to be in the PC to use this feature. A color graphics adaptor is required to display graphics using PC Logo.

The graphics resolution of all three versions is superior. In certain versions of Logo, it is difficult to ascertain the true heading of the turtle. In some, the turtle only turns in 15-degree increments. (For example, entering the command RIGHT 5 would show no change in the turtle's heading.) All three of these packages are quite precise, especially when connected to an RGB monitor. There is little difference using a composite monitor, however.

All three versions incorporate a FENCE command. Dr. Logo and PC Logo give you the choice of allowing the Turtle to stop at the fence, to go through the WINDOW or to WRAP. Waterloo Logo does not allow wrapping. Unless the WINDOW command has been entered, a message will appear stating, "The turtle cannot go past the fence." Some will criticize the lack of the wrap feature on the grounds that this gives children the opportunity to experience the effects of big numbers. Using the FENCE, however, channels their thinking more into procedural development on the screen, using numbers, distances and visual references which have a well-defined relationship to each other. However, I would still prefer to have the choice.

Two other features of Waterloo Logo are particularly useful, especially to young programmers. PAINT fills in an area in the same manner as do the FILL procedures which have appeared in YPLA's *Turtle News.* A SOUND command generates music. The documentation presumes you know the frequencies for each tone, however. Dr. Logo includes a TONE command for music and sound effects, which is considerably easier to use.

All three versions have commands for printing graphics and procedures. PC Logo and Dr. Logo both have commands for using light pens, joysticks and paddles. The PC Logo reference manual includes a section on accessing other peripherals, including modems.

PC Logo is priced at $199.95 and comes with a reference manual and tutorial guide which are very well-written and quite easy to understand. And, like MIT Logo, the package includes both language and utility disk. A back-up provision is included so that the language disk can be copied once. The language runs with DOS 1.0 or greater. However, DOS 2.0 is required to print out graphics.

Dr. Logo includes a comprehensive reference manual and tutorial along with two copies of the language disk. Hank Ketcham, the creator of cartoon character Dennis the Menace, has created a cartoon

INTRODUCING
THE BASICS OF BASIC
The easiest way to learn Programming for People of any age

For years, schools have been depending on us to help them create great lesson plans with our computer software and A.V. materials. Now we've developed a series of lessons which makes learning BASIC programming easy and reduces your planning time. No need for your students to struggle through difficult manuals. With our programs, all student learning is done right from the computer and you. We also include an extensive Teacher's Lesson Planner to assist you.

For about $8 a lesson, when you purchase this series of 12 lessons on 4 disks, your students learn:

• Programming in Direct Mode • Strings and Numeric Constants • Input and Conditional Statements • Formatting the Output • Looping and Branching • Mathematical Functions, Arrays • Error Messages and Disk I/O • And much more

AND BEST OF ALL, WITH OUR NO RISK GUARANTEE, HOW CAN YOU LOSE?

Only $99
Includes all 12 programs, 4 disks, free back-ups and Teacher's Lesson Planner.

Order Code #CC4001A (Apple II/IIe)
#CC4001I (IBM-PC)
(Available soon Commodore 64, TRS-80)

Call or write:

FOCUS

FOCUS MEDIA, INC.,
839 Stewart Ave., Garden City, NY 11530
800-645-8989 Except NY
516-794-8900 Call Collect

Focus Media Inc. 839 Stewart Ave., Garden City, NY 11530
Please send me ☐ Apple II/IIe — CC4001A ☐ IBM-PC — CC4001I
☐ Enclosed is a check for $99 (Save shipping & handling).
☐ Please bill my school (add 4% for shipping & handling)

Name ______________ Title ______________
School ______________
Address ______________
City ______________
Mastercard # ______________ Visa # ______________
Expir. Date ______________

character, Dr. Logo. This character explains Dr. Logo in a special Dr. Logo tutorial. The package is specially priced at $99.95 throughout 1983 and is $149.95 starting January 1, 1984.

Waterloo Logo comes with a 100-page tutorial and reference guide. The list price is $180 with quantity discounts available. Some people may be concerned because of the lack of documentation. However, with all of the material available for Apple Logo, Waterloo Logo users should have no problem learning the language. The original version would not run on the new PC models or on the XT. Reportedly, this is being corrected.

* * *

"Whatson, you have really whetted my appetite. Tell me more about Logo for the IBM PC!"

"Unfortunately, 'olmes, this is all we have time for now. I know we have just scratched the surface. We haven't even talked about the IBM Logo package. But all three of these we have discussed have a lot to offer. It depends on what individual users want, what features they seek and where, how and who will use it. All three have good features with distinct advantages. And just like the many other versions of Logo, each has some features missing. There is no perfect package yet."

"Well! If you're not going to tell the whole story, why bother?"

"Because, dear 'olmes, people need to know they have more choices. Let them go and look at these packages. Let them see that now, through the processing power of the IBM PC, there is Logo to challenge even the most active imagination; no threshold and no ceiling—really!"

END■

[James H. Muller, Young Peoples' Logo Association, 1208 Hillsdale Dr., Richardson, TX 75081.]

Bulk Mini-subscription

Educators running inservice or preservice courses for educators can purchase bulk mini-subscriptions to **The Computing Teacher.** A mini-subscription is four issues at a cost of $5 within the U.S. or $7 outside the U.S. **An order must consist of a minimum of 15 mini-subscriptions to one address.** The order must include information on who is teaching the course, the nature of the course, when the course starts and when it ends.

Place orders with Inservice Department, ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403. (Save $2.50 by sending payment with your order.)

COMPUTERS IN THE ARTS AND HUMANITIES

Linda F. Ettinger

*[Editor's Note: Art teachers are frequently asked to recommend microcomputer hardware and software for use in art teaching and computer graphics. And if they aren't asked, many art teachers are curious about potential microcomputer applications in art teaching and would like to participate in the purchasing of microcomputer equipment at all school levels. In this column, high school teacher Richard Adams presents a frank comparative analysis of three currently available computer graphics packages—*Graphics Tablet, VersaWriter *and the* KoalaPad. *Adams' insight into the practical applications and considerations of these packages will be helpful to art teachers faced with the task of selecting appropriate computer graphics packages for schools.]*

Low-Cost Graphics Tablets

by
Richard C. Adams

One of the reasons many of us bought an Apple computer when we did was the idea of doing high resolution color graphics.

We got our machines and found that we could indeed plot from point to point with the HPLOT command and use seven colors, two of which were black and two white. We soon found that HPLOTting was satisfactory for plotting mathematical functions and abstract designs but really wasn't all that neat for real free-hand artwork. It took so much *planning!* We then learned to hook up two game paddles (while they still worked) and practiced the fine art of Etch-A-Sketch™ on the screen. We got a joystick and that helped, but it still left something to be desired.

Then we heard about the **Graphics Tablet™**[1] that Apple Computer developed a number of years ago. It had a real stylus that you could use on the surface of the tablet, and you could draw in any of the colors. Not only that, you could calibrate it to find circumferences and areas of closed figures—that was something you could tell the math department about! However, it had two main problems: First, if you set a disk down on the tablet surface with its buried, electrified wires, the disk you picked back up was blank. Second, it costs $795.00!

I have found two capable and less expensive alternatives to the *Graphics Tablet*™. They have both been reliable and have suited the needs of students in art, biology and physics. Science and geography teachers are also using them.

The **VersaWriter**™[2] is one of those amazingly simple "why didn't I think of that?" designs. Basically, it is built like a human arm. There is a 12x14 sheet of white plastic with a "shoulder joint" at the top and an "elbow" joint in the middle of the arm. A clear plastic disk with a spot in the center is where the "hand" would be. There are variable resistors (potentiometers) in the shoulder and elbow. Any place on the surface has a unique pair of resistances that the Apple or Atari or Atari game paddle port can read.

That means that you can sketch something, or take a picture you find and lay it down on the plastic surface and go over the lines with the disk. As you draw, the lines appear on the screen. You can scale the drawing up or down from one-quarter size to four times the size as you draw (the x and y axes are scaled separately so you can distort the image if you wish), move the drawing later to anywhere on the screen or rotate it. You can "draw" in any of six "brush" sizes. A "microscope" mode is available for changing your drawing by individual pixels.

Then it is possible to fill in the open areas in any of 212 colors. That's right, **212 colors!** *VersaWriter* software does this by mixing those same seven colors. You can have, for example, an orange from the palest sand to pure orange and even dark orange with pixel-by-pixel mixing with black. Then all 7 of the primary colors can be mixed. The result is 212 colors for filling. The filling routine is not the best; you sometimes have to fill smaller sections with a second keypress. You can label any part of your picture in several alphabets, left to right, right to left, up, down, diagonally, or however you want.

For your math friends, there are still the functions of scaling distances and finding areas of closed portions.

What really got me, however, was the shape table-making part. If you've ever made up a shape table by hand, you know mental discipline. Several programs are written that enable you to "draw" something in lo-resolution and then have it converted to a shape, but you still have to worry about the vectors you use to draw it with. With the *VersaWriter*, you take a shape or part of a picture that you like, put a mark at the upper left-hand corner and another one at the lower right-hand corner, press a key and you have the shape all done. It will even preserve the *color(s)* that you used for the drawing!

Quite a number of shape tables are included on the disks that come with it, including architectural symbols, plumbing, electrical symbols, electronics, landscape architecture, chemical symbols and chemical apparatus, so you can use those for Computer Aided Design.

The *VersaWriter*'s mode of input is especially designed for people like me who can't really draw freehand, but who are good at making drawings with a ruler, compass and maybe parts of drawings that someone else did. The software with the *Versa Writer* allows you to manipulate drawings of any kind and make them much more useful.

If you still want to have the pencil in your hand to do freehand work, I offer my other favorite device, the **KoalaPad Touch Tablet.**[3]

I know, the pictures in the magazines show the cute koala and you're sure it can't *really* be anything a grown-up could stand to use without biting on a lemon first.

The *KoalaPad*™ *does work*, and I like it a lot. The pad itself has a writing surface which is 4.25 inches square. Two bars at the top signal drawing modes to the computer. If you have an Apple, the drawing surface corresponds to PDL(0) and PDL(1), with the bars on top being the paddle buttons. For the IBM, the surface is Stick(0) and Stick(1); for the Commodore 64, the surface is Peek(54297) and Peek (54298); VIC-20 is Peek(36872) and Peek(36873); for the Atari Paddle(0), Paddle(1), Stick(0), Strig(0).

Micro-Illustrator™ is the program included that does the drawing. You can use your finger or the stylus that is included. You have your choice of 15 actual colors (not counting the duplicate white, black and gray), 8 "brush" sizes and 13 drawing modes on the menu which are chosen by moving the cursor to that box and pressing the bar.

The *Draw* mode works by holding down the bar and drawing with the stylus. *Point* and *Line* are easy to understand. *Line* allows continuous point to point straight lines to be connected. *Ray* mode will draw lines from the same point to any other place. *Fill* mode uses a very fast and complete filling routine. *Frame* and *Box* allow you to draw either open rectangles or filled rectangles, with *Circle* and *Disc* accomplishing the same thing for circles. *Erase* will clear a screen to any color. *Normal* and *Magnify* allow you to see the whole picture or a pixel-by-pixel form. *Storage* will save a picture on a disk, and since they're stored in a standard format, they're accessible to most graphics utilities software. There is also a limited *Help* option.

Does the *KoalaPad* package allow circumference and area? No. Can you do shape tables? No. Can you rotate, scale or move the pictures? No. Can you label the pictures with an alphabet? No. *KoalaPad Touch Tablet* does not do all the things that the *Versa Writer* does, but it also costs considerably less.

I own both of them because they complement each other well. For freehand drawing and things using boxes, discs, circles or squares, I'll use the *KoalaPad*. For copying drawings from a sheet of paper, making shape tables, and more elaborate manipulations of drawings already made (even those made with the *KoalaPad*), I'll use the *VersaWriter*.

With the two together, you have many more functions than the *Graphics Tablet* with routines that are easier to use, and the cost is still about half.

On the other hand, you lose the capability of wiping out a disk just by setting it down on the tablet.

END■

[Richard C. Adams, Pleasant Hill High School, Pleasant Hill, OR 97455.]

[Linda F. Ettinger, Dept. of Art Education, Univ. of Oregon, Eugene, OR 97403.]

FOOTNOTES

1. *Graphics Tablet* from Apple Computer Inc. and Apple dealers. Apple II+, recently for IIe. $795. Apple is introducing a revised version of this product.
2. *VersaWriter* from Versa Computing, Inc., 3541 Old Conejo Rd., Suite 104, Newbury Park, CA 91320. (805) 498-1956. Apple II, II+ or IIe, Atari 800. $299.00.
3. *KoalaPad Touch Tablet* from Koala Technologies Corporation, 4962 El Camino Real, Los Altos, CA 94022, (800) 227-6703; (800) 632-7979 (in Calif). Apple II, II+, IIe; Atari 400, 800; IBM PC; Commodore 64; VIC-20. Prices range from $99.95 to $124.95.

Educational Equity and Excellence for a Nation at Risk: Programs That Work

June 20-22, 1984 Kalispell, Montana

The central theme of this third annual conference is around programs that work in schools and the community and business world in the areas of race, sex, national origin, disadvantaged and handicapped, as well as presentations which demonstrate effective cooperation between education and the business world.

For further information, contact Raymond E. Wong, Educational Equity Specialist, Mountain West Desegregation Assistance Centers for Race and Sex, Weber State College, Ogden, UT 84408; or contact Judy Johnson, Assistant Supt., Office of Public Instruction, Capitol Bldg., Room 106, Helena, MT 59620.

