



LOGO EXCHANGE

Journal
of the ISTE
Special Interest Group
for Logo-Using
Educators

Spring 1995

Volume 13 Number 3

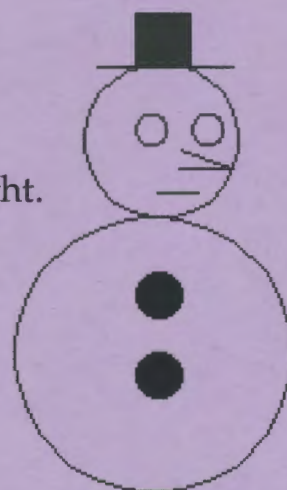
In this issue:... *It's Winter: Let's Build A Snowman*

Frost

Fun in the snow as we get
Red, red noses and we go
Over hills and hilltops in our
Oh! What fun as
The temperature drops.
—Allison Taurence

Snow

Snow is sparkling white.
No, please don't eat it.
Oh! It's so wonderfully bright.
What a sight!
—Shawn Hughes



Also—

***The Grammar Minefield—
Survival Techniques for Students***

***Control-Key Events:
Exploration and Inspiration***

What's New with Logo Plus 2.0 (for the Mac)!

International Society for Technology in Education



Editorial Publisher

International Society for Technology in Education

Editor-in-Chief

Sharon Yoder

Associate Editor

Ron Renschler

Copy Editors

Jennifer Lindsey
Corinne Tan

Editorial Assistant

Anne Moser-Kornfeld

Founding Editor

Tom Lough

International Editor

Dennis Harper

Contributing Editors

Eadie Adamson
Gina Bull
Glen Bull
Doug Clements
Sandy Dawson
Dorothy Fitch
Mark Horney
Robert Macdonald

Production

Kerry Lutz

Editorial Coordinator

Jodie Rogers

Director of Advertising Services

Lynda Ferguson

Marketing Director

Martin G. Boyesen

Submission of Manuscripts

Logo Exchange is published quarterly by the International Society for Technology in Education Special Interest Group for Logo-Using Educators. *Logo Exchange* solicits articles on all aspects of Logo use in education. Articles appropriate to the International column should be submitted directly to Dennis Harper.

Manuscripts should be sent by surface mail on a 3.5" disk (where possible). Preferred format is Microsoft Word for the Macintosh. ASCII files in either Macintosh or DOS format are also welcome. Submissions may be made by electronic mail as well. Where possible, graphics should also be submitted electronically. Please include electronic copy, either on disk (preferred) or by electronic mail, with any paper submissions. Paper submissions alone will NOT be accepted.

Send surface mail to:

Sharon Yoder
170 Education, DLIL
University of Oregon
Eugene, OR 97403

Send electronic mail to:

Internet: syoder@oregon.uoregon.edu

Deadlines

To be considered for publication, manuscripts must be received by the dates indicated below.

Volume 14, Number 3	July 1, 1995
Volume 14, Number 4	Oct. 1, 1995
Volume 15, Number 1	Feb. 1, 1996
Volume 15, Number 2	Apr. 1, 1996

ISTE Board of Directors 1994-95

M.G. (Peggy) Kelly, President
California State University - San Marcos
David Moursund, Executive Officer
Dennis L. Bybee, Associate Executive Officer

Executive Board Members

David Brittain, President-Elect
Florida Department of Education (FL)
Lajeane Thomas, Past President
Louisiana Tech University (LA)
Connie Stout, Secretary
Texas Education Network (TX)
Barry Pitsch - Treasurer
Heartland AEA #11 (IA)
Don Knezek
Education Service Center, Region 20 (TX)

Board Members

Shelia Cory
Terrie Gray
Terry Killion
Paul O'Driscoll
Lynne Schrum
Carla Schutte
Gwen Solomon

Ex-Officio Board Members

Roy Bhagaloo, DP/MIS Special Appointment
John Cradler, Legislative Action and Policy
Kathy Edwards, Minority Affairs
Nolan Estes, International Initiatives
Kathleen Hurley, Industry Representative
Marco Murray-Lasso, Director of Developing Country Initiatives
C. Dianne Martin, Legislative Action and Policy
Alfonso Ramirez Ortega, Director of Latin American Initiatives
Paul Resta, International Initiatives
Richard Alan Smith, Research Representative

Logo Exchange is published quarterly by the International Society for Technology in Education (ISTE), 1787 Agate Street, Eugene, OR 97403-1923, USA; 800/336-5191. This publication was produced using Aldus PageMaker®.

Individual ISTE Members may join SIG/Logo for \$24.00. Dues include a subscription to *Logo Exchange*. Add \$10 for mailing outside the USA. Send membership dues to ISTE. Add \$4.00 for processing if payment does not accompany your dues. VISA, Mastercard, and Discover accepted.

Advertising space in *Logo Exchange* is limited. Please contact ISTE's director of advertising services for space availability and details.

Logo Exchange solicits articles on all topics of interest to Logo-using educators. Submission guidelines can be obtained by contacting the editor. Opinions expressed in this publication are those of the authors and do not necessarily represent or reflect the official policy of ISTE.

© 1995 ISTE. All articles are copyright of ISTE unless otherwise specified. Reprint permission for nonprofit educational use can be obtained for a nominal charge through the Copyright Clearance Center, 27 Congress St., Salem, MA 01970; 508/744-3350; FAX 508/741-2318. ISTE members may apply directly to the ISTE office for free reprint permission.

POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 1787 Agate St., Eugene, OR 97403-1923. Second-class postage paid at Eugene OR. USPS# 660-130. ISTE is a nonprofit organization with its main offices housed at the University of Oregon. ISSN# 0888-6970

Contents

From the Editor

The Logo Generation	Sharon Yoder	2
---------------------------	--------------	---

Quarterly Quantum

To Gather Again	Tom Lough	4
-----------------------	-----------	---

The Grammar Minefield—Survival Techniques for Students	Christine A. Johaneck	5
--	-----------------------	---

Beginner's Corner

Logo Chart and Graphs	Dorothy Fitch	10
-----------------------------	---------------	----

Control-Key Events: Exploration and Inspiration	John Gough	15
---	------------	----

Musings...

It's Winter: Let's Build A Snowman	Robert Macdonald	18
--	------------------	----

What's New With Logo PLUS 2.0 (for the Mac)!	Dorothy Fitch	26
--	---------------	----

Meet LogoWriter's Four Turtles	John Gough	29
--------------------------------------	------------	----

Windows on Logo

Animation	Glen L. Bull, Gina L. Bull, and Todd Kent	34
-----------------	---	----

Logo: Search and Research

Looking Back and Looking Forward	Douglas H. Clements and Julie Sarama	38
--	--------------------------------------	----

What a Change!

This is my seventh year teaching at the University of Oregon. In that time there have been enormous changes in our College of Education. We have gone from a thriving Computer in Education graduate program, to having our entire section of the College of Education closed. We have seen continued budget cuts, even after the initial round of cuts that closed our program. But, contrary to popular opinion, we are not dead! In the last couple of years—under the leadership of a hard-working, dynamic dean—we have begun to rebuild. First, an undergraduate program in education was put in place. That program began this fall. Now efforts are being made to consolidate, restructure, and rebuild our graduate programs. Perhaps in another year or so we'll be back to offering a new, improved degree in Technology in Education. We can hope.

Like many public universities, we are becoming increasingly tuition driven. Tuition driven means having a lot of students, or SCR. (student credit hours in university jargon.) Lots of students means undergraduates, thus the decision to begin our rebuilding at the undergraduate level. Our new undergraduate major has three strands: technology, teacher education, and children and families. Every student entering the program must take a couple of pre-education majors, one of which is a technology course I am currently teaching, which brings me to the focus of this editorial.

The other day as I was looking out at my sea of 70 students, mostly freshmen, I realized that these 18- and 19-year-olds are the kids who were in elementary school during the logo-will-change-education-for-the-better movement. These are the kids who should show the mark of any effect that Logo may have had on their thinking, problem-solving skills, and relationship with technology. These kids are the "Logo Generation."

The Logo Generation

When students enter my class, I ask for their background in technology—what machines they have used, what software they have used, how they use a computer personally, and how they feel about technology. To date I have interacted with about 200 of these students. Only one or two of them have even mentioned Logo as part of their experience. But even if they didn't mention Logo I expected them to have a comfort

The Logo Generation

by Sharon Yoder

level with technology that exceeds previous generations of students.

Much to my surprise, the majority of these students express a tremendous anxiety about technology. They often refer to "doing computers" in junior high or high school, but they also are certain they "can't do computers." They take my course because it is required or because they know they will have to deal with technology in their future. To my amazement, this fear of technology is compounded by a fundamental belief that technology is bad, that technology will take over their lives, divide people, decrease our ability to communicate—in general, to make our world a worse place to live.

Do these young people really know nothing about technology? In fact, they are not as ignorant as they describe themselves to be. The definition of "know" has evolved over the past five or six years from *really* knowing nothing to knowing how to use a word processor to type papers. So, indeed, there has been progress albeit a small amount when you consider what "I can use a word processor" really means.

Another QWERTY Phenomenon

As far as I can tell, I've encountered another QWERTY phenomenon. Of the students in my classes who own a computer, more than half have an electronic typewriter/word processor. They come to college with their shiny new machine, convinced that is all the "compute power" they will ever need. They have no concept that desktop publishing ever happened. They don't know that there is an innate difference between using a computer to "publish" a paper and using a typewriter to put text on a page. Apparently their computer experience was not rich enough for them to realize that all they could ever need in life is the technology with the ability to produce text, and perhaps technology to play games. Somehow these kids didn't evolve in their computer knowledge as they were growing up.

We have all watched Logo grow up over the years. Logo moved with the waves in computer education. We've seen Logo grow from separate text and graphics environments, to products that integrated both. We've seen word processors built into Logo environments. We've seen telecommunications versions of Logo. We've seen hypermedia versions of Logo.

Logo as a QWERTY Phenomenon

What happened here? Were my students' school computing experiences so limited that they never moved beyond keyboarding and some simple computer-assisted instruction? If they did encounter Logo, did they never move beyond Apple IIs, the original Terrapin or Apple Logo, or AppleWorks?

Has Logo itself become a QWERTY phenomenon? I encounter a frightening number of teachers who "do Logo" as an ingrained bit of their curriculum. Often these educators are using what you and I would call an "ancient" version of Logo and are unaware of the changes in Logo over the past 10 years. These educators do Logo because it is the "thing to do," not because they understand its power.

It is also a bit frightening to see the Computers in Education texts that come across my desk. They always seem to have the requisite chapter on programming in BASIC and programming in Logo. It seems to be that a "Logo chapter" entirely misses the point of what Logo is all about.

Even though I'm sure we are all glad that Logo made it, using Logo in a mindless manner seems such a waste. Perhaps my students are simply the victim of the institutionalization of computing in education. Perhaps "doing computers" has become much like "doing math" or "doing spelling"—another chunk of curriculum to be gotten through.

Hmmmm ...

Like many of the editorials I write, this one took on a life of its own. I started with the goal of sharing my insights about my students. But before I knew it I found myself once again discouraged by the rate of progress in technology education. My computers are extensions of myself. They enhance and enlarge everything I do. They are good friends to me, both at home and at the office—as I sit comfortably on my bed, portable computer on my lap, writing this editorial. That young people are afraid of this powerful technology is almost unimaginable to me.