Logo in the Classroom Session 1

[Editor's Note: This article is an excerpt from Logo in the Classroom, *a new ICCE publication written by Shirley Torgerson, Mary Kay Kriley and Janet Stone. Twenty-three sessions introduce students to Logo in the context of the elementary curriculum. The 8½x11 packet contains detailed information on classroom management techniques for getting started and facilitating student use of limited computer resources. Six dialects of Logo are included which will enable teachers to adapt other Logo materials to the dialect they are using. Different student worksheets are included in the packet for each version.]*

*"The child cannot . . . reconstruct a perspective point of view until he has himself occupied the position to which it corresponds. This is because thought can only replace action on the basis of the data which action itself provides."**

Jean Piaget and Barbel Inhelder
The Child's Concept of Space

OBJECTIVE: To introduce students to the Logo primitives **FOWARD, BACK, RIGHT** and **LEFT** through self-involvement in spatial relationship activities.

CONCEPTS/SKILLS: An understanding of commands as directed movement in space using self as a point of reference and conceptualization of commands for future identification with the turtle.

FACILITIES/MATERIALS: Overhead, chalkboard or tagboard strips labeled FORWARD, BACK, RIGHT, LEFT; playground (gym, hallway, etc.); 5x8 card, envelope and sheet of paper for each group of 4-5 students; student copies of problem sheet.

PREPARATION: Make 5x8 cards with directions to envelopes located on the playground (gym, hallway, etc.). Number corresponding cards and envelopes. For each envelope, prepare a sheet of paper with an assignment to write directions to another specified location. (Sample 5x8 card and assignment sheet follow the Activity Section.)

PRIMITIVE CHART:

	APPLE	ATARI	MIT	TI	TRS-COLOR
Logo Version	Logo Computer Systems, Inc.	Atari, Inc.	Terrapin & Krell	Texas Instruments	Tandy Radio Shack
Computer	Apple II Apple IIe	Atari 400 Atari 800	Apple II Apple IIe Commodore 64	TI 99/4A	Radio Shack Color Computer
Primitives	FORWARD BACK RIGHT LEFT	FORWARD BACK RIGHT LEFT	FORWARD BACK RIGHT LEFT	FORWARD BACK RIGHT LEFT	FORWARD BACK RIGHT LEFT

COMMENTS: One of the central ideas in using Logo to explore geometry is that the turtle uses itself as a point of reference as opposed to using a Cartesian coordinate system. Students' actual experience in walking the commands will help them later in identifying with turtle movements on the screen, especially as the graphic displays become complex. The younger the student, the more essential the self-involvement in spatial activities. Some variations of this activity are listed at the end of this session—however, since debugging a procedure is often facilitated by knowing how to "play turtle," some walking of the commands is recommended.

The commands RIGHT and LEFT only change direction in place; there is no movement forward or backward. It may be necessary to call attention to this.

Students usually turn 90° when asked to turn RIGHT or LEFT. Make sure they use a 90° turn when following RIGHT or LEFT commands, but do not call attention to it unnecessarily at this time.

PRELIMINARY ACTIVITY:

Print the words FORWARD, BACK, RIGHT, LEFT on the overhead or chalkboard, or use labeled tagboard strips. Discuss the meanings with students.

*Reprinted by permission of Humanities Press Inc., Atlantic Highlands, NJ 07716.

ACTIVITY:

Part 1

Ask the students to stand, move FORWARD 2 steps, BACK 1 step, turn RIGHT (emphasize that there is no forward or backward movement), FORWARD 1 step, BACK 2 steps, turn LEFT, then FORWARD 2 steps, etc.

Explain to the students that they can move from one place to any other by using only these four commands. Ask one student to stand. Give commands to direct that student about the room to various locations. Then let a student give the commands for moving another student to a specific location.

Students also need to see the written commands. Write on chalkboard, calling attention to the single line, vertical listing format; use of capital letters; and the END command:

```
FORWARD 6
RIGHT
FORWARD 3
RIGHT
BACK 1
RIGHT
FORWARD 3
LEFT
FORWARD 6
END
```

Have a student walk the directions. When students demonstrate an understanding of how the directions are written and followed, give the assignment below:

Part 2

Have each student bring a pencil, and then form them into groups of four or five. Number the groups. Tell them that each group will be given a 5x8 card with directions to a different location on the playground (or gym, hallway, etc.). If they follow the directions correctly, they will find there an envelope with their group number on it. Inside the envelope they will find an assignment asking them to write a set of directions to a specified location.

Pass out each group's 5x8 card. Make students aware that the directions on the card are given using only FORWARD, BACK, RIGHT, LEFT.

Since each person's steps are different in length, tell the students just to walk normally, and they will come fairly close to their envelope. When they find it, they should open it, read it and complete the assignment. Suggest that when they write the commands FORWARD, BACK, LEFT or RIGHT, they should use a new line for each command, similar to the directions given on the 5x8 card.

Variations for Part 2:

- Use a hundreds chart. Play addition or subtraction games. Example: Begin at 40, heading north.

```
FORWARD 3 SPACES    (add the number in this
                                block to 40)
LEFT
BACK 6 SPACES       (add to previous total)
RIGHT
FORWARD 1 SPACE     (subtract the number
                           in this block)
END
```

What is the total?

- Secure a supply of city maps. Have students begin at a specified point and follow directions of FORWARD 3 BLOCKS, RIGHT, FORWARD 6 BLOCKS, LEFT, etc. Students then identify the intersection by street names. They, in turn, can write similar directions for returning to the original position. (This activity would correlate well with a social studies and map skills unit.)

SAMPLE 5x8 CARD:

Group 1:
Start at tetherball pole facing the school.

```
FORWARD 20
LEFT
FORWARD 15
RIGHT
BACK 8
RIGHT
FORWARD 20
LEFT
FORWARD 8
END
```

Remember, each 5x8 card should include the group number and directions to a different location in the same general area.

SAMPLE ASSIGNMENT SHEET:

Complete the directions below using only the four commands FORWARD, BACK, RIGHT and LEFT.

Start at the point where you found your group's envelope, and go to the bike rack.

Begin by facing north. Tell how many steps you take when moving FORWARD or BACK. Remember to write only one command on each line.

Write END when your directions are complete.
Sign your names at the bottom of this sheet.

DIRECTIONS TO BIKE RACK

FORWARD________________

____

____

____

____

____

ASSIGNMENT: Have students complete the problem sheet for this session.

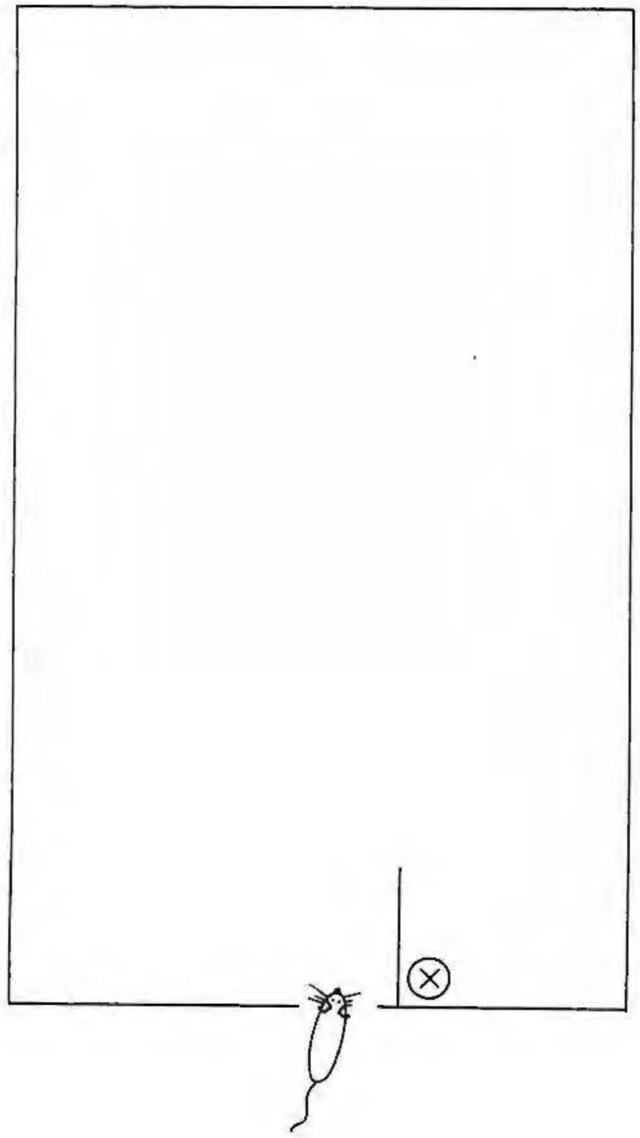
PROBLEM SHEET

"Will Hungry Mouse Reach the Food?"

Below is a 9 cm x 15 cm room. Just inside the door and immediately to the right is a wall extending 2 cm into the room. Behind this wall is a dish of food shown by the circled X. You are a hungry mouse. You enter the room at the door. You can smell the food but have a problem. **You can only move FORWARD and can only turn LEFT.**

Draw your path to the food. Use a ruler to make straight lines and the end of your ruler to make LEFT turns.

If you succeed in reaching the food, try sending a message to your friend (?), HUNGRY CAT, who also has the problem of only going FORWARD and making LEFT turns. If your directions are correct, HUNGRY CAT can finish the food. If your directions are *not* correct, HUNGRY CAT will not find the food—but might find *you!*



Dear Hungry Cat,

FORWARD ________ cm
LEFT
FORWARD ________ cm
LEFT
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
END

Your friend,

Well-Fed Mouse

END■

[Ed. note: This problem sheet has been reduced for reprinting in The Computing Teacher. *In* Logo in the Classroom, *all student pages (problem, practice and challenger sheets) are easy-to-read, full pages designed for duplication and use by elementary students.* Logo in the Classroom *is available from ICCE for $11.00.]*

THE LOGO CENTER

Kathleen Martin • Tim Riordon

Joan Newman and Andrew Berner, Contributors

Joan Newman, a participant in a recent Logo workshop, became excited at the possibilities for using Logo. Joan's comments serve as an introduction to this month's column.

> My interest has been in those things which can be done with Logo besides turtle graphics. . . . I was able to put together a rough program to stimulate the writing of poetry, which upper elementary students could probably develop a good deal further given the equipment, Logo software and a little encouragement.
>
> All in all, it seems to me that Logo is the ultimate general literacy program. . . . Our class explored uses of Logo which could give students an idea of how most computer programs work in areas such as word processing, the design of target and word games, geometric design and graphics, interactive programming, robotics (remote manipulation of an object to make it do things), some simulations and even how a "mouse" could be programmed to work. The most compelling use of Logo, I think, still is in teaching users to think logically, organize and control processes and write concise, effective procedures. Terrific!

Joan Newman, a former teacher, now serves as a Program Administrator for Learning Resources in the Office of the Superintendent of Public Instruction in the state of Washington. We wish to present Joan's "rough program" here—it's called POEMPRINT. Our other offerings this month are PIGLATIN, by Andrew Berner, University of Dallas, and MADLIBS. All these projects continue with a theme we began last month—non-graphic Logo applications. Next month, our focus will be on Logo and numbers. And now, Joan Newman's POEMPRINT (written in Apple Logo).

POEMPRINT

The superprocedure here is called POEMPRINT. First it calls upon a subprocedure POETIC.WORDS which creates and stores lists of adjectives, nouns, etc. These words will be used by a later subprocedure to generate the random lines of poetry. Next, POEMPRINT presents some instructions and then calls the subprocedure POEM. This procedure, POEM, does a lot of work. It presents a line of random poetry and then asks the user to type in an improved version of that line. The list of words that the user types in is stored as a variable :EDITEDLINE1. The subprocedure POEM does the process of presenting a random line and then storing the user's edited line three times. When POEM has finished its job of collecting three edited lines of poetry, control goes back to the superprocedure, POEMPRINT. This procedure continues by clearing off the screen and then prints the three edited lines of poetry.

```
TO POEMPRINT
  POETIC.WORDS          (subprocedure that creates lists
                         of adjectives, articles, nouns,
                         prepositions, verbs and adverbs)

  PRINT [ WELCOME TO POEMPRINT ]
  PRINT [ ]
  PRINT [ YOU WILL BE SHOWN 3 LINES OF
    "POETRY" GENERATED RANDOMLY BY
    THE COMPUTER ONE AT A TIME. ]
  PRINT [ ]
  PRINT [ YOU WILL HAVE A CHANCE TO
    EDIT EACH LINE. ]
  PRINT [ ]
  PRINT [ AT THE END YOUR "POEM" WILL
    BE PRINTED OUT FOR YOU. ]
  PRINT [ ]
    POEM 1              (subprocedure that generates
                         lines and collects and stores
                         the 3 edited lines of poetry)

  CLEARTEXT
  PRINT :EDITEDLINE1
  PRINT :EDITEDLINE2
  PRINT :EDITEDLINE3
  PRINT [ ]
  PRINT [ ]
  PRINT [ SO, HOW DO YOU LIKE YOUR
    POEM? ]
  PRINT [ ]
  PRINT [ ]
  PRINT [ TYPE IN THE WORD POEMPRINT
    IF YOU WANT TO PLAY AGAIN. ]
END
```

The first subprocedure is POETIC.WORDS. It stores lists of words which will be used later by the POEM subprocedure. The words are stored in lists which are associated with names that tell the type of word: "ARTICLE, "ADJECTIVE, "NOUN, etc.

```
TO POETIC.WORDS
  MAKE "ADJECTIVE [ BOWING SWAYING
    SHINING GREEN SIGHING LEAFY TALL
    BENDING DANCING FLOWING DELICATE ]
  MAKE "ARTICLE [ THE A EACH THOSE
    THESE AN THAT THIS ]
```

(Note to English purists: we know we have included demonstrative pronouns!)

```
  MAKE "NOUN [ TREE LEAF BRANCH EARTH
    WIND SKY AIR BARK BIRD DOVE WINGS
    EYES CLOUDS ]
  MAKE "PREPOSITION [ OVER UNDER
    AROUND THROUGH ]
  MAKE "VERB [ MOVES IS [ WOULD BE ]
    LIVES FLY GROWS RISES ]
  MAKE "ADVERB [ GENTLY SLOWLY
    GRACEFULLY QUICKLY QUIETLY ]
END
```

Next is the workhorse subprocedure, POEM. This procedure generates the line of random poetry by printing a sentence which follows a "formula." The formula is Article Adjective Noun Verb Adverb Preposition Adjective Noun. A line of poetry is generated by picking at random from the list named "ARTICLE, by picking at random from the list named "ADJECTIVE, by picking at random from the list named "NOUN, etc. (The PICKRANDOM procedure is given below.) After POEM presents a line of poem, it asks the user to improve the line. A direction is given: "TYPE IN YOUR EDITED SENTENCE AND PRESS RETURN. . . ." The POEM procedure then creates a variable with names like "EDITEDLINE1, "EDITEDLINE2, "EDITEDLINE3 and stores whatever the user types in as the value of the variable. This is accomplished by the line MAKE WORD "EDITEDLINE :COUNTER READLIST. POEM is a recursive procedure. It is called three times before it stops. Thus, it generates 3 lines of poetry and collects 3 edited lines from the user.

```
TO POEM :COUNTER
  IF :COUNTER > 3 [ STOP ]
  PRINT [ SENTENCE PICKRANDOM :ARTICLE
    PICKRANDOM :ADJECTIVE PICKRANDOM
    :NOUN PICKRANDOM :VERB PICKRANDOM
    :ADVERB PICKRANDOM :PREPOSITION
    PICKRANDOM :ADJECTIVE PICKRANDOM
    :NOUN ]
  PRINT [ ]
  PRINT [ ]
  PRINT [ HOW WOULD YOU EDIT THIS LINE
    TO MAKE IT SOUND LIKE A LINE OF
    POETRY? ]
  PRINT [ ]
  PRINT [ TYPE IN YOUR EDITED LINE AND
    PRESS RETURN WHEN YOU ARE
    FINISHED. ]

  MAKE WORD "EDITEDLINE :COUNTER
    READLIST

  CLEARTEXT
  PRINT [ ]
  PRINT [ ]
  POEM :COUNTER + 1
END
```

The line MAKE WORD "EDITEDLINE :COUNTER READLIST deserves some further comment. The Logo primitive WORD takes 2 inputs and binds them together and outputs the result. For example, the result of WORD "EDITEDLINE 17 would be EDITEDLINE17. The result of WORD "EDITEDLINE 1 is EDITEDLINE1. In the POEM subprocedure, the input :COUNTER begins with 1, then becomes 2, then 3 and finally 4. The purpose of the line in question

```
MAKE WORD "EDITEDLINE :COUNTER READLIST
```

is to store 3 edited lines of poetry. The device of using the phrase WORD "EDITEDLINE :COUNTER allows 3 different variables to be automatically created. The 3 variables are: EDITEDLINE1, EDITEDLINE2 and EDITEDLINE3. This device was used in last month's column.

Finally, here is the PICKRANDOM subprocedure, which has a list of words as its input.

```
TO PICKRANDOM :WORDLIST
  MAKE "NUMBER RANDOM COUNT :WORDLIST
  OUTPUT PICK :NUMBER + 1 :WORDLIST
END
```

How could this project be developed further? A variable that controls how many lines the final poem will have could be added. As the procedures are written now, they always produce 3-line poems. Or how about a fancier version of the POETIC.WORDS subprocedure? How about one which presents a poetry topic and then prompts the user to give 10 nouns related to the topic, 10 verbs related to the topic, 10 adverbs related to the topic, 10 adjectives related to the topic, 10 participles related to the topic. Or how about different sentence formulas to be the basis of the poetic lines? Or maybe a routine that asks if the user wants a printed copy of the poem? These modifications might lead to a pretty nice instructional tool.

A Return to Piglatin

If you are among those who bemoan the loss of Piglatin in the lives of our children, then you will delight in this program by Dr. Andrew Berner of the University of Dallas. Dr. Berner is a teacher of mathematics who has spent considerable time working with children on Logo. His Piglatin program utilizes Logo's list processing capabilities. It's not a program for beginners, but one that your students will enjoy after they have gained some experience in list processing.

Before rushing to the computer, it would be well to work through an off-computer activity with your students. It is an activity that will familiarize them with Piglatin (or at least with this modified version of Piglatin), and encourage them to think about the steps involved in changing from English to Piglatin.

Divide your class into groups of three or four students. Give each group a set of 3x5 cards with a different word on each card. Be sure that the words represent various parts of speech. Also provide a separate set of cards with "way" and "ay" written on them. Instruct the students to arrange some of the cards into a sentence such as this one:

I love to eat pink and blue flowers.

Give the students the rules for converting the sentence to Piglatin. They should examine the words in the sentence one at a time. If the word being examined begins with a vowel (A E I O U) the students should add "way" to the word. Otherwise they should cut the first letter from the front of the word and attach it to the end. Then they should add "ay" to the word. The sentence above would then read:

Iway ovelay otay eatway inkpay andway luebay lowersfay

Now let's look at Dr. Berner's Logo program that will convert an English sentence to this version of Piglatin.

```
TO PIGLATIN
 MAKE "SEN READLINE
 PRINT LATIN :SEN
END

TO LATIN :SEN
 IF :SEN = [] OUTPUT [] ELSE
 OUTPUT FPUT PIG FIRST :SEN LATIN BUTFIRST :SEN
END

 TO PIG :W
  IF CHECKFIRST :W [A E I O U] OUTPUT WORD :W "WAY ELSE OUTPUT
  WORD WORD BUTFIRST :W FIRST :W "AY
END

TO CHECKFIRST :W :L
 IF :L = [] OUTPUT "FALSE STOP
 IF FIRST :W = FIRST :L OUTPUT "TRUE STOP
 OUTPUT CHECKFIRST :W BUTFIRST :L
END
```

PIGLATIN is a procedure that waits for the user to type in a sentence (MAKE "SEN READLINE) and then prints whatever the procedure LATIN outputs (PRINT LATIN :SEN). LATIN is a recursive procedure that examines each of the words in the given sentence. It then outputs the sentence that results from the action of the procedure PIG on each word (OUTPUT FPUT PIG FIRST :SEN LATIN BUTFIRST :SEN). FPUT is a function that attaches an item (PIG FIRST :SEN) to a list (LATIN BUTFIRST :SEN).

PIG is a procedure that checks a word to see if it begins with a vowel. If it is a vowel, PIG outputs the word with "way" attached to the end (IF CHECKFIRST :W [A E I O U] OUTPUT WORD :W "WAY). If the word is not a vowel, PIG outputs the word without the first letter and then adds the first letter plus "AY" to the end of the word (ELSE OUTPUT WORD WORD BUTFIRST :W FIRST :W "AY).

CHECKFIRST is a recursive procedure that checks if a word begins with a letter on the list and is called with :L, which is the list of vowels. If the first letter of the word is not a vowel, the CHECKFIRST procedure stops (IF :L = [ ] OUTPUT "PLEASE STOP, where L is the list of vowels). CHECKFIRST also stops when the first letter of the word being examined is found to be the same as the vowel being examined in the list (IF FIRST :W=FIRST :L OUTPUT "TRUE STOP). Otherwise the procedure continues checking the first letter of the word against the next vowel in the list (OUTPUT CHECKFIRST :W BUTFIRST :L).

You may be familiar with another form of Piglatin where the consonants preceding the first vowel in the word are *all* put at the end of the word and then "ay" added. If so, you might encourage your students to modify Dr. Berner's program accordingly. Or, better yet, you might challenge them to design their own language and then write a program that translates English into their language. If they choose to do the latter and it works, please share the program with us.

MADLIBS

Mad Libs are one of the few times students find a reason to converse using words like "noun," "past tense verb," "adverb," etc. The examples used in this project come from "Popeye Mad Libs," published by Price/Stern/Sloan Publishers Inc., 410 North La Cienega Blvd., Los Angeles, CA 90048.

This is a project that can start simply. The first version will use the computer as a helper in the process. The superprocedure that runs the show is MADLIB1. It has three jobs: to present instructions; collect the words that the player types in; and present the Mad Lib using the words supplied by the player. Here are the 6 procedures for the first version of MADLIBS:

Comment

```
TO MADLIB1
  CLEARTEXT
  ERASE NAMES
  INSTRUCTIONS
  CLEARTEXT
  COLLECT
  PRESENT.MADLIB :WORDLIST
END

TO INSTRUCTIONS
  PRINT [ YOUR FRIEND AND I ARE GOING ]
  PRINT [ TO PLAY MADLIBS WITH YOU. I ]
  PRINT [ WILL ASK YOU TO TYPE IN SOME ]
  PRINT [ WORDS. YOUR FRIEND WILL TELL ]
  PRINT [ WHAT KIND OF WORDS ]
  PRINT [ WE WANT. ]
  PRINT ] ]
  PRINT [ PRESS A KEY TO GO ON ]
  MAKE "JUNK RC
END

TO COLLECT
  CLEARTEXT
  MAKE "WORDLIST [ ]
  PRINT [ HOW MANY WORDS ARE THERE
    GOING TO BE? ]
  MAKE "NUMBER FIRST REQUEST
  COLLECT.WORDS 1 :NUMBER
  PRINT [ ]
  PRINT [ PRESS A KEY TO SEE YOUR
    MADLIB. ]
  PRINT [ IF THE CURSOR BLINKS AND YOU ]
  PRINT [ WANT TO READ MORE, PRESS ]
  PRINT [ A KEY TO GO ON. ]
  MAKE "JUNK RC
END
```

:WORDLIST will be all the words that will be substituted in the Mad Lib.

```
TO COLLECT.WORDS :COUNT :NUMBER
  IF :COUNT > :NUMBER PRINT [ THAT'S ALL
    WE NEED ] STOP
  PRINT1 SENTENCE [ TYPE IN WORD # ]
    :COUNT
  MAKE "WORDLIST LPUT REQUEST
    WORDLIST
  PRINT [ ]
  COLLECT.WORDS :COUNT + 1 :NUMBER
END
```

This procedure runs enough copies of itself to collect all the words the player is going to supply. Notice how the variables :COUNT and :NUMBER control this.

The last two procedures present the Mad Lib. SELECT takes a word from the word list that was collected earlier and makes it available to be substituted into the Mad Lib sentence. The presentation has been broken into 3 parts to make the reading easier.

```
TO SELECT :N :WORDLIST
  IF :N = 1 OUTPUT FIRST :WORDLIST
  OUTPUT SELECT :N - 1 BUTFIRST :WORDLIST
END
TO PRESENT.MADLIB :WORDLIST
  CLEARTEXT
  PRINT [ POPEYE'S ENTRANCE ]
  PRINT [ ] PRINT [ ]
  PRINT ( SENTENCE [ SEVERAL YEARS AGO,
    CASTOR OYL WANTED TO SAIL TO ]
    SELECT 1 :WORDLIST [ TO GAMBLE AT
    THE ] SELECT 2 :WORDLIST [ CASINO. ]
    [ HE BOUGHT AN OCEAN-GOING ] SELECT
    3 :WORDLIST ", [ THEN HIRED POPEYE,
    A/AN ] SELECT 4 :WORDLIST [ SAILOR, ]
    [ TO BE THE SHIP'S ] SELECT 5
    :WORDLIST ". )
  PRINT [ ] PRINT [ ] MAKE "JUNK RC
  PRINT ( SENTENCE [ OLIVE OYL, WHO WAS
    CASTOR OYL'S ] SELECT 6 :WORDLIST ",
    [ STOWED AWAY IN THE SHIP'S ] SELECT
    7 :WORDLIST ", [ COOKING ] SELECT 8
    :WORDLIST [ AND WASHING ] SELECT 9
    :WORDLIST ". )
  PRINT [ ] MAKE "JUNK RC
  PRINT ( SENTENCE [ ALTHOUGH OLIVE
    LATER BECAME POPEYE'S BEST ] SELECT
    10 :WORDLIST "-FRIEND, [ SHE ] SELECT
    11 :WORDLIST [ HIM AT FIRST. SHE
    REALIZED THAT SHE TRULY ] SELECT 12
    :WORDLIST [ POPEYE WHEN HE SAVED
    HER FROM BRUTUS, AN EVIL ] SELECT
    13 :WORDLIST ". )
END
```

Each of these PRINT statements must be entered as though it were a single line. Don't press <RETURN> until you reach the final close parenthesis!

After we worked with this a number of times, we found ourselves wishing that the program had several Mad Libs to choose from. That's one possible extension.

Typing the Mad Lib into the procedure PRESENT.MADLIB was no fun either. Wouldn't it be nice if the computer could help this process? Could you write a set of procedures that would simplify the process of creating a computerized Mad Lib program? It would be desirable to have a procedure that asked you to type in the Mad Lib. If the computer could take what you typed in and ask you for the correct number of words and even tell you what kind of word, that would be helpful. The final step would be for the computer to present the Mad Lib without requiring that it be typed into a procedure like PRESENT.MADLIB.