Remember when you first discovered Logo? Remember your excitement? Remember how delighted you were with your successes? Well, I see the same thing happen with my university students as I move them from frightened novices to creating their own newsletter at the end of the term. The majority of them gain confidence and have tremendous ownership of their work by the end of the term. I'm sure that if I had time in this course to introduce Logo, they would be just as excited about that as they are about, say, learning to use graphics tools. Perhaps in a future course....

Maybe those of us who fell in love with Logo have forgotten what brought us to Logo in the first place. Most often, our move to a commitment to Logo was a

risk, a commitment to something better, a brush with the future. We were visionaries for that moment, when we stepped out of the conventional curriculum to try something new. But then we got stuck. We forgot to teach our students that the future of technology will be more exciting than anyone can imagine. We forgot to prepare them to look for and expect more of the technology than they saw in our classes.

Sharon Yoder
DLIL, Education 170C
1215 University of Oregon
Eugene, OR 97403-1215
syoder@oregon.uoregon.edu

My Favorite Videotapes

Below is a list of videotapes I use with classes. I have indicated how long each is and a bit about the content. Some of the videotapes are appropriate for teachers, others for secondary students, and some of them may work with elementary students. Perhaps you can use some of these to spark a sense of vision in your students.

Connections: AT's Vision of the Future. (14 minutes). This tape explores a wide range of uses of telecommunications technology by following several days in the lives of a family at work and at play. (Source: AT & T)

Knowledge Navigator. (5 minutes). This tape shows a College Professor interacting with his personal computer to access information, communicate with colleagues, and make appointments. (Source: Apple Computer)

Ulysses. (30 minutes). This videotape shows a speech given before the release of the IBM product *Ulysses*. It demonstrates using powerful multimedia materials to teach difficult concepts. (Source: IBM)

Chapter One. (12 minutes). This videotape shows a futuristic computer system in the home of an author. Her children are shown studying using technology and interacting with their teacher at the same time. (Source: Apple Computer)

The Machine That Changed the World. (5 hours). This is a series of videotapes that shows the history of technology from mainframes through the coming of the micro and into the future. (Source: Public Television)



To Gather Again

by Tom Lough

Recently, I attended a reunion of my college class. Of the 565 original members, 45% showed up for two and one-half days of fellowship and reminiscence. I was amazed at how much I had forgotten. Classmates whose names I could not remember at first recounted hilarious adventures and events, some of which I remembered with no prompting, and others of which I would rather forget completely!

The entire reunion event was wonderful, providing an opportunity to look at the past, to catch up with the present, and to project to the near future. I was impressed by how much we learned from and about each other in such a short time.

My thoughts turned down two different roads as I reflected on my reunion experience. First, I thought about my Logo students. Where are they now? What are they doing? Do they remember anything about our Logo activities? Are they still thinking procedurally?

It would be wonderful to be able to arrange a reunion of my Logo students. Unfortunately, time, circumstances, and finances have combined into a barrier that can withstand even my most enthusiastic advances. But what about you? Have you ever thought about a reunion of your Logo students?

Chances are that your Logo students from last year or the year before are not too far away. Would you be able to invite them back to your classroom for a special 40-minute reunion? They might enjoy getting together again to recount their Logo projects. Maybe your present students would enjoy meeting them and hearing about their Logo adventures of years gone by.

Next, I thought about Logo teachers and practitioners. One of the first gatherings I can remember was the fabulous Logo '84 conference at the Massachusetts Institute of Technology. This was followed by Logo '85 and Logo '86. Pepperdine University organized a West Coast Logo Conference series and the University of Virginia (and later ECCO) hosted an East Coast Logo Conference. There was a lot of other congregating activity as well. For example, Dan and Molly Watt held summer Logo institutes in the northeastern United States, and Geraldine Kozberg organized a continuing series of Logo institutes in St. Paul, MN. Gary Stager staged Logo events in New Jersey.

At each of these meetings, the reunion spirit was alive and well. Fellowship flourished and friendships

were forged. In several instances, students of the reunionees were introduced to each other through mutual class projects during the following school years.

Then came a period of decreased activity. Few reunion opportunities appeared on the national or regional scene. The informal Birds-of-a-Feather session for Logo, sponsored by Logo Exchange and ISTE at the National Educational Computing Conference, was one of the few national level focal points for several years.

Last year, on the day before the National Educational Computing Conference in Boston, a Logo symposium (Logosium) was organized by Marion Rosen, with assistance and support from ISTE, MIT, and the Logo Foundation. The spirit of reunion blended with a warm welcome for everyone, regardless of Logo experience. It was good to gather again.

This summer, Logosium '95 is scheduled for Friday, June 16, as a preconference activity at NECC '95 in Baltimore. Sponsored by the ISTE SIGLogo and the Logo Foundation, Logosium '95 offers everyone the opportunity to sample the spirit of reunion in a full day of Logo discussions, sharing sessions, and presentations.

If you are making plans for NECC '95, be sure to include an extra day for Logosium '95. Let's get together again!

Until then,

FD 100!

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

The Grammar Minefield— Survival Techniques for Students

By Christine A. Johaneck

When faced with grammar study, most English students groan, roll their eyes, slouch in their seats, and check the calendar to see when the unit will be over. At least that is the way it was with my seniors. It seemed to them that grammar—specifically the identification of parts of speech—was an educational minefield through which they are required to annually march, trudge, or crawl. As in all minefields, the lucky ones somehow survive; the unlucky ones never make it out.

Understanding grammar and identifying parts of speech is *not* a matter of luck. Early in their language careers most students learn the clues that help them categorize words. Often, these clues are learned intuitively. Students who struggle in language activities may not have learned these rules naturally. It then becomes the teacher's responsibility to "teach" these clues or help students discover them.

You may be thinking, "Clues? What clues? I don't know any clues to help me determine parts of speech." Of course you do! To refresh your memory, here are a few:

1. An article (a, an, the) tells you a noun is close by. Maybe even the next word.
2. If it ends in -ism, -ability, or -tion, it is probably a noun.
3. If it ends in -ful, -able, or -ous, it is probably an adjective.

The list goes on, but you have the idea.

Once students learn these clues, they must practice them. Using Logo, the teacher can provide the "clues" and the computer can generate the "practice."

The GRAMMAR.DRILL program is really an expanded set of PICK procedures that produces sometimes wacky sentences that grammar students can parse. Sentences can be simple, or complicated and involved, depending on the experience or ability of the students and the objectives of the teacher. The program can be augmented with new classes of words and new sentence constructions when new concepts are introduced. Exceptionally sophisticated sentences (such as those containing gerunds, dependent clauses, adjective and adverb phrases, etc.) have not been included in the sentences illustrated here, but they could certainly be added if the unit objectives and students' abilities required them.

Prior to working with the computer, students should learn specific clues to help them identify particular parts of speech. For example, in addition to reviewing easy-to-identify nouns, the teacher might spend one or more class periods examining the placement of nouns in sentences, common suffixes, the use of articles, and so on. Clues similar to those for word placement and suffixes can be discovered for adjectives and adverbs. The clues examined in class might be posted as reminders to help students when they parse sentences. Note, however, that some classes of words—articles, conjunctions, prepositions, etc.—do not lend themselves to the clue technique and are best memorized or recognized.

Following discussion of the discovery and use of clues, students begin working at the computer either independently or in small groups. GRAMMAR.DRILL begins with the following directions on the computer screen:

```
WELCOME TO GRAMMAR DRILL! EVERY TIME
YOU TYPE THE DRILL COMMAND, A NEW
SENTENCE WILL BE PRINTED ON THE
SCREEN.
SOME OF THE SENTENCES WILL MAKE
SENSE,
BUT SOME OF THEM WILL BE REALLY
CRAZY.
THAT'S O.K. !
YOUR JOB IS TO DETERMINE THE PARTS
OF
SPEECH USED IN EACH SENTENCE. THE
SENTENCES USE ONLY THOSE PARTS OF
SPEECH
YOU HAVE WORKED ON IN CLASS.
READY? IF SO, WAIT UNTIL THE SCREEN
CLEARS AND THEN TYPE THE COMMAND
"DRILL."
IF YOU HAVE A QUESTION, ASK YOUR
TEACHER
NOW.
```

Each time the DRILL command is entered, one of several sentence constructions containing randomly chosen words appears. Most of the sentences will not make sense.

QUIET MICHELLE AND PAULA PICK?
 IVORY FEELS QUIET.
 MEANINGLESS CHAIRS BLAME.
 BREATHLESS TURTLE CRINGES.
 FRED SOUNDS CLASSLIKE.
 BERNICE OOZES CONTEMPTUOUSLY.
 YOUR RED MELON EXCITED RELENTLESSLY!

It is probably better if the sentences *do not* make sense, because the students must then rely on the clues and sentence structure itself to determine the parts of speech. A variety of sentence constructions stored on a TOOLS page on disk ensures the students will not memorize the construction of one or two sentences that appear repeatedly.

The GRAMMAR.DRILL exercise can easily be augmented to provide more sophisticated challenges. For example, the VERB procedure contains the following list:

```
TO VERB
  OUTPUT PICK [ABANDON RELATE
    SUPERVISE EXCEED CREATE AFFORD
    CONDUCT PLAY READ DISCERN ACCEPT
    MOMERF ZAFURSTIZE CARTIBULATE
    WOPLOC]
END
```

Nonsense verbs have been added to the list of verbs. Another procedure, NEWNOUN—given at the end of this article—creates nonsense nouns from verbs. The result is sentences like these:

HER PEACH DREEZLE EXUDES
 EXHAUSTEDLY!
 JOHN, PAUL, GEORGE AND RINGO APPEAR
 GARFLESS?
 WOPLOCNESS WAS HOPELESS!
 WOPLOCTION APPEARS MEANINGLESS!

Previously unsuccessful students express such a feeling of accomplishment when they are able to identify the parts of speech exemplified by “words” that are not even words!

The teacher can assign or expect a variety of products in connection with the use of GRAMMAR.DRILL. If students work together, the ensuing discussion of grammar and grammar-related questions among colleagues may be the ultimate product desired by the teacher. The teacher can also require that random sentences be printed and labeled appropriately by the students. Because there are theoretically thousands of possible sentences that can be generated, there is little potential for the “sharing” of answers, if such activity is frowned upon. The program could also be used to generate individual grammar quizzes for students, their work either printed for the teacher or saved on disk

for later examination. Specific vocabulary words presented in class can also be added to the appropriate lists for further study. The teacher may ask students to suggest other words that fit the appropriate categories, or to generate their own constructions and categories of words.

My students will tell you that grammar and parts of speech are *still* a minefield. They still have to get through it, but they will be marching, not crawling, this time, thanks to an accurate map (the “clues”) and adequate practice.