For readers wishing to get started on this project, we present two procedures that may be helpful. Type in these procedures and then run the GET procedure.

```
TO GET
  PRINT [ TYPE IN YOUR MAD LIB. WHEN ]
  PRINT [ YOU COME TO A PLACE WHERE ]
  PRINT [ A WORD IS GOING TO BE ]
  PRINT [ SUBSTITUTED, TYPE IN A ]
  PRINT [ NUMERAL SIGN (#) AND A WORD ]
  PRINT [ THAT TELLS WHAT KIND OF ]
  PRINT [ WORD IS NEEDED. ]
  PRINT [ ]
  PRINT [ FOR EXAMPLE, YOU WILL BE ]
  PRINT [ TYPING IN SENTENCES THAT ]
  PRINT [ CONTAIN THINGS LIKE . . . ]
  PRINT [ J. W. WHIMPY IS FAT AND #ADJ ]
  PRINT [ AND HIS WHOLE LIFE IS ]
  PRINT [ DEVOTED TO #'ING'VERB ]
  PRINT [ HAMBURGERS. ]
  MAKE "MADLIB REQUEST
  MAKE "BUFFER [ ]
  MAKE "WORDLIST [ ]
  MAKE "NEWLIB [ ]
END
TO REORGANIZE :TEXT
  IF :TEXT=[ ] OUTPUT LPUT :BUFFER
    :NEWLIB
  TEST FIRST FIRST :TEXT = "#
  IFTRUE MAKE "WORDLIST LPUT FIRST :TEXT
    :WORDLIST MAKE "NEWLIB LPUT :BUFFER
    :NEWLIB MAKE "BUFFER [ ]
  IFFALSE MAKE "BUFFER SENTENCE
    :BUFFER FIRST :TEXT
  REORGANIZE BUTFIRST :TEXT
END
```

Now type REORGANIZE :MADLIB and press <RETURN>. Does this suggest some possibilities?

Next type in PRINT :WORDLIST and press <RETURN>. Hmm! Good luck on this project. Write us if you need some more help.

END■

[Tim Riordon, 3375 University, Eugene, OR 97405. Kathleen Martin, Dept. of Education, Univ. of Dallas, Dallas, TX 75061.]

Optical Illusions— A Challenge in Problem Solving

by
Donna Bearden

One of the reasons I like using Logo is that it really does promote clearer thinking and sharpen problem-solving skills. It's not just promotional hype. Because the computer can only reflect what the user is thinking, s/he can see those thought processes pictorially represented. People *can* work out the bugs in their thinking and develop problem-solving ability.

I've seen it happen. In fact, one of the rewards of working with children and Logo is being able to witness those "Aha!" moments when something clicks in the thinking process. I've felt it happen in my own head—and these moments become more frequent as I explore and learn more about Logo.

Recently I was thumbing through an educational products catalog and "the impossible triangle" happened to catch my eye.

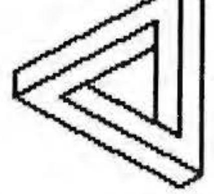


Figure 1.

"Hmm, that's just an equilateral triangle with a few extra lines surrounding it," I thought, innocently enough. And the challenge had been issued.

An optical illusion will either drive you crazy or sharpen your problem-solving skills. Just when you think you have it figured out, it shifts right before your eyes.

I tackled the triangle. I tried it on the screen. I tried it with pencil and paper. What I knew had to be right wasn't right. It was only after many frustrating attempts that I saw the pattern. It is indeed an equilateral triangle with three identical "legs" attached to it. Define the leg and attach it at the three corners. Aha!

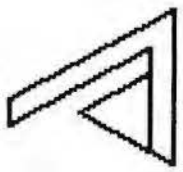


Figure 2.

Once I saw the pattern, the rest became easy. Not that the solution happened in the next five minutes, but I knew I was on the right track.

```
TO OP.TRI
TRI
REPEAT 3 [ LEG FD 30 LT 120 ]
END
TO TRI
REPEAT 3 [ FD 30 LT 120 ]
END
TO LEG
FD 40 LT 120 FD 60 RT 120
FD 10 RT 60 FD 70 RT 120
FD 60 RT 120 FD 10 RT 60
END
```

When the "impossible triangle" was tamed, it seemed too small and insignificant on the screen, at least in comparison to what it had put me through. I had drawn it small, using increments of 10 to keep the math simple. I stared at the tamed triangle and thought, "All that for this? Hardly something to show the neighbors!"

So I rotated the impossible triangle and made the impossible hexagon. Aha! That felt good!

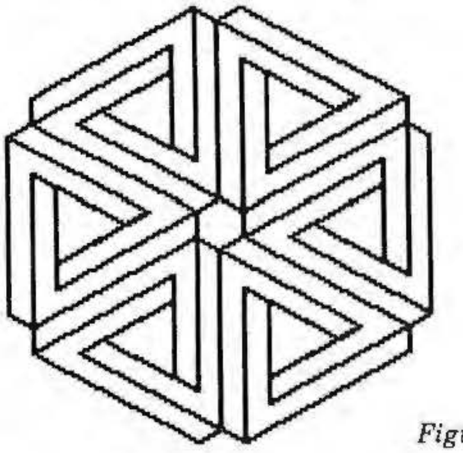


Figure 3.

```
TO SWIRL.OP.TRI
REPEAT 6 [ OP.TRI MOVE ]
END
```

```
TO MOVE
PU BK 20 LT 120 FD 20 RT 60 PD
END
```

It wasn't long after that I discovered John Locke's book *Isometric Perspective Designs and How to Create Them* (Dover, 1981). The entire book is filled with optical illusions made from cubes. The cube is made from three parallelograms. Each parallelogram is made from two equilateral triangles put together.

Because the equilateral triangle is used as a base, every design has only two angles: 60 and 120. It's a good place to start if you want to tackle some optical illusions. In fact, it's fun to simply study the designs and figure out the repeating patterns. Every turtle turn will be RT or LT 60 or 120. With luck, you'll be right 50 percent of the time. (Don't laugh—optical illusions shift on you.)

I studied Locke's book for several hours to pick out my next challenge. Here it is:

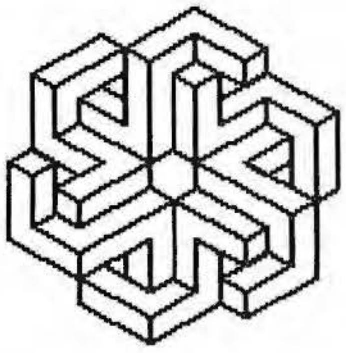
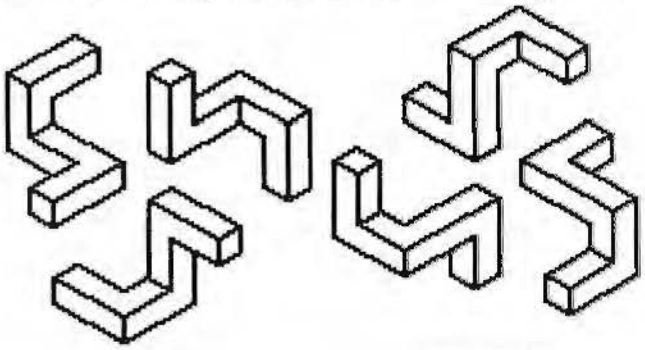


Figure 4.

It appeared to me that it was a hexagon with six crooked, but identical, legs surrounding it. So I defined the leg and rotated it on the screen. I refuse to show you the MESS.

After several attempts of trying to figure out where to position the turtle to rotate the legs correctly, I realized that three of the legs were as I had defined them, the other three were not. So I defined TRIPLET1 and TRIPLET2. And figured out how to rotate the triplets and then put them together:



TRIPLET1 TRIPLET2

Figure 5.

Procedures—Optical Hexagon

```
TO OPTICAL.HEXAGON
TRIPLET1
TRIPLET2
END
```

```
TO TRIPLET1
S
REPEAT 2 [ PU FD 20 PD S ]
END
```

```
TO S
SIDE
LT 60 DIA
RT 60 FD 10 LT 60
VEE
VEE2
VEE3
SIDE2
END
```

```
TO SIDE
REPEAT 2 [ FD 10 RT 60 FD 30 RT 120 ]
END
```

```
TO DIA
REPEAT 2 [ FD 10 RT 60 FD 10 RT 120]
END
```

```
TO VEE
FD 10 RT 120 FD 20 LT 120 FD 20 RT 120
FD 10 RT 60 FD 30 RT 120 FD 30
END
```

```
TO VEE2
RT 60 FD 10 RT 120 FD 20 RT 180
FD 10 RT 60 FD 20 RT 60 FD 30 RT 120
FD 10 RT 60 FD 20 LT 60 FD 20 RT 120
END
```

```
TO VEE3
RT 60 FD 20 RT 60
FD 20 RT 60 FD 30 RT 120 FD 10 RT 60
FD 20 LT 60 FD 10 RT 60 FD 10
END
```

```
TO SIDE2
RT 120 FD 20 LT 60 FD 10 RT 120
FD 30 RT 60 FD 10
END

TO TRIPLET2
S2
REPEAT 2 [ PU BK 30 LT 60 BK 30 LT 60 PD ]
END

TO S2
L
L2
L3
BK 10 RT 120 FD 20 LT 120
DIA
END

TO L
FD 10 LT 60 FD 20 RT 60 FD 20 LT 60
FD 10 LT 120 FD 30 LT 60 FD 30
END

TO L2
LT 60 FD 30 RT 120 FD 20 RT 120 FD 10
RT 60 FD 10 RT 60 FD 10 RT 120 FD 20
LT 120 FD 10 LT 60 FD 30 LT 120 FD 40
END

TO L3
BK 40 RT 60 FD 10 LT 60 FD 30 RT 60
FD 10 LT 60 FD 10
END
```

What a feeling of accomplishment!

The process I went through to solve the optical illusions re-emphasized the importance of looking for patterns in problem solving—a very important concept that spills over into every facet of life. And that's what Logo is all about. END■

[Donna Bearden is the author of 1, 2, 3 My Computer and Me *(Reston);* ATARI Logo in the Classroom: A Teacher's Manual *(Atari, Inc.); and* A Bit of Logo Magic *(Reston). She is co-author of* The Turtle's Sourcebook *(Reston). She and Kathleen Martin ("The Logo Center") have formed a Logo computer curriculum development and training firm and are working on a series of booklets on mathematical activities on and off the computer. Donna Bearden, 1908 Sandy Lane, Irving, TX 75060.]*

MACUL '84

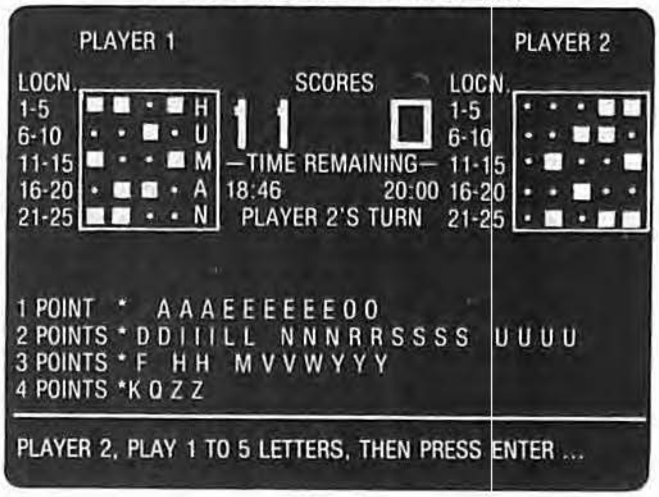
March 15-16, 1984

Grand Rapids, Michigan

WORD WARS

A Vocabulary-Building Crossword Game
Dictionary Use During Play Encouraged!
Solitaire and Two-Player Options
Nine Playing Levels (6 in TRS-80 16K)
Recommended for Grades 6-12
Suitable for School and Home Use (Subroutines defeat most attempts to use naughty words!)

TRS-80 32K SCREEN FORMAT:



```
     PLAYER 1                                         PLAYER 2
LOCN.                       SCORES           LOCN.
1-5   ■ ■ • ■ H    1 1               0        1-5    • • • ■ ■
6-10  • • ■ • U                               6-10   • • ■ ■ •
11-15 ■ • • ■ M   —TIME REMAINING—           11-15   • ■ • • ■
16-20 • ■ ■ • A   18:46          20:00       16-20   • • ■ • •
21-25 ■ ■ • • N    PLAYER 2'S TURN           21-25   • ■ • ■ ■

1 POINT  *  A A A E E E E E E O O
2 POINTS * D D I I I L L  N N N R R S S S S  U U U U
3 POINTS * F  H H  M V V W Y Y Y
4 POINTS *K Q Z Z

PLAYER 2, PLAY 1 TO 5 LETTERS, THEN PRESS ENTER ...
```

- Players try to form words across and down in the grids.
- They can use only letters in the Letter Pool below the grids. The computer randomly generates a new pool and grids at the start of each new game.
- After the player selects letters, the computer asks for Starting Location & Direction (Across/Down). If the move is legal, the letters transfer to the playing grid.
- Non-words generate penalties that lower scores.
- Higher final score wins. If playing alone, player tries to beat the target score for playing level.