A sample list of parts of speech and sentence constructions are given in the following procedures.

```
TO STARTUP
HT
RECYCLE
PRINT [WELCOME TO GRAMMAR DRILL!
  EVERY TIME YOU TYPE THE DRILL
  COMMAND, A NEW SENTENCE WILL BE
  PRINTED ON THE SCREEN. SOME OF
  THE SENTENCES WILL MAKE SENSE,
  BUT SOME OF THEM WILL BE REALLY
  CRAZY. THAT'S O.K.!]
PRINT [ ]
WAIT 40
PRINT [YOUR JOB IS TO DETERMINE THE
  PARTS OF SPEECH USED IN EACH
  SENTENCE. THE SENTENCES USE ONLY
  THOSE PARTS OF SPEECH YOU HAVE
  WORKED ON IN CLASS.]
PRINT [ ]
WAIT 40
PRINT [READY? IF SO, WAIT UNTIL THE
  SCREEN CLEARS AND THEN TYPE THE
  COMMAND "DRILL." IF YOU HAVE A
  QUESTION, ASK YOUR TEACHER NOW.]
RECYCLE
WAIT 100
CT
END

TO PICK :OBJECT
  OUTPUT ITEM (1 + RANDOM COUNT
    :OBJECT) :OBJECT
END

TO VERB
  OUTPUT PICK [ABANDON RELATE
    SUPERVISE EXCEED CREATE AFFORD
    CONDUCT PLAY READ DISCERN ACCEPT
    GARPHINKLE MOMERF ZAFURSTIZE
    CARTIBULATE WOPLOC]
END
```


TO SVERB
 OUTPUT PICK [CRIES SLITHERS CREATES
 EXCITES UNVEILS DISCOVERS IDENTIFIES
 PLUCKS THWARTS EXUDES OOZES DESTROYS
 INHIBITS DONATES ENTERTAINS SCAMPERS
 WHISTLES WHITTLES CRINGES
 OBLITERATES ENCLOSSES POLLUTES] END

TO PLVERB
 OUTPUT PICK [CRY SLITHER CREATE
 EXCITE UNVEIL DISCOVER IDENTIFY
 EAT UNLOAD PICK BLAME ABSOLVE
 CATEGORIZE PLUCK THWART EXUDE
 OOOZE SCAMPER WHISTLE WHITTLE
 CRINGE OBLITERATE ENCLOSE
 POLLUTE]

END

TO NOUNIZE
 OUTPUT PICK [ABILITY ITY TION ISM
 NESS MENT ICE]
 END

TO SLINKINGVERB
 OUTPUT PICK [IS WAS SEEMS FEELS
 APPEARS SOUNDS [WILL BE]]
 END

TO PLINKINGVERB
 OUTPUT PICK [ARE WERE SEEM FELL
 APPEAR SOUND [WILL BE]]
 END

TO ADJECTIVE
 OUTPUT PICK [QUICK HONEST QUITE LOUD
 SLOW SILENT EXHAUSTED EXCITED
 COLORFUL HOPELESS MEANINGLESS
 UNREQUITED UNREMITTING RELENTLESS
 BREATHLESS CONTREMP TUOUS CRUSHING
 UNCOMPROMISING]
 END

TO ADJECTIVEIZE
 OUTPUT PICK [EN OUS FUL AL LESS ATE
 LIKE]
 END

TO ADVERBIZE
 OUTPUT PICK [LY]
 END

TO HUE
 OUTPUT PICK [RED ORANGE YELLOW GREEN
 BLUE INDIGO WHITE BLACK TEAL
 PEACH BERMILLION [BURNT SIENNA]
 BEIGE CHARCOAL IVORY SCARLET]
 END

TO ADVERB
 OUTPUT WORD ADJECTIVE ADVERBIZE
 END

TO SNOUN
 OUTPUT PICK [CRETAN FURNITURE CHILD
 DAISY TWIT TURTLE TELEVISION
 [BILL COSBY] [CHOCOLATE CHIP
 COOKIE] LAMP CLASS MELON KUMQUAT
 GARF THORK CONHURIT DREEZLE]
 END

TO PLNOUN
 OUTPUT PICK [CRETANS CHAIRS TWITS
 [THE HUXTABLES] PUPPIES ZEBRAS
 CRAYONS [MICHELLE AND PAULA] [THE
 KIDS IN THIS CLASS] [THE PEOPLE
 ON "AMERICA'S FUNNIEST VIDEOS"]
 [RAMON AND HIS FAMILY]]
 END

TO DETERMINER
 OUTPUT PICK [A AN THE]
 END

TO SPOSSESSIVE
 OUTPUT PICK [MY YOUR HER HIS IT]
 END

TO PLPOSSISSIVE
 OUTPUT PICK [OUR YOUR THEIR]
 END

TO PREPOSITION
 OUTPUT PICK [ABOUT ABOVE ACROSS
 AFTER AGAINST ALONG AMOUNG AROUND
 AT BEFORE BEHIND BELOW BENEATH
 BESIDE BETWEEN BY DOWN DURING FOR
 FROM IN INTO OF OFF ON OUT OVER
 TO TOWARD UNDER UNTIL UP UPON
 WITH WITHOUT]
 END

TO PICK :LIST
 OUTPUT ITEM (1 + RANDOM COUNT :LIST)
 :LIST
 END

TO SPROPERNOUN
 OUTPUT PICK [[GRANDMA MOSES] LILLIAN
 BERNICE [F. SCOTT FITZGERALD]
 [PRESIDENT BUSH] [MIGHTY MOUSE]
 [AUNT MARIA] YOLANDA PETER RYAN
 MICHELLE PAULA VIC JENNIFER RAMON
 DEVIN ED CONNIE BRIDGET A.J.
 [MICHAEL JORDAN] [MICHAEL
 JACKSON] [MIKHAIL GORBACHEV]
 [QUEEN ELIZABETH] SOPHOCLES
 HERCULES [DUDLEY DORIGHT] [JON
 BON JOVI] SPOT ROVER HAMLET
 GERTRUDE KITTY JIM FRED ETHEL
 TYLER CLINT [EARL "THE PEARL"
 MONROE] [TOM OSBORNE]]

END

TO PLPROPERNOUN
 OUTPUT PICK [[THE SIMPSONS] [PAULA
 AND MICHELLE] [MIKE AND JUDY]
 [JOHN, PAUL, GEORGE, AND RINGO]
 [THE HUXTABLES] [GEORGE AND
 BARBARA] [DAN AND MARILYN]
 DOROTHY, TOTO, THE SCARECROW, THE
 TINMAN, AND THE COWARDLY LION]
 [THE BEATLES] [THE ROLLING
 STONES] [THE NEW KIDS ON THE
 BLOCK] [PRINCE CHARLES AND
 PRINCESS DIANA] [BOB HOPE AND
 BING CROSBY] [RYAN AND JIM]
 [CLINT AND TYLER] [BO JACKSON AND
 MICHEAL JACKSON]]

END

TO NEWNOUN
 OUTPUT WORD VERB NOUNIZE
 END

TO NEWADJECTIVE
 OUTPUT WORD SNOUNT ADJECTIVEIZE
 END

TO PUNCTUATION
 OUTPUT PICK [. ! ?]
 END

TO NEWSSENTENCE1
 OUTPUT (SENTENCE ADJECTIVE NEWNOUN
 ADVERB WORD SVERB PUNCTUATION)
 END

TO NEWSSENTENCE2
 OUTPUT (SENTENCE ADJECTIVE PLNOUN
 WORD PLVERB PUNCTUATION)
 END

TO NEWSSENTENCE3
 OUPUT (SENTENCE ADJECTIVE SNOUN WORD
 SVERB PUNCTUATION)
 END

TO NEWSSENTENCE4
 OUPUT (SENTENCE SPOSSIVE SNOUN
 ADVERB SVERB PREPOSITION
 DETERMINER HUE WORD NEWNOUN
 PUNCTUATION)
 END

TO NEWSSENTENCE5
 OUTPUT (SENTENCE NEWNOUN
 SLINKINGVERB WORD ADJECTIVE
 PUNCTUATION)
 END

TO NEWSSENTENCE6
 OUTPUT (SENTENCE PLNOUN PREPOSITION
 NEWNOUN PLBERB ADVERB PREPOSITION
 DETERMINER WORD SNOUN PUNTUATION)
 END

TONESSENTENCE7
 OUTPUT (SENTENCE DETERMINER NEWNOUN
 SVERB PREPOSITION WORD PLNOUN
 PUNCTUATION)
 END

TO NEWSSENTENCE8
 OUTPUT (SENTENCE NEWADJECTIVE PLNOUN
 PLVERB WORD ADVERB PUNCTUATION)
 END

TO NEWSSENTENCE9
 OUTPUT (SENTENCE SPROPERNOUN SVERB
 WORD ADVERB PUNCTUATION)
 END

TO NEWSSENTENCE10
 OUTPUT (SENTENCE PLPROPERNOUN PLVERB
 PREPOSITION HUE WORD SNOUN
 PUNCTUATION)
 END

TO NEWSSENTENCE11
 OUTPUT (SENTENCE SPROPERNOUN
 SLINKINGVERB WORD NEWADJECTIVE
 PUNCTUATION)
 END

TO NEWSSENTENCE12
 OUTPUT (SENTENCE PLPROPERNOUN
 PLINKINGVERB WORD NEWADJECTIVE
 PUNCTUATION)
 END

```

TO NEWSSENTENCE13
OUTPUT (SENTENCE HUE SLINKINGVERB
      WORD ADJECTIVE PUNCTUATION)
END

TO NEWSSENTENCE14
OUTPUT (SENTENCE PLPOSSIVE SNOUN
      SVERB PREPOSITION DETERMINER WORD
      PLNOUN PUNCTUATION)
END

TO NEWSSENTENCES
OUTPUT PICK [NEWSSENTENCE1
      NEWSSENTENCE2 NEWSSENTENCE3
      NEWSSENTENCE4 NEWSSENTENCE5
      NEWSSENTENCE6 NEWSSENTENCE7
      NEWSSENTENCE8 NEWSSENTENCE9
      NEWSSENTENCE10 NEWSSENTENCE11
      NEWSSENTENCE12 NEWSSENTENCE13
      NEWSSENTENCE14 NEWSSENTENCE15]
END

TO DRILL
PRINT RUN (SENTENCE (NEWSSENTENCES))
PRINT []
END

```

Christine A. Johanek teaches sophomore and senior English at Daniel J. Gross High School in Omaha, Nebraska. She also coaches varsity cheerleading. In her spare time she enjoys traveling, does cross stitching, and sings in her church choir.

Christine A. Johanek
4921 Vinton Street
Omaha, NE 68106

THE CRYSTAL RAIN FOREST

A Mathematical Learning Adventure

The planet Oglo is in trouble. Its rain forests are being destroyed.
The king has been poisoned. Only YOU can save them!

The Crystal Rain Forest helps students in grades 3-8 use math to learn about the environment. They hunt for clues in the town, then search for the lifesaving magical crystals deep in the rain forest.

On their quest, they face a series of mathematical puzzles and challenges to solve. They give instructions to robots, guide and rotate shapes to mend bridges, navigate a boat, estimate

distances and angles to connect wires, draw shapes to make nets, change box sizes using simple algebra, and so on.

As a result of these carefully sequenced activities, students learn to use the Logo language. *Crystal Logo*, an easy-to-use version, can be run separately from the adventure, and its command names can be modified.

The Crystal Rain Forest, award-winning software from England, is available here as a single user version (\$49.95), as a single version for school use with curriculum materials (\$59.95), and as a building site license (\$250.00).

PC version requires a 286 or better with VGA and a mouse.
Mac version requires System 7, color

TERRAPIN SOFTWARE, INC.
400 RIVERSIDE ST. • PORTLAND, ME 04103
207-878-8200

1-800-972-8200

Logo Charts and Graphs

by Dorothy Fitch

I am a counter and a lister. Are you? As a small child, I remember counting anything and everything. And I have been known to make lists of lists. So here I sit amidst piles of probably useless data. But it might be interesting to make charts and graphs of this information. Let's use Logo to experiment! In this column, we will look at different types of charts and graphs and ways of creating them with Logo.