TRS-80/III 16K	$19.95 Cassette
TRS-80/III 32K	$24.95 Cassette
Apple II/IIe 32K	$24.95 Diskette
TI 99/4A 16K	$19.95 Cassette
Commodore 64	$24.95 Cassette

For FREE Instruction Booklet,
or to order, write to:
Rollingwood Publications
P.O. Box 364, San Pablo, CA 94806 (415) 724-6033

advertisement

Equipping Our Schools with Hardware/Software: How to Be a Reasonable Recipient of Donated Equipment

by
Shirley McCandless

Education is forming a new partnership with business and government across the nation to prepare our citizens for the "technology revolution." One way of fostering the relationship between business and education is to allow a business a tax deduction for donating equipment to an educational institution.

A bill, which provides a 40 percent tax credit to a business who donates computer hardware and software/courseware to educational institutions, recently passed in the Louisiana Legislature. The bill's author, State Senator Ned Randolph, stated that the bill requires that the donor provide staff development/orientation on the equipment within two weeks after its installation.

The bill also requires that the donated software/courseware be compatible with existing hardware, and that the staff development/orientation be done at the site of the educational institution at no cost to the institution.

The receipt of donated equipment requires that the educational institution have criteria for the kind of hardware or software it will accept. Educators must examine free hardware just as carefully as if they were purchasing it and must consider the following questions:

1. What are the applications of the computer? (What do I want to do with a computer?)
2. What are the objectives and goals of a computer education program?
3. Have other schools purchased this hardware?
4. What are the reliability statistics on the computer hardware and peripheral devices?
5. What documentation exists for the operating system? Languages?
6. How does this hardware compare to its competitors for the uses I have in mind?
7. Can the vendor relate the technical specifications of the hardware to the user's needs?
8. What software exists for the hardware?
9. What kind of repair support is provided?
   a. Is there a trouble-shooting manual?
   b. Is there a "hot line" telephone service?
   c. Will the vendor make repairs on site?
   d. Where is the closest walk-in service center?
   e. Where is the "mail-in" service center?
   f. How long will services take?
10. What is the warranty period?

These are only a few of the questions that educators must answer before they accept donated hardware.

Accepting donated software/courseware should be done with the same care devoted to selecting other educational materials and for many of the same reasons. The following questions should be answered:

1. Does the software meet the identified goals and objectives?
2. Does the software take advantage of the capabilities of the computer (not just an electronic page-turner)?
3. Does the software fulfill its purpose?
   a. For initial teaching materials?
   b. Testing?
   c. Record-keeping?
   d. Review and practice?
4. Does the software call for peripherals?
5. Does the software provide useful feedback to the user?
6. Does the software make allowances for human factors? Is it "user-friendly"?
7. Has software been reviewed by content experts to assess the accuracy of content?
8. Is there a written validation report?
9. Are reviews and evaluations of courseware available from organizations, journals, and signed by other educators?
10. Is a list of sites available where it is being used in an educational setting?
11. Is the free software/courseware compatible with existing hardware? END■

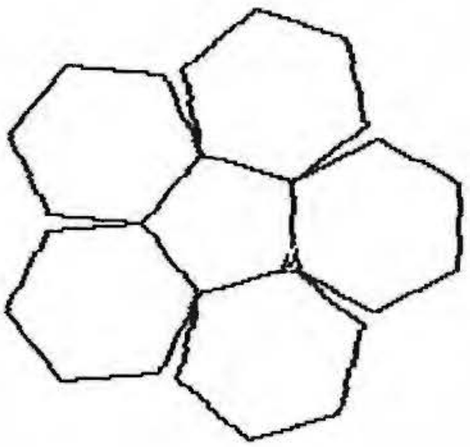
[For a copy of the Act, contact Shirley McCandless, Administrative Officer, Computer Education, State of Louisiana Dept. of Education, P.O. Box 44064, Baton Rouge, LA 70804.]

Dumping the Turtle

by
James H. Muller

Having a printer as part of your Logo system offers innumerable opportunities to enhance the learning process plus add to a child's confidence at the computer.

For example, I observed eight- and nine-year-old children who enjoyed the challenge of translating the two-dimensional computer screen into three-dimensional objects. Ohviously they didn't think about it in those terms. To them, it began with the seemingly simple task of duplicating the pattern of a soccer ball on the screen. This involved developing five hexagons connected by a pentagon.



The gaps between the hexagons were a bit puzzling until the pattern was dumped to the printer. Actually, twelve of these patterns were dumped to the printer, cut out and glued together into a soccer ball.

Naturally, once the ball was put together, they decided that it was no longer a soccer ball. Stuffed with candy, it became a class piñata.

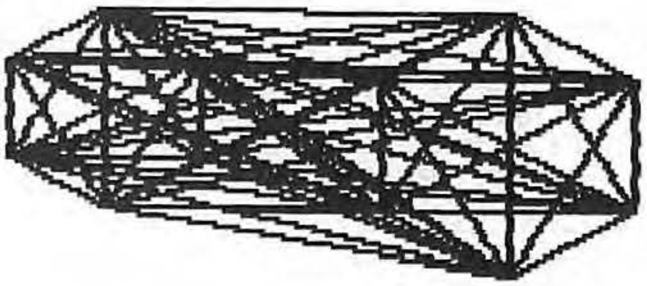
This exercise with elementary school children got some junior high school students thinking about three-dimensional objects. Were there Logo procedures that could simplify the designing of polyhedrons?

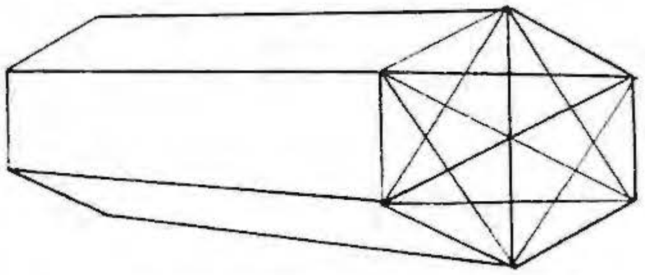
We have all learned to "never say never" when working with Logo. While working with some procedures from the YPLA Software Exchange, the junior high group found the way to explore a third dimension on the screen. We named the procedure POLYS.

POLYS is a single-keystroke procedure that allows the user to define and interconnect a set of points on the screen. Since some shapes can become too crowded, an ERASE command allows the user to eliminate unwanted lines.

One of the more fascinating aspects of this simple procedure is watching shapes evolve as points are placed on the screen. Stars come out in pentagons. Diamonds, flowers and a variety of other shapes show themselves as more and more points are defined. You can then extend them off into apparent infinity by defining another point off in the distance.

However, the fun really begins when the graphics are dumped to the printer. Using tracing paper, the students pick out the different planes of the figure and drop what would be hidden lines. The results are polyhedrons of different complexities.





Another enjoyable exercise has been to define three-dimensional shapes using the POLYS procedure, and then "flatten" them into a series of interconnected planes. Dumped to the printer, these figures can be folded into the three-dimensional objects.

Folding paper to make different figures is an enjoyable activity for all ages. The figures don't have to be geometric shapes. They can be boats, hats, houses or almost any repetitive design created on or off the computer.

* * *

To dump graphics and text from the computer, you first need a printer which will print graphic characters. Not all do, so this is a first consideration. Most dot matrix printers will print graphics characters, but you need to be able to talk to them in the right language. In some cases, this can mean talking to them in machine language.

Virtually all versions of Logo and Turtle Graphics allow you to print graphics and text. Versions of TI Logo and Color Logo are the only ones we're aware of that don't allow you to dump graphics. Logo versions for the Apple, Atari, Commodore 64 and IBM PC offer printout commands or utility routines. Delta Drawing and CyberLogo also allow the printing of pictures.

Because of the variety of options available for the Apple II versions of Logo, let's first consider the Apple Silentype Thermal Printer:

```
TO PRINTPIC
.PRINTER 1
.DEPOSIT 65536 -
  12524 0
.DEPOSIT 65536 -
  12528 7
DEPOSIT 65536 -
  12529 255
PRINT CHAR 17
.PRINTER 0
END

TO DUMP :SLOT
.PRINTER :SLOT
.DEPOSIT 53007 128
.DEPOSIT 53012 0
PRINT CHAR 17
.DEPOSIT 53012 255
.DEPOSIT 53007 0
.PRINTER 0
END
```

Those using other graphics printers with an Apple computer will require an appropriate interface card. Some have graphics software on board the interface card. Some others require you to crash out of Logo and then print the graphics. Still others have some limited graphics dumping capability on board the interface card. But dumping the graphics causes the language to crash. The language and the graphics dumping programs fight for the same memory space, and since Logo got there first, it usually wins and calls it quits.

The Grappler+ interface card is among those offering graphics dumping features. Earlier versions of the Grappler were designed for different printers. Now, however, the Grappler+ has a dip switch which allows you to set it to meet the requirements of a particular printer. These are listed in the Grappler's documentation.

Most computer manufacturers offering Logo software also offer dot matrix printers. Most often these are other brand name printers carrying the major manufacturer's name. For example, the Apple dot matrix printer is a lookalike of the C.Itoh Prowriter. It will operate at the settings listed in the Grappler+ manual for the NEC 8023 and C.Itoh Model 8510.

Remember also to check the switches in the printer itself. Factories set these to print text. But the interface card and/or the printing of graphics may require different settings. Unfortunately, some instruction manuals describe these settings in what appears to be another language. It pays to have your dealer check this out for you.

Once you have set the switches on the Grappler+ Board, and on the printer if required, a simple procedure will print graphics.

```
APPLE LOGO:              MIT LOGO:
TO DUMP :CMD             TO DUMP  :CMD
.PRINTER 1                 OUTDEV 1
(TYPE CHAR 9 :CMD          (PRINT1 CHAR 9
  CHAR 13)                   :CMD CHAR 13)
.PRINTER 0                 OUTDEV 0
END                      END
```

To print the pictures, enter DUMP followed by one of the following letter codes; for example, DUMP "GE.

"G	Regular print.
"GE	Enhanced or double-strike pictures.
"GED	Enhanced, double-sized pictures.
"GEDR	Enhanced, double-sized pictures rotated 90 degrees. This command is required when printing double-sized graphics on the Epson and Microline printers. This is also a good way to solve the distortions inherent in the Epson printers.

Epson printers that use an Epson Interface card generally require you to crash out of Logo, load a graphics printing program, and then print the graphics. When you have a picture to print out, place the graphics program disk in drive 1. Then enter .PRINTER 6 (Apple Logo) or OUTDEV 6 using MIT Logo. The screen will then prompt you through the steps to print the graphics.

Another printer interface card for the Apple II computer is the PKASO card. To utilize the graphics capabilities of this card with Apple Logo or MIT Logo, define the procedure PRTPIC:

```
APPLE LOGO:              MIT LOGO:
TO PRTPIC  :CMD          TO PRTPIC :CMD
.DEPOSIT 47299 6         .DEPOSIT 47299 6
.DEPOSIT 47365 5         .DEPOSIT 47365 5
.PRINTER 1               OUTDEV 1
(TYPE CHAR 9 :CMD)       (PRINT1 CHAR 9
                           :CMD)
.PRINTER 0               OUTDEV 0
END                      END
```

To print graphics, enter:

DUMP "H	Regular size prints
DUMP "2H	Large-size prints
DUMP "10H	Large-size prints rotated 90 degrees
DUMP "12H	Extra-large prints rotated 90 degrees

For convenience, these graphics dumping procedures can be incorporated into the START-UP file by using the auto loading feature described in the Apple Logo manual.

"But what if I don't have the procedure in memory? How can I dump a picture I like?"

This happens quite a bit. And it is a problem easily solved. Once the procedure is stopped, simply enter on one line:

```
APPLE LOGO:
  .PRINTER 1 (TYPE CHAR 9 "G CHAR 13)
MIT LOGO:
  OUTDEV 1 (PRINT1 CHAR 9 "G CHAR 13)
```

Once the picture is printed, enter .PRINTER 0 or OUTDEV 0 to get out of the printer mode and back to the screen.

These procedures work with the Grappler+ interface card, and can be modified for other interface cards which have graphics printing capabilities on the interface card.

"But what if I don't have a Grappler or other interface with graphics printing software?"

The choice here is to do it the hard way. First you save the picture as a high resolution graphics binary file. Then you can use whatever graphics printing routines are available for your printer. Here are the steps required for Apple Logo:

1. Develop the Turtle Graphics procedures and save these to disk. Run the procedures so that the picture is on the screen.
2. Put an initialized disk into the drive you will use to save the picture.
3. Enter .PRINTER 6 to crash out of Logo.
4. When the flashing cursor appears, enter BSAVE (name of picture), A8192,L8192. You can save this picture to either disk drive by adding ",D2" to the BSAVE command if that's where your initialized disk is located.
5. To load the picture, boot the DOS 3.3 System Master. Enter HGR and then BLOAD (name of picture). You can then follow the instructions provided for dumping high resolution graphics to the printer. This is also an excellent way to send pictures home to the family with an Apple computer but without Apple Logo.

There is nothing hard and fast about these procedures. There are many other variations which may suit individual needs. How you do it is not important. What is important is to make effective use of the printer as another tool to make Logo come alive for all ages—including the teacher!

END■

[James H. Muller, Young Peoples' Logo Association, 1208 Hillsdale Dr., Richardson, TX 75081.]

1984 ACM Computer Science Conference

February 14-16, 1984
Philadelphia, PA

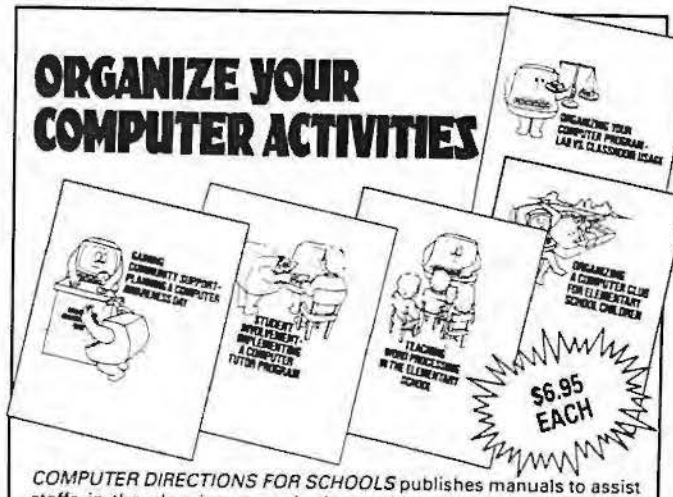
The Twelfth Annual ACM Computer Science Conference will continue ACM's emphasis on abstracts of current research along with invited papers, panels and refereed papers. Central themes are "Factory of the Future," "Coping with Small Computers" and "Social and Ethical Implications of Computers."

For further information and registration forms, contact Dr. Frank Friedman, ACM Computer Science Conference Chairman, Computer and Information Science Department, Temple University, Philadelphia, PA 19122, 215/787-1912.

The Technical Symposium on Computer Science Education of the ACM Special Interest Group on Computer Science Education (SIGCSE) will be held on February 16 and 17 in conjunction with the conference. For further information, contact Dr. Richard H. Austing, Department of Computer Science, University of Maryland, College Park, MD 20742, 301/454-2004.

advertisement

ORGANIZE YOUR COMPUTER ACTIVITIES



COMPUTER DIRECTIONS FOR SCHOOLS publishes manuals to assist staffs in the planning, organization and implementation of various computer-related activities at their schools. The manuals were written primarily with the elementary school in mind; however, many of the ideas can be easily adapted at the junior high level. The material is presented in an easy-to-read format, with helpful attachments to facilitate effective school computer management.

California residents add applicable sales tax. Shipping and Handling: 1-2 books $2.00, 3-5 books $3.00; 6-above 10%.

SCHOOL PURCHASE ORDERS ACCEPTED. INDIVIDUALS PLEASE ENCLOSE PAYMENT IN FULL. MAKE CHECKS PAYABLE TO:

COMPUTER DIRECTIONS FOR SCHOOLS
P.O. BOX 1136 H
LIVERMORE, CALIFORNIA 94550

COMPUTERS IN THE TEACHING OF ENGLISH

Robert Shostak • Lester S. Golub

[Editor's Note: While English teachers fortunate enough to have access to microcomputers to use with their students agonize over the dearth of good software, some of their colleagues are creating effective "programs" of their own with help from an unexpected source—the simple word processor.

These teachers, who have already discovered the benefits of using the word processor for composing and editing, are now finding exciting new applications for what many of us believed was a single-purpose piece of software. In the article which follows, Karen Piper describes how she has capitalized on some of the word processor's basic capabilities and developed "programs" to generate structured writing exercises and encourage creative writing. I have a feeling that after reading about Karen's efforts, many of you will not only want to try writing similar exercises but will also discover other useful applications. Why not share them with other TCT readers?]

The Electronic Writing Machine: Using Word Processors with Students

by
Karen L. Piper

Tempting students to become writers is not easy because learning to write well requires practice and hard work. You can make the writing act easier for your students, however, by the instructional use of word processors. Writing instruction using word processors will still require practice, but the mechanical process of writing can be eased. Students of all ages can benefit from the use of a tool that makes writing less cumbersome and more motivating.

The word processor allows students to write and revise with ease. Several aspects of writing with word processors contribute to this effect:

- Writing with word processors allows students to make insertions and deletions quickly and easily.
- Students can move words, phrases, sentences or paragraphs without messy erasures and scribbling.
- Drafts can be printed out, allowing students to edit clean copies and revise on the screen at a later time.
- Students don't experience "writer's cramp."

Using the Electronic Writing Tool

Many instructional uses exist for word processing programs. Students can use the word processor to complete creative writing assignments. Students do such writing more fluently as they develop writing techniques. I recently explored the use of word processing programs to deliver sentence combining instruction and encourage creative writing through story expansion. This technique allows students to practice manipulating written language structures in an effort to expand certain areas of writing ability.

With the guidance of Dr. Michael Angelotti, Texas Tech University, I completed an eight-week research study with fifth graders in Abernathy, Texas. I used a word processor to develop sentence-combining exercises for student completion and expansion. I wrote "clusters" of kernel sentences, which the students manipulated into one "writeout" per cluster—a sentence that combines all of the thoughts present in the cluster. To do so, the students delete redundant information; add conjunctions and commas; combine independent and dependent clauses; embed prepositional phrases, adjectives and adverbs; and make decisions concerning word order and choice.

For example: students are to combine the following cluster:

1. The coach was strong.
2. The coach was smart.
3. The coach encouraged her players.

To do so, students might embed and combine to create the sentence "writeout":

The strong, smart coach encouraged her players.

Once the combination is complete, students check the sentence to make sure the original meanings are present. In many cases, there is more than one correct writeout, and students are encouraged to try various approaches and manipulate the language to find the "writeout" that reads best to them.

Each exercise consists of approximately five clusters. A specific theme or story line runs through each exercise. As students complete the exercise, they expand the resulting writeouts into a story of their own.

The results of using microcomputer-delivered instruction in sentence combining and story expansion with fifth graders indicate that:

1. Fifth graders enjoy completing microcomputer-delivered sentence combining exercises.
2. Students eagerly expand the writeouts into their own stories.
3. Students who worked with sentence-combining exercises developed positive attitudes toward writing with the computer and toward sentence-combining activities.
4. Students who participated in the study improved in their writing ability, as measured by factors of writing maturity.
5. Students in the experimental study were highly motivated to write using the word processor.

Student Comments

Student comments concerning the sentence-combining activities of the microcomputer reveal specific points that are useful to teachers who wish to use the microcomputer as a writing tool. Some of these comments include:

"I liked that my hand did not get tired."

"It was easy to correct my mistakes."

"I liked writing on the computer because it was easier to erase and move things."

"It [the words on the screen] looks neater than my handwriting and I can find my mistakes better."

From the student comments and the overall results of the study, we conclude that the microcomputer can be used as an effective tool for structured writing instruction. For those of you who wish to try it in your own classroom, I offer a few tips:

1. Choose a word processing program that students can quickly and easily use.
2. Provide instruction in the use of the word processing program chosen.
3. Provide typing instruction or keyboard familiarity prior to using the word processor as a writing tool.
4. Train student assistants to help students get started with the word processor.
5. Prepare your text files carefully and well in advance. Know the objectives of your exercises and structure them accordingly.
6. Make sure you have a printer that will work with your microcomputer and word processing program.
7. And finally, be prepared to make arrangements for students who won't want to stop, or who will ask to come in early or stay late.

Given appropriate instruction and practice through activities such as sentence combining, students can become better, more motivated writers.

END■

[Karen L. Piper, Texas Tech University, P.O. Box 4560, Lubbock, TX 79409.]

[Dr. Robert Shostak, Florida International University, School of Education, Tamiami Campus, Miami, FL 33199. Dr. Lester Golub, Dean, School of Education, Baruch College, City University of New York, 17 Lexington Ave., New York, NY 10010.]

advertisement

DIGITAL IMAGES

COMPUTER SUPPLIES FOR EDUCATION

DISKETTES

wabash DataTech

6 Year Warranteed — Error Free

for APPLE II and APPLE IIe

5¼ SS/SD with Hub Ring $1.60*

for TRS-80 Model III and IV

5¼ SS/DD with Hub Ring $1.90*

* In case quantities of 100 diskettes; packed 10 per box with envelopes and labels.

for all other popular computers... call for information and prices.

APPLE II and APPLE IIe are registered trademarks of APPLE COMPUTER, INC. TRS-80 is a registered trademark of TANDY RADIO SHACK CORP.

Payment — Purchase Order, Visa, MasterCard, Check
For C.O.D. orders add $3.00.
Shipping — Please add $5.00 per case of 100 diskettes.

We also stock a complete line of diskette storage devices, printer ribbons, diskette care kits, voltage surge protectors and joy sticks.

----Write or call for details and prices.----

Easy Direct Ordering — Toll Free
(800) 222-1248 In Illinois (312) 882-8315

DIGITAL IMAGES

1133 Tower Road, Schaumburg, IL 60195

BOOK REVIEWS

Regan Carey

LEARNING WITH LOGO

By Dan Watt
1221 Avenue of the Americas, New York, NY 10020.
1983, 365 pages, $19.95. Educator discount: 1-4: 20%; 5+: 25%.
ISBN # 0-07-068570-3
Reviewed by Jim McCauley, Santa Clara County Office of Education, 100 Skyport Dr., #237, Santa Clara, CA 95115.

Teachers around the world have enjoyed and learned much from Dan Watt's many magazine articles on the educational uses of Logo. At long last, his first book on the subject is available.

It was worth the wait.

The book is very large, fairly expensive and absolutely indispensable. About two-thirds of the book is intended for direct use by students and teachers on a day-to-day basis. The balance is resource material, including instructions for using the book with Logo dialects other than Terrapin Logo, creation of special procedures disks to supplement the text and a reference list of Logo commands.

While there have been several books on Logo aimed at learners (Abelson's Logo primers, Thornburg's *Discovering Apple Logo* and Ross' *Introducing Logo*), few publications have directly addressed the needs of classroom teachers. By means of clever formatting, Watt succeeds in speaking to both audiences in one book. The bulk of the book is student-centered, beginning with a solid introduction to programming through turtle graphics and continuing with several large projects and their variations, which form the centerpieces of chapters 3 and 8 through 13. Some examples: an interactive turtle graphics game of "target practice"; a drawing program for young children that "remembers" what it has drawn; and a list-processing program called POET that writes free verse.

Other sections of the book help students develop some specific computing skills. Chapter 4 is perhaps the strongest introduction to Logo's editor and workspace management written to date.

Cartoonist Paul Trap has provided charming illustrations that complement the book's cheery, open style. Special cartoon symbols help to draw attention to themes repeated throughout the book: Pitfalls, Powerful Ideas, Explorations and Helper's Hints.

These "Helper's Hints" are specifically designed to help teachers and parents who are new to Logo. Many of these deal with the mechanics of Logo, but several are intended to encourage a style of relationship between teacher and learner that promutes learner independence and a sort of "educational democracy." A good example is on page 158, where the author asks teachers to avoid creating an environment in which "students are usually expected to know the 'correct' answer before they speak." Instead, he asks teachers to "establish an expectation of free discussion of the ways people are thinking about what the computer is doing," and he concludes:

"If you are a teacher or a parent, the best way to foster this kind of atmosphere is to expose bugs in your own thinking. More than anything else, this will help the learners you're working with develop the confidence to risk explaining their thinking in public."

The philosophical base of the book is built entirely on this sort of relationship. Students who are using this book will expect nothing less from their teachers. This is welcome and refreshing news for those who seek to create a cooperative learning environment in their classrooms. Most such teachers would agree that to claim sole authority and expertise is to place oneself in an uncomfortable position. *Learning With Logo* is a solid guide to a computer-education style which places the student in the center of classroom concerns.

The only things missing from the book are a bibliography and a separate index to the Helper's Hints, which could be particularly useful to teachers. Many of the Hints make good reading on their own outside the context of the book, and an index might also simplify lesson planning for teachers who are forced to use Logo under less than ideal conditions (one computer for thirty children, for example).

The publisher has thoughtfully produced the book in spiral-bound form, making it easy to use while at the computer.

Other editions specific to other dialects of Logo are promised in the future, but the present book is usable for Apple Logo and TI Logo. (Terrapin's Commodore 64 Logo is syntactically identical to their Apple version, so the book may be used immediately with that product.)

I recommend *Learning With Logo* without hesitation or reservation.

Distributors:
UK: McGraw-Hill Book Co. (UK) Ltd. Shoppenhangers Rd., Maidenhead, Berkshire SL6 2QL.
Canada: McGraw-Hill Ryerson Ltd. 330 Progress Ave., Scarborough, Ontario M1P 2Z5.
Australia: McGraw-Hill Book Co., Australia Pty. Ltd., 4 Barkoo St., Roseville East, N.S.W. 2026.

THE RULE OF 360

By Kathleen Martin and Donna Bearden
Martin-Bearden, Inc., 1908 Sandy Lane, Irving, TX 75060
1983, 49 pages, $7.95.
Accompanying Disk $4.95
Reviewed by Nickie Polson, 2978 Martin Rd., Campbell River, B.C., Canada V9W 1M3.

The Rule of 360 provides opportunities to use Logo as a tool in a math classroom. It is a sourcebook of ideas, activities and worksheets—the kind of book you can read today and use in your classroom tomorrow. Kathleen Martin and Donna Bearden have designed materials which allow children to explore important mathematical ideas starting from concrete activities. They have specifically targeted ages ten to twelve, but these limits may be extended.

The title reflects the type of ideas explored in the activities. A variety of polygons and their angles are investigated, first using mirrors and protractors and then using turtle graphics. Students are challenged to write Logo programs which will extend their understanding of those ideas.

The Rule of 360 has five sections building from this central theme. First is an investigation of polygons to discover that their central or exterior angles always add up to 360. Then relationships between geometric shapes are examined using tangrams and their computer cousins. Pattern recognition is exercised by looking at a card trick, then at circular number lines and multiplication tables. More manipulation of polygons occurs in the fourth section, with a look at tesselations. Finally, repeating patterns which form optical illusions are shown.