Making Charts and Graphs

Why make charts and graphs? Take a look at the process of creating them:

- decide what data to use or to collect
- design data-collection experiments
- collect the data
- decide how best to represent the data that is collected
- use the graphing tools to show the data
- analyze the results

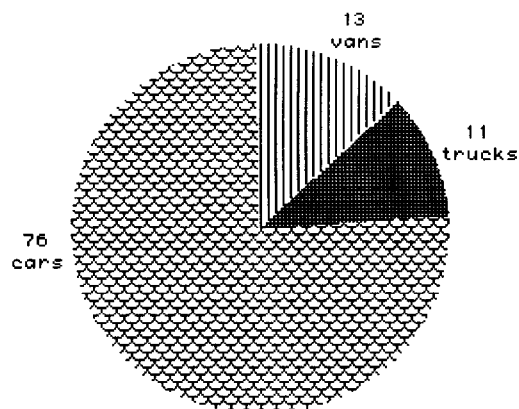
This sounds like a worthwhile set of activities for students of any age!

In the interests of time and space, rather than show the development of each graphing tool, I will simply present the procedures I used. Modify them as you wish to use with the data you collect.

Pie Chart: Distribution of Types of 100 Vehicles

Still a kid at heart, on a long driving trip I usually keep track of the states spotted on license plates. On a recent trip, I decided to collect data to use in this column.

I kept a list of the first 100 unique vehicles we passed (or were passed by) on a trip heading north from Richmond, VA. There were 76 passenger cars (vehicles in which people could ride comfortably either in front or back seats, by my definition), 13 pick-up trucks or small cargo vans, and 11 large trucks. From this data I made a pie chart showing the vehicle types.



Here are the instructions to make this chart using Logo PLUS for the Macintosh. For the chart above, type `PIE` followed by the individual values, as in: `PIE [13 11 76]`


```

TO PIE :LIST
  CLEARGRAPHICS
  SLICES :LIST TOTAL :LIST          ; calls the main procedure
  END                                with the data list and the sum of the numbers in the list
                                    (calculated by TOTAL)

TO SLICES :LIST :SUM
  IF EMPTY? :LIST [STOP]            ; stops the procedure when the list of numbers is empty
  SETPC 4 + RANDOM 10                ; picks a color at random
  SETPP 1 + RANDOM 38                ; picks a pattern at random
  LOCAL "ANGLE                      ; makes the variable ANGLE seen only by this procedure
  MAKE "ANGLE ((FIRST :LIST) / :SUM) * 360 ; computes the angle to turn
  (STAMPARC 100 100 :ANGLE "TRUE)    ; draws a "slice" using the STAMPARC primitive ("TRUE fills
                                    it in)
  RIGHT :ANGLE                      ; turns the turtle for the next slice
  SLICES BUTFIRST :LIST :SUM          ; calls a copy of the same procedure for the next slice, using the next
                                    number in the list and the sum, which doesn't change
  ; adds a list of numbers and reports the sum

TO TOTAL :LIST
  IF EMPTY? :LIST [OUTPUT 0]
  OUTPUT (FIRST :LIST) + (TOTAL BUTFIRST :LIST)
END

```

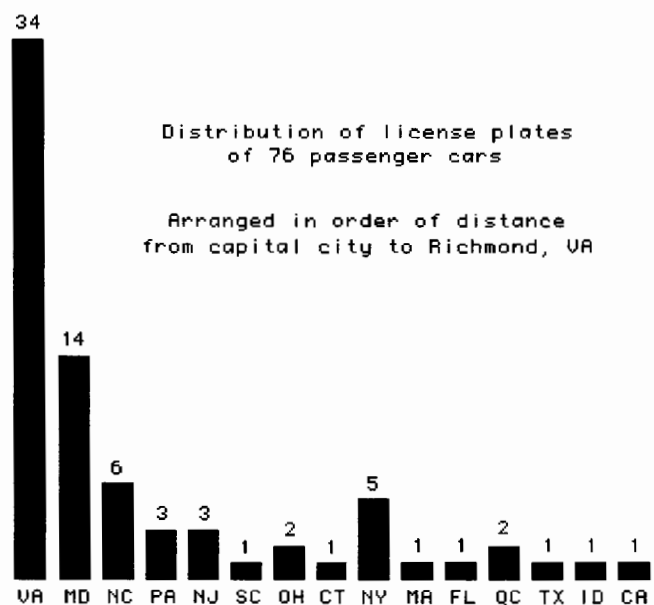
It is not necessary to understand exactly how the TOTAL procedure works. You can use it as if it were a primitive, a "black box" that reports the sum of numbers given in an input list. For example, TOTAL [1 2 3 4 5] reports 15. TOTAL is used here to calculate the angle for each turn. We can find the percent of each type of vehicle by dividing its number by the total number of vehicles. Then we can multiple that number by 360 to determine the angle to turn.

Run the PIE procedure until you get the colors and patterns you desire. Note that as there is no test to see if a particular color or pattern has already been used, adjoining "slices" of the pie could appear in the same color and/or pattern. This might cause a misleading or inaccurate chart to be drawn. The SLICES procedure could be revised to prevent this from happening or could allow you to select the specific patterns and colors you wanted.

I added the text using Logo's Write command. This tool causes a text cursor to appear in the Turtle window, allowing typing anywhere in the graphics window.

Bar Chart: States Represented

Next I decided to use a bar chart to show the distribution of states on the license plates of the 76 passenger cars.



Not surprisingly, most cars were from Virginia, the state in which we were driving. I decided to arrange the states in order of distance from each state's capital to Richmond, VA. (QC is the abbreviation for Canada's province of Quebec.) Some interesting questions arise:



- Is there an unexpectedly high number of cars from New York? Why?
- If trucks were included in the chart, would the results look the same?
- Are more different states likely to be found in a survey taken in Virginia than in other states?
- Was the day and time of year (a Saturday in early December) a factor in the number of states recorded?
- Would another sampling of 100 cars show the same results?

The procedures for the bar chart are as follows:

```
BARChart [34 14 6 3 3 1 2 1 5 1 1 2 1 1 1] 8
```

```
TO BARChart :LIST :FACTOR
  SETXY -150 -150
  CLEAN
  HIDEturtle
  BARS :LIST :FACTOR
END
```

; move turtle for start of chart
; clear the screen without
moving the turtle
; LIST is the raw data;
FACTOR adjusts the size of the bars

```
TO BARS :LIST :FACTOR
  IF EMPTY? :LIST [STOP]
  LOCAL "AMOUNT

  MAKE "AMOUNT :FACTOR * FIRST :LIST
  (STAMPRECT 15 :AMOUNT "TRUE)
  NEXTBAR
  BARS BUTFIRST :LIST :FACTOR
END
```

; stops the procedure when the list of numbers is empty
; makes the variable AMOUNT visible only to the BAR
procedure
; determines the size of the bar
; draws the rectangular bar; "TRUE fills it in
; moves the turtle to prepare for the next bar

```
TO NEXTBAR
  PENUP
  SETX XCOR + 22
  PENDOWN
END
```

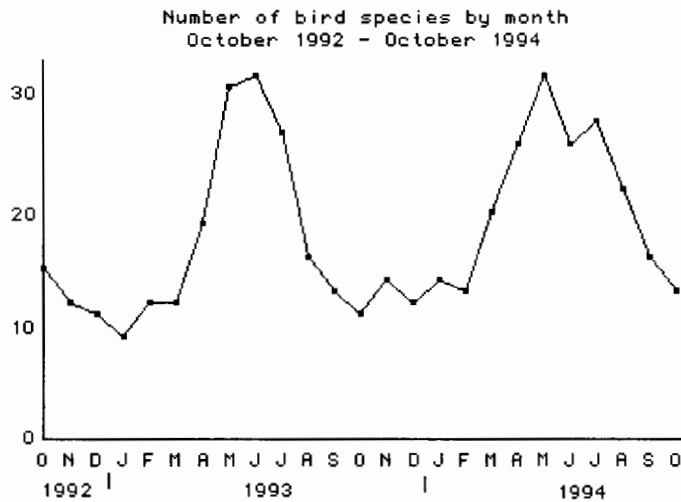
After some experimentation with an initial simpler BARS procedure, I found that I wanted to add the FACTOR variable. This made it easy to adjust the size of the bars proportionately. If my data ranged from 0 to 10, I wouldn't want the turtle to go forward just 3 or 7, but move in larger increments. By using an input of 10 for FACTOR, the turtle would then go forward 30 or 70, making the bars much easier to see. In the previous example, I used a size factor of 8. In the procedure BARS, this factor is multiplied by the data number to move the turtle forward.

After the bar chart was drawn, I typed the title, numbers, and state abbreviations using the Write tool.

Line Graph: It's For the Birds!

Line graphs are perhaps best suited to show a change over time. Because my car data did include time information, I turned to my "backyard bird" lists for this graph.

For several years we have kept a daily record of the bird species observed in our New Hampshire backyard. The following line graph shows the numbers of species seen or heard in each month from October 1992 through October 1994. The number of species ranges from 9 to 32.



Questions that arise from this graph are:

- Why does the line rise and fall over time?
- What pattern emerges when you look at two years' worth of data?
- Would a graph of species seen in your state look the same or different?
- What might account for the jagged lines within the general rises and dips?
- Are any birds seen all year round?

The procedures for the line graph are as follows.
Again the FACTOR variable allows the graph to be resized easily.

```
LINEGRAPH [15 12 11 9 12 12 19 31 32 27 16 13 11 14 12 14 13 20 26 32 26 28 22 16
13] 5
```

```
TO LINEGRAPH :LIST :FACTOR          ; FACTOR
SETXY -160 0                        (5 here) allows the graph to be resized
SETHEADING 0
CLEAN                                ; clears the screen, leaving the turtle where it is
FORWARD 200 BACK 200 RIGHT 90        ; draws the vertical axis
FORWARD 300 BACK 300 LEFT 90        ; draws the horizontal axis
PENUP SETY :FACTOR * FIRST :LIST PENDOWN ; sets the turtle for the first dot
SETPS 3 3 DOT SETPS 1 1             ; makes the pen thicker, draws a dot, resets the pen
LINES BUTFIRST :LIST :FACTOR        ; calls the procedure to draw the rest of the lines

TO LINES :LIST :FACTOR
IF EMPTY? :LIST [STOP]              ; stops when the list of numbers is empty
LINE FIRST :LIST :FACTOR             ; calls the procedure to draw the next line segment
LINES BUTFIRST :LIST :FACTOR        ; recursive call with all but the first number in the list

TO LINE :NUMBER :FACTOR
SETXY (XCOR + 12) (:NUMBER * :FACTOR ; draws a line segment
SETPS 3 3 DOT SETPS 1 1             ; makes the pen thicker, draws a dot, resets the pen
```

In Summary

Try some of these tools yourself! Have your students gather data in an area they are studying or that interests them. Help them create appropriate charts and graphs. Discuss the results.

Happy Logo adventures!

Dorothy Fitch has been director of product development at Terrapin since 1987. A former music educator, she has also directed a computer education classroom for teachers and students and provided inservice training and curriculum development for schools. She is the author of *Logo Data Toolkit* and coauthor of *Kinderlogo*, a single-keystroke Logo curriculum for young learners. At Terrapin, she coordinates software development, edits curriculum materials, writes documentation, and presents sessions at regional and national conferences.

Dorothy Fitch
Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103-1068

CompuServe: 71760,366
Internet: 71760.366@compuserve.com
800/972-8200

A First Course in Programming ... in Terrapin Logo, LogoWriter, and PC Logo

A First Course began as a curriculum for our own classes. Today it is used around the world in hundreds of school districts.

The comment we hear most often is, "These materials were obviously written by classroom teachers."

Secondary as well as elementary teachers have found these materials to be a valuable resource.

This fall will be our eighth year of teaching Logo. We hope all who teach Logo enjoy it as much as we do.

A First Course in Programming is a directed learning environment in structured programming. Its 450 pages emphasize problem solving strategies, critical thinking skills and solid principles of computer science.