Most of the diagrams used as illustrations and for worksheets are produced using turtle graphics. The appendix has listings in Apple Logo of all the programs used. They are also available on disk for Atari, Commodore, TI and 2 Apple versions, MIT and Apple Logo. Worksheet masters which support the activities are also found in the appendix.

To use this book to best advantage, both students and the teacher should have some experience with Logo. It is necessary to be able to write and edit simple procedures and to be familiar with angles and distances as seen by the turtle. With such experience, both teacher and students can have fun with these activities while developing and absorbing some important ideas. The material is designed to be interesting while developing problem-solving skills, exercising skills in Logo programming and building a vocabulary of geometric terms. It is useful for individuals, small groups or for a whole class.

TURTLE'S SOURCEBOOK
Second Edition

By Donna Bearden, Kathleen Martin and James Muller

Reston Publishing Co., Inc., 11480 Sunset Hills Rd., Reston, VA 22090

1983, 226 pages, $21.95

ISBN #0-8359-7890-7

Reviewed by Rick Billstein, Dept. of Mathematical Sciences, University of Montana, Missoula, MT 59812

The *Turtle's Sourcebook* is a collection of activities and worksheets that the authors have compiled over a period of time. Suggestions for teaching the material are also included. It should be noted that many activities have been developed for off-computer use and can be used with groups in a classroom situation. The authors state that all activities have been classroom tested.

The book is written for four versions of Logo: TI 99/4, TI 99/4A, Apple Logo and MIT Logo (Krell and Terrapin for the Apple). Each time a new topic is introduced, the differences for the listed versions are given. For example, to erase an error the following are listed:

TI 99/4	Hold the SHIFT key down and press T
TI 99/4A	Hold the FUNCTION key down and press 3
Apple	Use the left arrow key
MIT	Press the ESC key

Although this may be a distracting format for many new Logo users, it was better than I thought it would be. Although not as clear as having a book written for the particular version of Logo being used, it is workable.

Many reproducible "cue cards" are included. Cue cards are sheets with large printing that can be hung up in the room to cue users for particular commands. For example, there is a cue card for the basic turtle commands and another one for pen commands. When the commands are different for the various Logo versions, a separate cue card is provided for each version. However, in an attempt to include cue cards for all versions, some mistakes were made. For example, all the pen commands listed for the MIT version (p. 37) will not work. PENERASE, which is an Apple Logo primitive, is listed and it will not work with MIT Logo. Also, PC0 will give an error message unless a space is placed between PC and 0.

The Turtle's Sourcebook has many interesting ideas for teaching Logo. For example, there is an activity where students make turtles out of walnuts and perform motion activities with them. In another activity, the clock is used to teach turtle turning. Big Trak™ is also discussed as a way of introducing Logo. I particularly like the Turtle Baseball activity (pp. 65-66) for introducing initial turtle movements. Familiar items in the classroom such as geoboards and tangrams are explored in a Logo setting. Another interesting activity called the Rule of 360 lays the groundwork for the Total Turtle Trip Theorem.

Recursion is introduced, and it is pointed out that recursion is not looping and that it is much more powerful than looping. Yet all the examples that are introduced involve tail recursion which can give the impression that recursion can be thought of as looping. Some examples of embedded (non-tail) recursion should be included.

There are a few errors scattered throughout the book, but they do not usually detract from the text. For example, the END statements are missing on many of the procedures on pp. 154-155 and 183. The orientation of some of the figures (pp. 119-121) are horizontal when they should be vertical. One item which might cause difficulty later is the way that a WAIT procedure is defined for MIT Logo. The authors introduce a procedure called WAIT that essentially has no purpose except to cause program execution to pause. The procedure they define (p. 152) is given below:

```
TO WAIT :T
REPEAT :T[REPEAT 4 [RT 90]]
END
```

This procedure causes the turtle to spin in circles the designated number of times. This has the disadvantage that if used outside of the DRAW mode, it causes Logo to enter the DRAW mode in order to perform the pause. This can cause problems in certain situations when work is being performed in the NODRAW mode. A WAIT procedure which avoids this problem is given below.

```
TO WAIT :T
IF :T = 0 STOP
WAIT :T-1
END
```

List processing is also included as a topic for exploration. Because list processing skills take practice to develop, many of the activities can be hard to follow if the reader is not familiar with some list processing. As Apple Logo and MIT Logo handle lists differently, many procedures will not work for both versions; for example, the CIRCLE procedure (p. 184) will work for Apple Logo but not for MIT Logo. It is implied that the procedure will work for both versions.

Educational SourceDisk

A Data Base Program!
A Word Processor!
A Directory

Select from over 200 software publishers who can meet your needs:

By: Computer
Apple — IBM — Atari
Commodore — Radio Shack
By: Curriculum Area
Twelve to Choose from!
By: Grade Level
K-8 College
H.S. Adult

Letterwriting Capabilities

Automatic Addressing From Database

Eliminates weeks of effort in Database Collection

Eliminates hundreds of hours of Data Entry!

Publishers Only — Eliminates Duplicate Requests, Unnecessary Source Confusion.

Tailor Your Request to Fit Your Needs, — Make Every Inquiry Count!

NEW

$79.95

Back Ups
Periodic
Updates

EDUCATIONAL ASSOCIATES
P.O. Box 35221
Phoenix, AZ 85069
602-869-9223

Runs on Apple* II, II plus, IIe
Apple is a registered trademark of Apple Computer, Inc.

advertisement

The text contains an excellent Bibliography and Resource Guide for Logo materials and articles. A Logo bibliography will be changing rapidly in the next year, but this is an excellent start.

In summary, I find the *Turtle's Sourcebook* a valuable resource for any person learning Logo or for teachers looking for ideas on how to teach Logo concepts. I expect the authors to be continually upgrading and expanding the *Sourcebook*. It is a great idea.

1,2,3 MY COMPUTER AND ME: A LOGO FUNBOOK FOR KIDS

By Donna Bearden
Reston Publishing Co., Inc., 11480 Sunset Hills Rd., Reston, VA 22090
1983, 99 pages, $10.95
ISBN #0-8359-7890-7
Reviewed by Rick Billstein, Dept. of Mathematical Sciences, University of Montana, Missoula, MT 59812

1,2,3 My Computer and Me is a workbook for elementary students. It is essentially a "kid's version" of the *Turtle's Sourcebook*. Many of the activities and pages are taken directly from the *Sourcebook*. This book, like the *Sourcebook*, is written for four versions of Logo: TI 99/4, TI 99/4A, Apple Logo and MIT Logo (Terrapin or Krell for the Apple). This four-version format can be confusing for many beginning users who are not even aware that various versions exist. The artwork is light and attractive and many of the sheets are actual worksheets for students to write on.

The book starts with an introduction to the basic turtle movements and proceeds through procedures involving variables. Recursion and color are introduced in later chapters. All recursion examples are tail recursion, and little discussion of recursion is included. A teacher would need a good understanding of recursion or some misleading ideas might be formed. Chapter 6 presents students with some project ideas and allows them some originality in designing some of the projects. The only new commands introduced in this chapter are SETX and SETY (Apple) and SX and SY (TI). Teachers planning to use this text with a class should have a copy of the *Sourcebook* as a resource.

Many of the comments concerning the content of the book have already been made in the *Sourcebook* review and will not be repeated here. This is one of many new Logo books coming out for classroom use. It is worth your time to examine it to see if it meets your needs.

Distributors:
UK: Prentice Hall International, 66 Wood Lane End, Hemel Hemstead, Herts HP2 4RG.
Canada: Prentice Hall of Canada, 1870 Birchmont Rd., Scarborough, Ont. M1P 2I7.
Australia: Prentice Hall of Australia Pty. Ltd., 7 Grovesnor Place, Brookvale, N.S.W. 2100.

PUBLISHER'S REPLY:

Thank you for giving us the opportunity to reply to Rick Billstein's review of the *Turtle's Sourcebook* and *1,2,3 My Computer and Me*. We appreciate the favorable comments and would like to respond to Dr. Billstein's criticisms. (We introduced a preliminary version of the *Turtle's Sourcebook* months before the final version went to press, and incorporated many users' suggestions into the final version.)

The MIT pen command cue card (p. 37) does not include PENERASE, as stated by Dr. Billstein, but PC0 should be typed PC 0. As for the discussion of recursion in Chapter 6 and the tail vs. non-tail recursion examples, Chapter 7 includes more about recursion and introduces conditionals as well as a discussion of recursion versus iteration. The Towers of Hanoi program is included in Appendix C for further exploration of the recursive process. Regarding the WAIT procedure on p. 152, Donna Bearden comments that this procedure is merely one suggested method to handle a pause in program execution, and that she much prefers Dr. Billstein's WAIT procedure. Finally, Dr. Billstein caught another typo in the CIRCLE procedure on p. 184. The MIT version of this procedure should not include brackets. END■

[Regan Carey, ICCE, Univ. of Oregon, 1787 Agate St., Eugene, OR 97403.]

Classified Ads

FREE PREVIEW OF OUR SOFTWARE FOR 30 DAYS ON YOUR OWN APPLE COMPUTER. We have diskettes on Algebra, Spanish, French, German, Russian and English. Send your request to: George Earl, 1302 South General McMullen, San Antonio, TX 78237.

CREATE-A-TEST—Apple II program makes perfectly formatted tests in 10 minutes! Use prepared question disks or make your own. 19 question disks available, cover chemistry, physics, biology and physical science—over 9000 questions. Write your own questions disks with the built in text editor. CREATE-A-TEST can handle almost any kind of question. Write: CROSS EDUCATIONAL SOFTWARE, Box 1536, Ruston, LA 71270 or call 318-255-8921.

ORGANIZE YOUR K-8 COMPUTER ACTIVITIES. We publish manuals to assist you. Titles include: Teaching Word Processing, Organizing a Computer Tutor Program, Lab vs. Classroom Computer Usage, Organizing a Computer Club, Organizing a Computer Awareness Day. Only $6.95 plus applicable CA tax. Shipping: 1-2 books, $2; 3-5 books, $3. Order today from COMPUTER DIRECTION FOR SCHOOLS, P.O. Box 1136(C), Livermore, CA 94550.

LOGO UTILITIES/APPLE. Logo-machine code gives EASIEST two-drive use, five-way text on graphics, sounds, animation. Terrapin or Krell Logo needed. $14.95 for disk: SRG, 431 Washington Ave., Bethlehem, PA 18017.

FREE Catalog of Educational Software. Why pay more for your favorite educational software? Get a 20% discount from CHAMBERED NAUTILUS, 16 RIVERGLADE CT., SACRAMENTO, CA 95831.

CLASSIFIED ADVERTISING

Rates: $2.50 per line. Minimum 5 lines. Based on 40 characters or fewer per line, including spaces and punctuation.

Discount: Applies to multiple insertions without copy change which appear in consecutive issues only:

2-4 insertions: $2.25/line per insertion
5-9 insertions: $2.00/line per insertion

Classified advertising is not subject to 15% agency or 2% prepayment discounts.

Type ad clearly. Use no abbreviations, indicate number of times the ad should run. **Enclose payment with copy.** Include name, address and phone number. Mail to Classified Advertising, *The Computing Teacher*, University of Oregon, 1787 Agate St., Eugene, OR 97403.

NEW! MICROCOMPUTER WORKSHOPS HAS A NEW 1983-84 CATALOG.
Our popular math and distinguished English Achievement Series have been made available for more systems. We have NEW programs to offer such as French and Spanish Achievement, a NEW decimals series, additions to our algebra and fractions series, more science programs, and NEW this year, software for computer science teachers. 30-day evaluation period. Send for our FREE CATALOG OF K-12 SOFTWARE FOR THE APPLE, TRS-80, PET, COMMODORE 64, ATARI AND IBM PC. MICROCOMPUTER WORKSHOPS, 225 Westchester Ave., Port Chester, NY 10573, 914/937-5440.

TRS-80 COLOR Computer—educational program—preschool through grade 12 including language arts, reading, elementary and higher math, foreign language. Send for free catalog. Computer Island, Dept. T, 227 Hampton Green, Staten Island, NY 10312.

COMPUTER PARTS KIT—Contains circuits from each generation, disks, magnetic tape, core plane, developer fluid and more. Use for display, lecture and group activities. Brochure available. $87.50 (US funds) plus shipping. Educational Computer Shoppe, Rt. 3, Box 601, Cambridge, MN 55008.

FOREIGN LANGUAGE AND ESL software for Apple and other micros. Wide selection. Free catalog. Preview of software by school request. LINGO FUN, INC., P.O. Box 486, Westerville, OH 43081.

The SCHOOL MICROCOMPUTING BULLETIN is published monthly Sept.-June. One year $28; two years $50; additional copies/same address $12 each/year. Easy-to-read, will save educators time, energy and $. For subscriptions call 616/372-1045 or write SMB, Learning Publications, Inc., P.O. Box 1326, Holmes Beach, FL 33509. Submit manuscripts up to 1000 words. For style guide/author rates, send stamped self-addressed envelope to Dr. Lee M. Joiner, 303 Bay Dr. N., Bradenton Beach, FL 33510.

KEEP YOUR CARDREADER WORKING ALL YEAR with QuickScore. For Apple equipped with Chatsworth OMR or Mountain 1100A, there's useful, reliable test scoring software. Single package price: $300. Includes diskette, operator's manual, sample cards, and shipping. Write or call: PICA Foundation, Box 36349, Charlotte, NC 28236; (704) 334-6444.

SPECIAL BOOKS BY TEACHERS, hands-on, easy-to-understand format. A BYTE OF THE APPLE—BEGINNER'S GUIDE $7. LET'S GET WITH THE PROGRAM $8, with SOLUTION DISK $13. Apple software at great prices: Format II $118, Delta Drawing $43, Bank Street Writer $45, Koala Pad $118, Gertrude's Puzzles $36, Flip File $22—$2 shipping per item. SLM, 3027 Tomahawk Dr., Lawrence, KS 66044.

Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685)

1. Title of Publication: *The Computing Teacher.*
   a. Publication No. 02789175
2. Date of filing: September 28, 1983.
3. Frequency of issue: Monthly except June and July; combined December/January issue.
   a. No. of issues published annually: 9
   b. Annual membership dues: $21.50 in U.S., $25 elsewhere.
4. Complete mailing address of known office of publication (*not printers*): ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403-1923.
5. Complete mailing address of the headquarters or general business offices of the publishers (*not printers*): ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403-1923.
6. Full names and complete mailing addresses of the publisher, editor, and managing editor. Publisher, none. Editor-in-chief, Dave Moursund. Managing editor, Dick Ricketts.
7. Known bondholders, mortgages, and other security holders owning or holding one percent or more of total amount of bonds, mortgages or other securities: None.
8. Extent and Nature of Circulation

	Average No. Copies Each Issue During Preceding 12 Months	Actual No. Copies of Single Issue Published Nearest to Filing Date
A. Total no. copies (*net press run*)	16,833	17,500
B. Paid Circulation		
1. Sales through dealers and carriers, street vendors and counter sales	0	0
2. Mail subscriptions	12,515	13,328
C. Total Paid Circulation (*sum of 8B1 and 8B2*)	12,515	13,328
D. Free distribution by mail, carrier or other means, samples, complimentary, and other free copies	1,045	1,164
E. Total Distribution (*sum of C and D*)	13,560	14,492
F. Copies not distributed		
1. Office use, left over, unaccounted, spoiled after printing	3,273	3,008
2. Returns from news agents	0	0
G. Total (*sum of E, F1, and 2—should equal net press run shown in A*)	16,833	17,500

9. I certify that the statements made by me above are correct and complete.

Joe Snyder
Treasurer, ICCE

PROGRAMMING THE PET
For elementary children and teachers. Learn BASIC in a non-technical, activity oriented way. Adaptable to other computers. Send $5.50 to Barbara Feddern, 57 Neil Drive, Smithtown, NY 11787.

Even though *Purser's Magazine* is out of business, I still have a few boxes of magazines left. Any American School overseas wanting free copies can send me up to five postage-free (penalty) mailing labels and return address. Robert Purser, Box 466, El Dorado, CA 95623.

TRS-80 COLOR PROGRAMS, fully documented. Make easy Hi-Res graphics, save to tape, print to printer, use saved pictures in your other programs, with built-in lowercase letters and over 25 commands, all in SCOLOR PAINTING—$24.95. MATH 3 game set—$24.95. SPELLING program—$14.95; input your own words or use optional graded word lists, eight available $5.95 each. For extra spelling fun, get WORD SHAKER, a scrambled word game—$12.95. All these programs are designed to keep children's interest. 16K Extended Basic required. Special learning package includes all the above and all eight word list tapes too for just $79.95! HURRY! Offer lasts two months only. Send check or money order to: COLORFUN, P.O. Box 1175, Monroe, NY 10950.

When you read a super article about your favorite subject, enter it into MAGAZINE CATALOG for the Apple II+, IIe. Later when you've forgotten where the article was, have the computer search and tell you. Create your subject categories. Option of printing all or part of file contents. Can also change data already entered. Send $12 to: RMH SOFTWARE, Box 41, Wilsall, MT 59086.

COMPUTER EDUCATION POSITION

Applications are invited for tenurable faculty position in computer education. Rank: Ass't. or Assoc. Professor. Salary: $20,868 to $28,884. Requires Ph.D. by fall 1984 in computer science, computer education, or math education, along with good teaching references, relevant research background, and interest in curriculum development related to computers. Closing date: Jan. 15, 1984. Applications received after that date will be considered if position is still open. Send vita and three references to Computer Education Search Committee, Dept. of Mathematical Sciences, San Diego State Univ., San Diego, CA 92182. AA/EO employer; we do not discriminate against the handicapped.

IS YOUR GRADEBOOK FAILING YOU? Does it instantly calculate the present average and letter grade for each student, class average for each test, print results? Will it effortlessly maintain student records, weight tests according to importance, keep a running total of points for each student, show incompletes? *Gradebook* does this and more—so you do less. Available for PET, Comm. 64. Apple & TRS-80 available on disk only. $29.95 includes postage if prepaid (school P.O. add $2). Order today without risk on 30-day approval from K-12 MicroMedia, Dept. GB, 172 Broadway, Woodcliff Lake, NJ 07675.

ONLY $4.95

Do apple hi-res graphics the FAST DRAW way. No accessories. Turtle-type input. Adjust aspect ratio! Type it yourself—just 48 lines. Enclose this ad for special offer. Check or M.O. to: MICRO CRAFTS, Box 670, Glendale, CA 91209.

Teachers, are parents asking you which computer they should buy for their kids? Robert Purser has prepared a buyer's guide to personal computers for parents which costs only $1. At that price, they cannot go wrong and they might even thank you for recommending it. Robert Purser, Box 466, El Dorado, CA 95623.

SCOPE

Scholarly Communication:
Online Publishing and Education

A bimonthly newsletter covering all aspects of the computer revolution in academe: data bases, networks, courseware, hardware, publications, meetings. For a sample issue, write to SCOPE, Box C, Queens College, Flushing, NY 11367.

END■

ICCE Organization Members

Alaska Association for Computers in Education (AACE)
Alberta Association for Educational Data Systems
Arizona Computer Users in Education
Association for the Development of Computer-Based Instructional Systems (ADCIS)
Computer Education Group of Victoria
Computer Education Group of Queensland
Computer Education Society of Ireland
Computer Educators of Idaho
Computers, Learners, Users, Educators Association (CLUES)
Computer-Using Educators (California)
Computer-Using Educators of British Columbia (CUEBC)
Computer-Using Educators of Kentucky (CUE-KY)
DIDACOM
Educational Computing Consortium of Ohio
Educational Computing Association of Western Australia
Educational Computing Organization of Ontario (ECOO)
Educational Microcomputer Users Group of Central New York (EMUGCNY)
Educators Interest Group of the San Diego Computer Society
Florida Association of Science Teachers (FAST)
Illinois Association for Educational Data Systems (ILAEDS)
Indiana Computer Educators
Manitoba Association for Educational Data Systems (MAN-AEDS)
Michigan Association for Computer Users in Learning (MACUL)
Minnesota Association for Educational Data Systems
Montana Council for Computers in Education
National Institute for Microcomputer Based Learning (NIMBL)
New Hampshire Association for Computer Education Statewide (NHACES)
New York State Association for Educational Data Systems
New South Wales Computer Education Group
Northwest Council for Computer Education (Oregon, Washington and Northern Idaho)
Oklahoma Educational Computer Users Program (OECUP)
Pennsylvania Learning Resources Association (PLRA)
Quebec Association of Computer Users in Education
Saskatchewan Association for Computers in Education
The Science Teachers' Association of Ontario
Society of Data Educators (SDE)
Society for Microcomputing in Life & Education (SMILE)
Texas Computer Education Association
The Utah Council for Computers in Education
Wyoming Educational Computing Council
Young People's LOGO Association (YPLA)

advertisement

the floppy copy you've been looking for

Across the nation, Data Command microcomputer programs are winning the applause of teachers and students alike! They zero in on crucial reading and math skills, constantly challenging pupils toward mastery with color, excitement, success!

Data Command programs feature:

- **Innovative activities with animated graphics**
- **Carefully controlled vocabulary levels**
- **Instant summarization of pupil results**
- **Concise teacher support materials**
- **Fast, efficient implementation**

Some typical comments:

"Your programs are some of the best I've ever seen."
Carolyn Rutledge, Reading Director Eagle Elem. School, Van Horn, TX

"You hooked us on both" (of the reading programs we previewed.)
Judy Henry, LLC Director Rhodes Elem. School, River Grove, IL

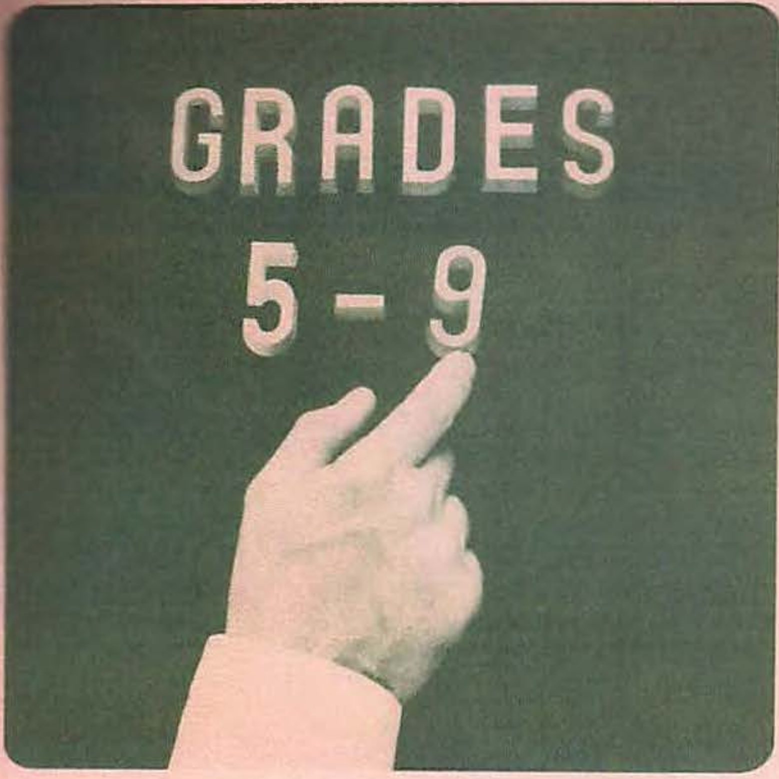
data command
"The workhorse software for microcomputers"

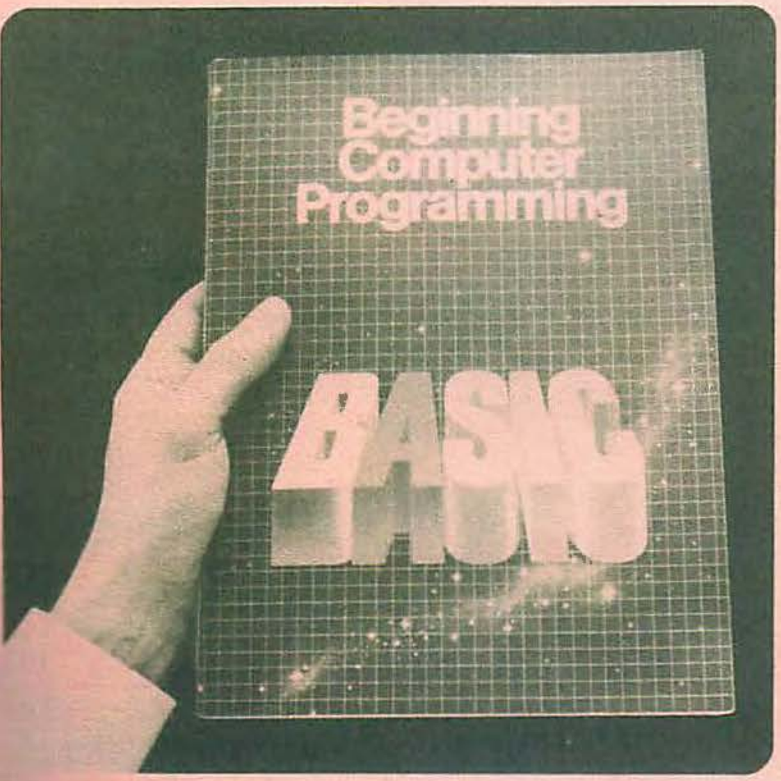
To find out more about the "Floppy Copy" you've been looking for, mail the coupon to Data Command, P.O. Box 548, Kankakee, IL 60901.

Name ______________________
School ______________________
School Address ______________________
City ____________ State ________ Zip ________ CT

Data Command programs are now available for
APPLE TRS-80 COMMODORE

PROGRAMMING AND AWARENESS FOR...





TEACHER-FRIENDLY, TOO!

Now, with this exciting new text, it's easy to teach BASIC programming and build computer awareness with middle and junior high school students. **NOT** a manual from a computer company, this textbook was specially designed for classroom use.

- Easy to teach.
- Compatible with all major micro systems currently used in schools.
- A practical hands-on approach to programming.
- Stresses computer literacy.
- A complete Teacher's Guide gives you the flexibility to plan short units or a full semester course. Blackline worksheets and the supplemental diskette (which includes programs and answers to exercises) lighten your workload.

REMEMBER: For grades 9-12 we offer "Computer Programming in the Basic Language"

For more information: Call Toll Free
1-800-526-4799
In NJ **1-800-932-0117**

Allyn and Bacon
Dept. 25 / 7 Wells Avenue / Newton, MA 02159

YES I'd like to know more! Please send more information on

☐ Beginning Computer Programming: BASIC (grades 5-9)
☐ Computer Programming in the BASIC Language (grades 9-12)

Name ____________________
School ____________________
School Address ____________________
City ____________________
State __________ ZIP __________
Phone ____________________