This is a complete curriculum for a semester course in programming. It includes student activity sheets, teacher lesson plans, tests, quizzes, assignments, and sample solutions for all student assignments (hard and soft copy!). Only \$295 for a building site license. For information or orders contact:

Logo Curriculum Publishers
1510 Allegheny Drive
Colorado Springs, CO 80919
1-800-348-5646 (FIT LOGO)

Curriculum written BY teachers FOR teachers!

Control-Key Events: Exploration and Inspiration

by John Gough

Introduction

I must confess that I do not find standard Logo references at all helpful in learning what I can do with some of the more obscure primitives of Logo. Consequently, a lot of my time as a teacher of Logo is spent trying to understand the cryptic explanations of what a certain primitive does. Then, hoping that I have understood what a primitive is meant to do, I try to find something I think my students may find easy to understand and worth doing with that primitive. This is easy enough with the visual and auditory effects available in various versions of Logo because you can see or hear what happens. Making cartoons and songs has immediate appeal for my students. But some primitives are far from obvious, and many of the guides do not help.

Consider the **when** command. According to the *LogoWriter* Reference Guide,

```
when letter listtorun
```

programs a Control-key event. Whenever the Control key and the letter key are pressed, *listtorun* is run. *Listtorun* will be run immediately when the key combination is pressed, regardless of whether another program is running. Only 10 keys can be used as control keys: NOPQRVWXYZ. Control-key events are only saved on a page if the when instruction is put inside a procedure.

Then we are given this example to illustrate the power of this primitive:

```
when "z [print "hello]
```

Press the Control-Z key combination and you see

```
hello
```

It is true that this example actually does what the *LogoWriter* Reference Guide says it does. But it fails to demonstrate what on earth might be the point of doing such a thing as getting "hello" to appear if you press Control-Z!

Of course, in the general computer environment we are used to special key combinations such as Apple-S or Control-S to save, or Control-Q to Quit.

But I remained baffled. Why would we want to use Control-key events?

A Digression

As I was looking through the manual I became sidetracked by exploring the primitives **char** and **ascii**. Every character that can be typed on a computer keyboard is represented by a corresponding ASCII number. That is, computers use a special numerical code to represent letters, numbers, and other characters. According to the *LogoWriter* manual, the primitive **char number** returns the character that corresponds to the ASCII numerical value given as input. Similarly, in an inverse way, **ascii "n** reports the ASCII number that corresponds to the character **n**. ASCII stands for American Standard Code For Information Interchange.

Try the following to see how this works

```
print ascii "@"
```

or

```
print char 45.
```

Intrigued that **char** could reveal hidden characters that did not ordinarily exist on my keyboard, I developed the following procedure and explored all the Macintosh **char** numbers from 0 to 256.

```
to check.char :number
ht
clean
ct
print :number
label char :number
end
```

In this way I discovered all the non-English alphabet letters or characters the computer could print. Later I discovered that these were not necessarily the same (number and character) on Apple *LogoWriter*, Macintosh *LogoWriter*, or IBM *LogoWriter*. Obviously different systems work with different versions of ASCII codes.

Then I tried to do the same thing in reverse.



```

to check.ascii
make "input readchar
(print:input ascii :input)
end

```

Later I made this recursive. I discovered that certain keys had no ASCII code (such as the Open-Apple), but other keys, such as the Tab and Escape keys did. Which keys in your system have no ASCII code?

At this stage I realized that **readchar** could be used to identify keys that were not alphanumeric. For beginning students, I could design a keyboard-familiarizing procedure that would ask them to press special keys such as the Escape key, and reward them when they were successful. The following is a fragment of such a procedure.

```

to key.teach
print [Press the Escape key]
ifelse 27 = ascii readchar
  [print[Good!]] [print[Try again]
  key.teach]
end

```

I could also ask a user to press arrow keys, as in an arcade game, and make the computer respond, without having to activate the Turtle-Move mode

```

to move.turtle
ct
print[Press arrow keys to move, c
  for color, Escape to stop]
make "input readchar
if 21 = ascii :input [setx (first
  pos) + 10]
if 8 = ascii :input [setx (first
  pos) - 10]
if 10 = ascii :input [sety (last
  pos) - 10]
if 11 = ascii :input [sety (last
  pos) + 10]
if 27 = ascii :input [show pos
  stopall]
if or :input = "c :input = "C [setc
  color +1]
turtle.move
end

```

Almost any key could be made to do any kind of work that I liked, without having to type a procedure name or press Return after each procedure or command.

However, this exploration of ASCII values still hadn't led to a reason to use **when**. But I had a vague feeling that it would allow me to interrupt a procedure, in the same way that a key stroke or joystick can interrupt a video arcade game. I found I could use

when to interrupt procedures—the key-buffer can be cleared at the end of each **when** event by using

```
make readchar []
```

Such an interruption could make an action happen, such as the turtle moving forward, or make a whole collection of actions occur. In essence this is the idea of a “macro”—a small program that can be called on or activated at any time in the middle of other tasks. Could this be a worthwhile way to use **when** events? And then I had a moment of inspiration. I could use these primitives to make my English-language version of *LogoWriter* type with a French alphabet.

My Turtle “Parles” Français

It is easy to forget that we can combine the power of *LogoWriter* as a word processor with its power as a programming language. *Parlez vous français?* In word processors more powerful than that provided in *LogoWriter*, we can get special characters by switching which “keyboard” or “character set” we use. That can be somewhat cumbersome. *LogoWriter* to the rescue! Try the following procedure written in Apple *LogoWriter*

```

to a.acute
when "n [insert char 1]
end

```

What are the commands inside the square brackets? The primitive **insert** will cause text—in this case a particular letter—to appear where the cursor is active on the page. The command **char number** specifies the keyboard character—that is, a letter or number or symbol—whose ASCII code is “number.” So the expression **char 1** reports to **insert** the character whose ASCII code number is 1.

You can use the following recursive procedure to find the ASCII number for any key you press. Can you make a recursive procedure that will report and print the character for any number given as input for an ASCII code?

```

to report.ascii
make "input readchar
(print :input ascii :input)
report.ascii
end

```

Type

```
report.ascii
```

and press Return, then press any key on the keyboard. You will need to use the Stop key to interrupt the procedure.

How does this procedure work? The command **readchar** makes the computer wait until you press a key and then it "reads" and reports that key to other commands. Whatever key has been pressed is put into the value of a new variable called **"input**". In the next line, the command **print** will print on the Page whatever is contained inside the round brackets containing **print**. This means it prints the value **:input**—the key you pressed—followed by the ASCII number of that key. The command **ascii** accepts the character returned by **readchar** and turns it into its corresponding ASCII number. When this has been done, the next line in the procedure is **report.ascii**.

Using **when** we can create a special event that will be triggered by pressing the Control key simultaneously with one of the keys N, O, P, Q, R, V, W, X, Y, Z. We are ready to build this into a procedure that can let us customize our keyboard so that pressing the Control key plus other keys will give us the nonstandard characters we need to be able to word process in French, Spanish, Norwegian, Swedish, Danish, German, Spanish, and other variations of the standard English-Latin alphabet. Create a new page called, say **"FrenchWriting**". Enter this **startup** procedure to run automatically when you open the Page.

```
to startup
when "n [insert char 1]
when "o [insert char 2]
when "p [insert char 5]
...
and so on
...
end
```

You will need to find the remaining special characters you need, and their ASCII codes, and write **when** commands for them.

You can now type your text. When you need an "e" with an acute accent, for example, press the Control key and the corresponding event key you have programmed for this, and simply continue typing. Of course these "event-keys" (programmed by **when** commands) will also work on the Flip side of the page, inside **print** commands as well as **label** commands. Note, however, that these keys do not work when you are in Label mode. Keep a "cheat sheet" to remind you which specially programmed event keys you need for which special characters. Bon appetit! Saludos amigos! Processez-vous les mots!

John Gough
Deakin University
Toorak Campus
336 Glenferrie Road
Malvern, Vic 3144

THE RUSH IS ON!



Today, the mother lode of knowledge is the "network of networks"—the Internet.

In *Way of the Ferret—Finding Educational Resources on the Internet*, network explorer and guide Judi Harris shows you how to ferret out the information you need—original

source documents, free and inexpensive software, up-to-the-minute news and discussions, cooperative classroom projects, and invaluable personal contacts.

Learn to use information mining tools, including Telnet, FTP, and Gopher. Reach out to colleagues and remote classrooms through e-mail, listservs, and newsgroups.

More than a technical manual on telecommunications, *Way of the Ferret* is a guide to practical classroom and personal use of the Internet, presented with skill and humor by one of the most active authors in educational technology. *Way of the Ferret* is based on Judi Harris's

"Mining the Internet" column in *The Computer Teacher* journal, with information updated to stay current with the constantly evolving Internet environment.

Order your copy of *Way of the Ferret*, and stake your claim to the information gold mine.



International Society for Technology in Education
1787 Agate Street, Eugene, OR 97403-1923 USA
Order Desk: 800/336-5191 Fax: 503/346-5890

The phrase "Mining the Internet" was borrowed with permission from Computing Services personnel at the University of California, Davis.



It's Winter: Let's Build A Snowman

by Robert Macdonald

It's early January in wintry Michigan. A good foot of snow has fallen during the past week. Because I live adjacent to a golf course, cross country skiing is a popular winter sport among the fellow residents in my condominium development. Today, while out skiing near the golfing clubhouse, I spied a former teaching colleague who was building a snowman with the help of some of her students who live nearby. I stopped and helped. We were all rather pleased with the results. After finishing the snowman, the children departed and my former colleague and I entered the clubhouse for some warm refreshments. While drinking hot chocolate and munching on sandwiches, I remarked that I could think of a warmer way to build a snowman. "How?" my friend queried. "Why not create it in a Logo environment?" I hastily replied. "And the students could write winter poetry to enhance their graphic work," she retorted. So we drew the outline of how we could accomplish this task as we finished our refreshments. This article is an outline of what we planned and how we carried it out with a group of her fourth graders the following week.

First, I had to determine the background of the students with whom we would be working. Had they done any work with *LogoWriter*? Some. Many had worked through the basics of turtle graphics, had a limited concept of paired numbers (Cartesian Coordinates), understood the basic principles of creating procedures, could use the word processing possibilities of *LogoWriter*, and knew how to save files and print materials. So we would not be starting from scratch. What background they lacked, we could enhance with instruction.

In analyzing the form of the snowman—snowwomen, snowperson, take your pick—we agreed that using a circle procedure would be essential. We did not have to start with the basic physical movement of moving a little step forward and then turning a bit to the right any number of times to complete a circle:

```
repeat 360 [forward 1 right 1]
```

This the students understood. So we introduced a procedure that would permit us to change the size of the circle with an input. We would need circles of various sizes.

```
to circle :size
  repeat 36 [forward :size right 360 /
    36]
end
```

Try some of these commands:

```
circle 2
circle 4
circle 6
circle 8
circle 10
```

If you try each of these commands in order without clearing the screen, the arrangement of circles is interesting. So interesting, in fact, that it produced another graphic encounter, which will be produced at the conclusion of the article.

The procedure circle works well enough to permit us to build our snowman. But what parts of the snowman do we need? Let's build a procedure to take care of that.

```
to snowman
  preparation
  body
  head
  face
  hat
end
```

This will serve as a structural outline in our top-down approach to the problem. We will now have to construct the subprocedures. Because we were going to use a Macintosh attached to an overhead projector in the classroom as the medium for instruction, we decided to write the program in *LogoWriter* for the Macintosh. The classroom also provided us with an Apple IIe and a GS. On these we had to adjust the circle sizes, positions, and the Cartesian Coordinates to a somewhat smaller screen. While we may click on a turtle on the Mac and drag the turtle to another position, we must resort to turtle-move mode and the arrow keys to position the turtle, then escape to the Command Center to discover the position of the turtle on the IIe and IIGS. This did not prove to be a great problem for the students assigned to those computers.

Winter

White as can be,

Incredibly beautiful.

Nice weather for some people.

Tenderly falling snow.

Exactly what I wished for. I like

Running through the snow.

—Jessica Carver

When we call up the page and give it a name, the turtle on that page is in home position [0 0]. This is where we want it to be. As a safeguard we use a preparation procedure.

```
to preparation
if not front? [flip]
cg
ht
ct
cc
end
```

We need to change the heading of that turtle. Rather than use right and left with inputs, it was decided to introduce the concept of setting headings with the command **seth** with a numerical input. For example:

```
seth 90
seth 180
seth 270
```

You may have to demonstrate these commands physically by having the students act the role of a turtle and follow commands. It is very important when working with turtle graphics that you always know where the turtle is located and the direction in which it is headed.

```
to body
seth 90
circle 9 (circle 7 on I1e, GS)
position
end
```

Now the turtle will come back to the top of the circle, rather than being on the side. It will be in a much better position to draw the circle for the head. But we need to change the position just a bit or the two circles comprising the figure will be a little off center.

```
to position
pu
forward 5 <— You may have to vary
the input, e.g. forward 3 on I1e
pd
end
```

We are ready to construct the head. But we must change the heading of the turtle with another **seth** command.

```
to head
seth 270
circle 5 (circle 4 on I1e, GS)
seth 0
end
```

We decided we needed the **seth 0** to bring the turtle into a position facing straight up. This is essential when we begin the construction of the face.

To construct the face, we make use of a nice feature of *LogoWriter*: the power of clicking on a turtle with the mouse and dragging it directly to a new position by means of the mouse. (Remember you cannot drag an invisible turtle. If you don't see it, use a **st** command.)

It is very easy to discover the position of that turtle by giving the following command in the Command Center:

```
show pos
```

The coordinates will be given within brackets. For example:

```
[19 32]
```

You may then use these coordinates with a **setpos** command:

```
setpos [ 19 32]
```

Note: Do not use a comma to separate your two coordinates as in paired numbers. In Logo, a space is sufficient.

Dragging a turtle is a very easy way of moving it to a new position. Be certain that you put the pen up before you give the **setpos** command and put it down after you have arrived in the new position for the turtle. Most fourth graders have been given instruction in the use of paired numbers in the first quadrant. You might like to expand the instruction to the second, third, and fourth quadrants. (Many publishers offer a rich array of materials suitable for younger students desiring to develop skills with paired numbers—Cartesian Coordinates. We also devised some lessons for developing skills in using paired numbers. But this will require another article.)

```

to face
left. eye
right. eye
nose
mouse
end

```

The subprocedures indicated in the above procedure are provided. Note specifically the **setpos** commands.

```

to left. eye
seth 0
pu
setpos [-2 32] (setpos [-4 30] on
  I1e, GS)
pd
circle 1
end

```

We decided to place the eyes a little to the right, looking at the screen, as we wished to give the impression of the snowman gazing in that direction, rather than looking straight ahead. We also used the circle procedure for drawing the eyes. The eyes are reminiscent of those of Little Orphan Annie. Some students on the I1e and GS decided that a small circular shape from the Shapes page was a better alternative. We agreed.

```

to right. eye
seth 0
pu
setpos [19 32] (setpos [11 30] on
  I1e, GS)
pd
circle 1
end

```

For the nose we wished to simulate a carrot. We did it as simply as possible.

```

to nose
pu
setpos [15 25] (setpos [10 23] on
  I1e, GS)
pd
seth 110
forward 20      (forward 16)
seth 270
forward 20      (forward 16)
end

```

For the mouth we thought we would use a simple straight line:

```

to mouth
pu
setpos [6 9] (setpos [0 8])
seth 90
pd
forward 15
end

```

Our snowman needs a hat. Because the students had some experience building procedures for squares with the **repeat** command, we decided to combine that procedure with a straight line to construct a hat and fill it in.

```

to hat
pu
seth 90
setpos [-16 56] (setpos [-17 45])
pd
forward 50
back 36
seth 0
square 20
pu
setpos [9 65] (setpos [6 57])
pd
fill
end

```

The procedure for the square needs an input (see above).

```

to square :size
repeat 4 [forward :size right 90]
end

```

All well-dressed snowmen need buttons. A good button shape (shape 12) is already available to us on the Shapes page—a filled-in circle. We shall stamp it in place, then move on to another button and do the same.

Snow

Snow is sparkling white.

No, please don't eat it.

Oh! It's so wonderfully bright.

What a sight!

—Shawn Hughes

Ice

International holiday;

Christmas has passed;

Everyone's happy.

-Drew WippichSled.

```
to buttons
pu
setpos [7 -30] (setpos [6 -23])
pd
setsh 12
pd
stamp
pu
setpos [7 -60] (setpos [6 -48])
pd
stamp
end
```

Remember that the pen must be down or you cannot stamp. (The buttons appeared too large on the Ile and GS, so we cut down the size of the circle on the shape 12 on the shapes page. Remember some students had used another smaller circular shape for the eyes previously.)

The Complete Computer Program

To clarify any ambiguity, the complete program is listed below. The command, of course, is **snowman**.



```
to snowman
preparation
body
position
head
face
hat
buttons
poem
end
```

```
to preparation
if not front? [flip]
cg
ht
ct
cc
end
```

```
to body
seth 90
circle 9
position
end
```

```
to circle :size
repeat 36 [forward :size right 360 /
36]
end
```

```
to position
pu
forward 5
pd
end
```

```
to head
seth 270
circle 5
seth 0
end
```

```
to face
left.eye
right.eye
nose
mouth
end
```

```
to left.eye
seth 0
pu
setpos [-2 32]
pd
circle 1
end
```



```

to right. eye
  seth 0
  pu
  setpos [19 32]
  pd
  circle 1
end

to nose
  pu
  setpos [15 25]
  pd
  seth 110
  forward 20
  seth 270
  forward 20
end

to mouth
  pu
  setpos [6 9]
  seth 90
  pd
  forward 15
end

to hat
  pu
  seth 90
  setpos [-16 56]
  pd
  forward 50
  back 36
  seth 0
  square 20
  pu
  setpos [9 65]
  pd
  fill
end

to square :size
  repeat 4[forward :size right 90]
end

to buttons
  pu
  setpos [7 -30]
  pd
  setsh 12
  pd
  stamp
  pu
  setpos [7 -60]
  pd
  stamp
end

```

```

to poem
  tab print [NOEL]
  print []
  print [Now the snow is falling
    down,]
  print [On the frozen ground.]
  print [Everywhere we chance to
    glance,]
  print [Lightly snow falls down.]
end

```

We had to redo the formatting on the poem for the Ile and Gs. We give it below:

```

to poem
  tab print [NOEL]
  print []
  print [Now the snow is]
  print [ \ falling down,]
  print [On the frozen]
  print [ \ ground.]
  print [Everywhere we]
  print [ \ chance to glance,]
  print [Lightly snow]
  print [ \ falls down.]
end

```

The final procedure is entitled Poem. The students were given the task of writing some form of poetry to conclude the assignment. One student decided to write an acrostic poem. Williams (1986, p. 139-40) has defined the structure as follows:

An acrostic is any poem in which the first letters of the lines, read down the page, spell out a message or a name.

The student entitled his poem NOEL. He began each line of his poem with a letter from the title. Students may use the word processing features of *LogoWriter* to type in their poetry if they so choose.

Snow

Snow is so cold,

Not warm at all.

Only in wintertime.

Wow! But it's fun.

-Jill Funnell



Possible Additions

We have used a number of procedures to construct our snowman. Other possibilities are as follows:

- Construct some arms resembling broken branches
- Have the snowman hold a broom
- Put a scarf around his neck
- Perhaps one could animate a little rabbit running by, stopping, looking at the carrot nose, and then hopping away when he realizes he cannot reach it.
- Place a horizon line in your picture
- With the square procedure you already have, construct a classic Logo house.



For example:

```
to logo.house :size
square :size
move.to.roof :size
triangle :size
ht
end

to square :size
repeat 4 [forward :size right 90]
end

to triangle :size
repeat 3 [forward :size right 120]
end

to move.to.roof :size
forward :size
right 30
end
```

Ice

It's slippery and

Cold, but

Everyone still has fun!

—Meredith McLaughlin

Frost

Fun in the snow as we get

Red, red noses and we go

Over hills and hilltops in our

Oh! What fun as

The temperature drops.

—Allison Taurence

A good input might be 40:

logo.house 40

If you desire a smaller house, decrease the input. You may also place the house anywhere in the scene by dragging the turtle. Then give the command. If you make a house large enough, you might like to put in windows and a door. With all you have done up to this point, that task should be somewhat simple. (We also found that providing a lesson on the Rule of 360—The Total Trip Theorem—was also profitable. However, that may prove to be the base of another article.)

A Little Summation

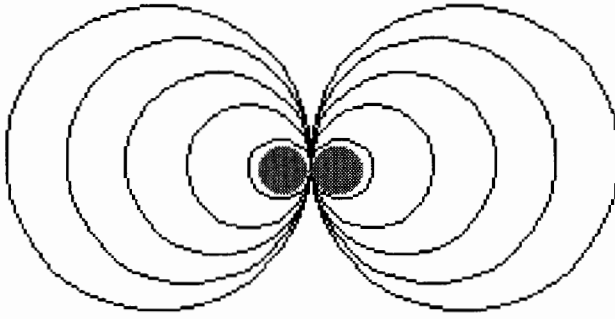
The preceding program is an example of top-down design. It is easy to follow. Going through the program step-by-step with a class may pay dividends.

As students are working through an assigned task, they may be introduced to elements that may prove helpful. It was with this in mind that paired numbers were introduced by means of a **show pos** command. But it was *LogoWriter's* ability to drag a turtle from one position to another that made it profitable. The **seth** commands are also a useful way of looking at the orientation of a turtle. Being able to integrate skills one has previously gained, keeps them in mind. Hence the use of **setsh**, **stamp**, and **fill** commands. All have their place in working through this little assignment in turtle graphics. Finishing off the assignment with a writing assignment merely underlines the versatility of the software.

A Concluding Program

The idea for this program was generated by the interest shown in the design produced by consecutive circle commands presented at the beginning of this article. One student thought it looked like a comic eye or "someone who did too much reading." (We didn't encourage that.)





The group immediately wanted to construct an eye on the other side for a sense of facial symmetry. After due consideration, the group chanced on the need for two circle procedures—one for the right side, another for the left. Everything was built up from that.

Typing the following program and duplicating for a class discussion will fix in the mind of students many of the elements employed in the previous program. Students should be encouraged to analyze their own programs and those of others. Learning is an open-ended activity. It never ceases.

This program is also written in *LogoWriter* for the Macintosh. The program begins with a startup procedure. It may be repeated the same way. You will be coached in the Command Center at the conclusion of the program.

```
to startup
  preparation
  circular.eyes
end
to preparation
  rg
  if not front? [flip]
  ct
  cc
end
to circular.eyes
  first.message
  ht
  right.eye
  right.stamp
  pu
  home
  pd
  left.eye
  left.stamp
  set.up
  repeat 20 [move move.again]
  final.message
  wait 100
  clear
  continue
end
```

```
to first.message
  print [Have you been using your eyes
    too much?]
  print []
end
to right.eye
  right.circle 10
  right.circle 8
  right.circle 6
  right.circle 4
  right.circle 2
end
to right.circle :size
  repeat 36 [forward :size right 360 /
    36]
end
to right.stamp
  setsh 12
  setc 221
  seth 90
  forward 10
  pd
  stamp
end
to left.eye
  setc 1
  left.circle 10
  left.circle 8
  left.circle 6
  left.circle 4
  left.circle 2
end
to left.circle :size
  repeat 36 [forward :size left 360 /
    36]
end
to left.stamp
  setsh 12
  seth 270
  forward 10
  setc 221
  pd
  stamp
end
to set.up
  setc 0
  stamp
  wait 4
  setc 221
  stamp
  wait 3
end
```

```
to final.message
print [THEN REST THEM!]
end
```

```
to clear
rg
if not front? [flip]
ct
end
```

```
to move
seth 90
forward 21
set.up
end
```

```
to move.again
back 21
set.up
end
```

```
to continue
cc
type [If you wish to run the program
      again, type in STARTUP and touch
      the
RETURN.]
type char 13
end
```

Sled

Slick and

Loud we go

Evening or

Dawn.

-Pam Rutkowski

Reference

Williams, Miller (1986). *Patterns of Poetry. An Encyclopedia of Forms*. Baton Rouge and London: Louisiana State University Press.

Robert Macdonald
Hawthorne Meadows
10225 Nancy's Blvd.
Grosse Ile, MI 48138

PC Logo™ FOR WINDOWS™

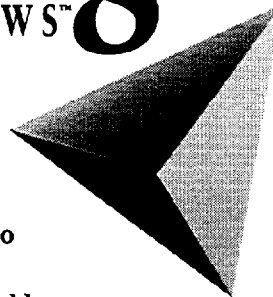
It's Here!

A powerful new version of the Logo programming language is now available for Windows.

PC Logo for Windows combines a full-featured Logo with the ease of use of the Windows environment.

Economical multiple-workstation licensing is available.

Exciting robotic connections as well as a full line of Logo materials, books, and curriculum enhance PC Logo for Windows.



Features include:

- Illustrated, full-color on-line help
- Multiple turtles: command separately or together
- Turtle color, speed and shape control
- Supports Windows installed fonts and printers
- Compatible with other Windows programs

For more information or to order, call:

800-774-LOGO

HARVARD
ASSOCIATES, INC.

10 Holworthy St. • Cambridge, MA 02138 • Fax (617) 492-4610
CompuServe 70312,243 • Internet 70312.243@compuserve.com



What's New With Logo PLUS 2.0 (for the Mac)!

by Dorothy Fitch

At last, Logo PLUS for the Macintosh 2.0 has arrived! Here is a sampling of the new features, many of which are in response to user suggestions.

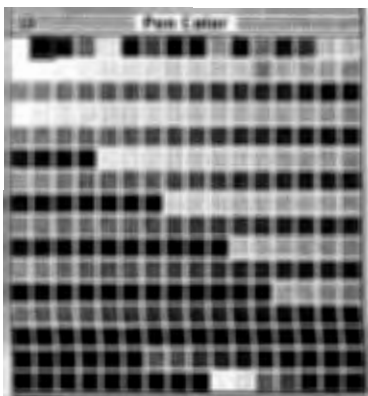
Color!

The most prominent new feature is color! Let's take a look at the color options.

Logo PLUS comes set up for 256 colors, although you can switch to show just 16 colors using a command or menu option. (Young children may find it easier to work with a more limited choice, and the color selection boxes are larger with fewer colors.)

You can select a pen color in a variety of ways:

- use the SETPC command, abbreviated PC, with a number from 0 to 255
- use SETPC with one of 16 color words, as in SETPC RED or SETPC BLUE
- choose Set Pencolor from the **Display** menu (or press K) and click on the color you want
- choose Set Display from the **Display** menu and preview color combinations

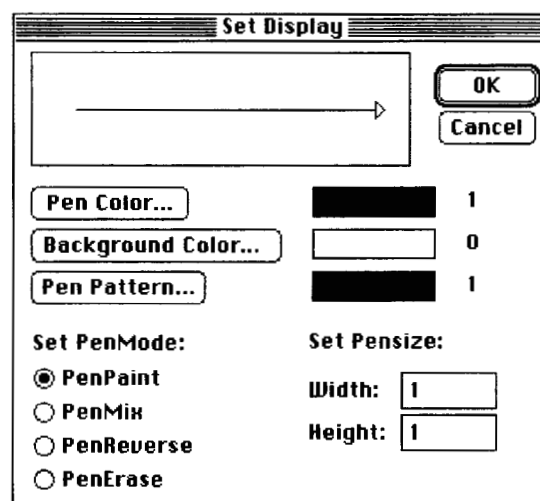


You can set a background color just as easily. Choose a menu option or press B. GETPC and GETBG report the current pen and background colors, and COLORUNDER reports the color under the turtle.

Selecting a pen pattern is easy; just select Set Penpattern from the menu or press T and click on the pattern you want.



The handy Set Display box lets you preview your graphics environment. Choose pen and background colors and pen patterns, size, and mode. View your settings in the sample box. Change them until you get the combination you want.



You can also create custom colors using either a menu option or the MAKECOLOR primitive. COLORINFO reports RGB (red, green, or blue) information about any color, and you can reset an individual color or the entire palette to its original value. The SWAPCOLOR tool procedure lets you change the position of the colors in the palette.

Music!

Logo PLUS now has built-in music commands. The NOTE command takes two inputs: a pitch value (where middle C is 60, C# is 61, D is 62, etc.) and a duration value (where 60 is one second).

The NOTE command loads a set of note utilities to make composition easier. In addition, six songs are included in a sample file.

These Logo PLUS instructions play the first phrase of "Twinkle Twinkle Little Star:"

NOTE C 30
 NOTE C 30
 NOTE G 30
 NOTE G 30
 NOTE A 30
 NOTE A 30
 NOTE G 60

Built-in primitives let you change the volume of sound from Logo.

Shapes!

Logo PLUS shapes now rotate in 4 directions. The Shape Edit program has been enhanced to allow shapes to be flipped and rotated. You can also lock the heading of a shape—it doesn't have to point the same way the turtle is headed.

The 125 ready-made shapes can be addressed either by name or number. Type RESETTURTLE to restore the original turtle shape. Windows showing the shapes can be loaded for easy reference. Here is a sample shape from each of the shapes files:



Easy Customization!

Although we have always encouraged users to modify the Logo.init file to customize their Logo environment, few are courageous enough to do so. In Logo PLUS, we have made many of the common customization options a snap to implement. For example:

- to have Logo PLUS position your windows on startup automatically, just arrange your windows to your liking, then type KEEPWINDOWS.
- to have Logo PLUS load a set of procedures or instructions automatically, simply save them in a file named STARTUP in your Logo PLUS folder.
- to have Logo PLUS for the Mac look and feel more like Logo PLUS for the Apple II, load the Apple II Emulator file.

More New Commands!

Many of you have asked for specific procedures. As these may not be useful to everyone, we have implemented them as autoloading procedures. Use them as you would primitives—each procedure automatically loads when you first use it. A benefit of autoloading

procedures is that they do not take up memory until you use them.

The tool files provided are:

- **GraphicsTools** for filling circles, arcs, and rectangles; using background patterns; swapping colors; and switching graphics windows
- **MathTools** for converting numbers from base to base, a number formatting tool, polar coordinate commands, and additional trig functions
- **MusicTools** for using note names in songs
- **ProgTools** for handy programming tools
- **ShapeTools** for locking shape headings, looking at shape names, and saving shapes
- **SlowTurtleTools** for controlling the speed at which the turtle moves and turns
- **TextTools** for typing text directly in the Turtle window in any font, size, or style
- **TurtleTools** for arranging multiple turtles
- **WorldTools** for saving and restoring a complete Logo environment

All New Documentation!

Based on user suggestions, our new documentation contains much more help in getting started with Logo and guided lessons with lots of activities. Included in the complete package are:

- Logo PLUS Guide, with nine step-by-step lessons, two complete projects, and five specialty "Exploring" chapters: Graphics, Shapes & Animation, Multiple Turtles, Music and SuperText
- Getting Started! booklet with quick activities to try in all areas of Logo
- Quick Reference Card, with commonly used commands
- Quick Reference Guide, with a brief description of every command
- Logo PLUS Reference Manual

But Wait... There's More!

Logo PLUS now offers text cursor commands. Use SETCURSOR to position the cursor in a text window. You can determine how many characters are in a window and even highlight text from within a program. Related graphics cursor commands position the turtle for text in the graphics window.

Printing a picture is now as easy as pressing P.

New programs in the Games folder include Solar Explorer (travel from planet to planet), The Escape of Robert Smalls (help a slave escape during the Civil War in this fact-based adventure), Solitaire (play a card game using the mouse), Balloons (estimate fractions and decimals), and Jotto (the classic word game).

You can now edit the online help information for built-in commands and create help messages for your own procedures. Closing an Edit window gives you a choice of saving, closing, or defining its contents. New serial port commands allow you to change the baud rate from Logo as well as modify other communications settings.

Don't forget that Logo PLUS contains all the Terrapin Logo for the Mac standard features: unlimited multiple turtles; resizable windows; multiple graphics, text, and edit windows; procedure burying; program tracing and variable watching; mouse-handling primitives; arrays; and stream I/O.

Upgrades and Packaging Options

Logo PLUS is 32-bit clean, and requires 4 MB of memory when running System 7. Upgrades from Terrapin Logo/Mac single packages and site licenses are available. In addition, you can switch to the Macintosh from other Terrapin Logos. Multistation licenses (for 5, 10, and 20 computers) are offered to schools with small numbers of computers. For networks or schools with many computers, a one-time site license fee of \$450 covers up to 25 computers, with a nominal fee for additional machines.

Logo Works: Lessons in Logo is also now available for the Macintosh as a site license for just \$200.

Crystal Rain Forest, an adventure set in a tropical rain forest that teaches Logo fundamentals, is a great way to introduce any version of Logo.

Schools may preview Terrapin software by sending a request on school letterhead. For more information on any of its products, contact:

Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103

Telephone: 800/972-8200
Fax: 207/878-0956
e-mail: 71760.366@compuserve.com

Meet *LogoWriter's* Four Turtles

by John Gough

Using a turtle to draw lines and polygons is a well-known Logo activity. Some versions of Logo allow several turtles at one time. In the dialect *LogoWriter* there can be four turtles at one time carrying out instructions. The turtle that appears when you start *LogoWriter* is turtle number zero. Until you give special instructions it is the only turtle you can see or work with. But the other three turtles (1, 2, and 3) are present, with pens down, waiting to be told what to do. Try these groups of commands to see them spring to life.

```
tell 1
right 45
forward 80
```

```
tell 2
seth 90
setc 4
pu
forward 20
pd
forward 20
st
```

```
tell 3
st
setsh 22
setc 2
left 1
forward 2000
```

The command

`tell number or list of numbers`

makes turtles (0, 1, 2 or 3, whichever numbers are specified) become active so that they will carry out all the following commands until another turtle is instructed to become active.

To make more than one turtle active at a time, use square brackets after the `tell` command to list the numbers of the turtles that are to be active. For example, try this

```
tell [0 3]
right 30
forward 80.
```

The command `tell [0 3]` makes only turtle number 0 and turtle number 3 be the active turtles, in the order specified.

You can also tell all of the turtles to be active. For example

```
rg
tell all
st
```

Then `tell all` makes all four turtles become active. Any further commands will be carried out by all of the turtles simultaneously. Incidentally, `tell all` is equivalent to the command `tell [0 1 2 3]`?

Some Multiple-Turtle Procedures

The procedure `whip` moves the turtle relatively slowly forward through a total of 30 steps then brings the turtle back in one jump. In `whizz` the first four `tell` commands give specific instructions about shape and heading for each of the four turtles, then the `tell all` line makes them all appear and they all carry out the procedure `whip`.

```
to whip
pu
repeat 15 [forward 2]
back 30
end
```

```
to whizz.1
turtle.0
turtle.1
turtle.2
turtle.3
tell all
st
whip
end
```

```
to turtle.0
tell 0
setsh 26
seth 0
end
```

```
to turtle.1
tell 1
setsh 15
seth 90
end
```

```
to turtle.2
tell 2
setsh 16
seth 180
end
```

```
to turtle.3
tell 3
setsh 17
setc 4
seth 270
end
```

Next try replacing the middle line of **whip** with these lines

```
pu
repeat 30 [forward 1 wait 1]
back 30
```

Then try experimenting with different numbers after the **forward** or the **seth** or after the **wait**. Can you make the turtles do their whizzing while slowly altering their headings so that they appear to be rotating in between each **whip**? Can you make them change their starting position so they move to the right between each **whip**? Can you make the turtles rotate and roll? Can you make them **whizz** from the lower left of the screen to the upper right?

Four turtles can do the same thing simultaneously. However it is harder to make more than one turtle appear to do different things at the same time. By moving one turtle a bit, then the next a bit differently, and so on, it is possible to create animation that almost looks as though several turtles are all active at the same time, doing different things. Examine these procedures.

```
to piston
aim.0
aim.3
repeat 50 [tell 0 forward 1 tell 3
forward 2]
repeat 25 [tell 0 back 2 tell 3 back
4]
end
```

```
to aim.3
tell 3
seth 270
st
pu
end

to aim.0
tell 0
seth 90
st
pu
end
```

Can you extend this so that turtle 0 moves in a curve while turtle 3 moves in a straight line, or so that they both move in curves of different amounts of curvature? Can you use more than one turtle to draw petals on a flower? Wings of a butterfly? Objects with rotational and reflection symmetry? Can you use turtle shapes and the command **stamp** to create stamped symmetrical patterns?

Using the Command "Each"

A different way of giving instructions to more than one turtle at a time uses the command **each**. The specific commands that each turtle will carry out are listed in square brackets following **each**. The order in which each specified turtle carries out these listed instructions is the order of the turtles given in the **tell** command. For example

```
tell all
each [print pos]
```

tells all the turtles, each in turn, to print their current turtle coordinate position. Consider what happens with the following commands.

```
tell [0 2 1 0]
print pos
```

The commands

```
tell [3 2 1 0]
each [print pos]
```

cause them to print their position in the specified reverse order.

Compare the procedure **whizz.2** using **all** and **each** with **whizz.1**.

```

to whizz.2
  setup.0
  setup.1
  setup.2
  setup.3
  tell all
  st
  repeat 15 [pu forward 4 pd forward
    0]
  tell [3 1 2 0]
  each [px back 60]
end

```

```

to setup.0
  tell 0
  setsh 12
  seth 90
end

```

```

to setup.1
  tell 1
  setsh 13
  seth 180
end

```

```

to setup.2
  tell 2
  setsh 14
  seth 270
end

```

```

to setup.3
  tell 3
  setsh 11
  seth 0
end

```

In the procedures **whizz.1**, **whizz.2**, and **piston**, experiment with different values for the amount movement and the order in which successive turtles in the **tell** command are told to do things. Try different headings for the turtles; insert some amount of **turn** inside the **each** instructions; set different starting positions for the turtles; use **pd**, or set a background color and use **pe** so the pen will erase as the turtle moves; use different colors for the turtles to draw; use large numbers for moving forward and slightly skew headings; use **px**, which will make the turtle draw where there is nothing already drawn and erase where there is something already drawn, or change the order of the turtle numbers in the **tell** command. Try incrementing the turtles' x or y coordinates, or their headings.

Using Turtles to Label

In *LogoWriter*, turtles can also be used to place specified writing in "graphics mode" at the turtle's current position rather than the "text" mode when using the **print** command. Such a label will be erased when the turtle repeats exactly the same labelling command in exactly the same position.

```
repeat 2 [px label [whatever message]]
```

Try this procedure. You too can have your name in neon lights!

```

to scroll.label
  set.up.turtles
  repeat 10 [
    tell 0 label [Eat]
    tell 1 label [At]
    tell 2 label [Flash]
    tell 3 label [Sam's!]
    wait 1]
end

```

```

to set.up.turtles
  tell all
  ht
  pu
  tell 0
  setpos [-130 0]
  tell 1
  setpos [-70 0]
  tell 2
  setpos [0 0]
  tell 3
  setpos [70 0]
end

```

You may want to adjust the initial horizontal coordinates given in the **setpos** commands of **set.up.turtles** so the words appear suitably spaced apart. Can you extend this so that the flashing words gradually drift up the screen? Or across the screen? Or diagonally from bottom left to top right?

Here is a different way of using the command **label**. It sets random positions by using the command **random** to choose values for the two coordinates—horizontal first and vertical second. The **setpos** numbers come in multiples of 12 or 10, and so the positions will not overlap.

```

to boo
tell all
pu
ht
repeat 10
  [tell all each [setpos list
    (12*random 10) (10*random 8)]
  tell all pd label [Boo!] wait 1
  pe label [Boo!]]
end

```

One turtle can follow another, erasing whatever the first turtle has drawn.

```

to lightning
tell [0 1]
ht
pu
repeat 10 [lightning.bolt]
end

to lightning.bolt
tell 0
setpos list random 135 random 85
tell 1
setpos ask 0 [pos]
tell 0
pd
bolt
tell 1
pe
bolt
end

to bolt
seth 220
forward 50
right 30
back 15
left 30
forward 30
end

```

Here the reporter **ask** interrupts whichever turtle is currently active, so that it can have the list of instructions following in the square brackets run by the specified turtle number. In this case, the command **pos** asks turtle 0 to report the coordinates of its current position, and turtle 1 uses this report as the coordinates for its **setpos** command. Can you change this so the second turtle does not erase but changes the first turtle's color?

Turtles in Motion

We can set several turtles dancing or cartooning on the screen. The commands **seth random 360** give each turtle a randomly chosen heading, or initial direction, to go walking.

```

to step
setsh 17
wait 0
setsh 18
wait 0
end

to walk.four
tell all
each [seth random 360 pu]
tell all
st
repeat 20 [step forward 5]
end

```

Experiment with different values of **wait** to control the speed of the animation, or use **tone 200 10**, instead of a **wait** command, to get a sound effect (on IBM *LogoWriter* use **tone 200 10 wait 10**).

And Now Back to One

As a final exercise, go back through each multiturtle procedure that uses separate **tell** commands or an **each** command to have one turtle moving at a time, and try to make new procedures that will do the same work using just one turtle. Some adapted one-turtle versions of multiturtle procedures can be run in ordinary Logo, which has only one turtle, when Logo uses commands such as **label**. For example,

```

to one.scroll.label
ht
pu
repeat 10
  [tone 200 50
  setpos [130 0]
  label [Eat]
  setpos [-70 0]
  label [At]
  setpos [0 0]
  label [Flash]
  setpos [70 0]
  [Sam's!]]
  wait 1]
end

```

Try both

```
scroll.label
```

and

```
one.scroll.label
```

Which works quickest?

Here is another procedure adapted for one turtle only, which will work in a one-turtle Logo. It uses a **make** command to create a variable called **random.pos**, which can store the randomly chosen numbers for the starting position of each bolt: once with the pen down to draw the bolt and once with the pen set to erase to remove it.

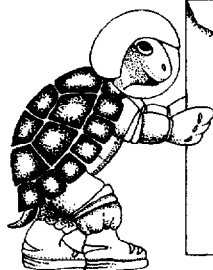
```
to lightning.1
ht
pu
repeat 10 [lightning.bolt.1]
end

to lightning.bolt.1
make "random.pos list random 135
random 85
setpos :random.pos
pd
setc random 5
bolt
pu
setpos :random.pos
pe
bolt
end

to bolt
seth 220
forward 50
right 30
back 15
left 30
forward 30
end
```

Can you find other ways of using more than one turtle at a time? Have multiple fun—the more the merrier! Turtles love company, and some of them are party dudes!

John Gough
Deakin University
Toorak Campus
336 Glenferrie Road
Malvern, Victoria 3144
Australia

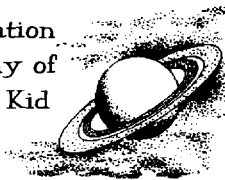


Push the envelope with MicroWorlds



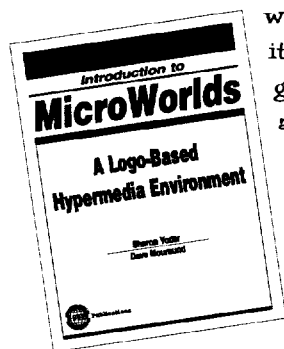
Launch your students toward stimulating new classroom challenges and adventures. They can soar to a galaxy of new skills using MicroWorlds.

MicroWorlds is an exciting application that offers in a single program many of the features available in HyperCard, Kid Pix, and LogoWriter.



You're the pilot. Get ready now to grab the controls, blast off, and explore these exciting new worlds with your class! To help you prepare your crew for this fantastic voyage, Sharon Yoder and Dave Moursund back at mission control have developed a 222-page flight manual titled *Introduction to MicroWorlds—A Logo-Based Hypermedia Environment*.

As a computer application, MicroWorlds contains a wide range of features that make it easy and fun to work with color graphics, sound, text, and animation. As a programming environment, MicroWorlds includes a powerful and modern version of the Logo programming language.



It's time to broaden your universe. The countdown is underway.

Have a nice trip!



International Society for Technology in Education
1787 Agate Street, Eugene, OR 97403-1923 USA
Order Desk: 800/336-5191 Fax: 503/346-5890

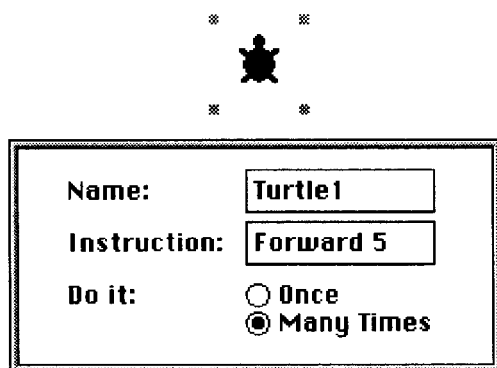
HYPERCARD® IS A REGISTERED TRADEMARK OF CLARIS CORPORATION, SANTA CLARA, CA, U.S.A. KID PIX® IS A REGISTERED TRADEMARK OF BRØDERBUND SOFTWARE, NOVATO, CA, U.S.A. LOGO®, LOGOWRITER®, and MICROWORLDS® ARE REGISTERED TRADEMARKS OF LOGO COMPUTER SYSTEMS, INC., HIGHGATE SPRINGS, VT, U.S.A.



The term “multimedia” refers to combinations of various media—text, digitized images, sound, digital movies, and animation—within a single digital document on the computer screen. There are several ways of achieving animated effects. Traditional animation in film is achieved through a series of frames in which an object is moved slightly from frame to frame. When the film is projected at a rate of 30 frames per second, the illusion of motion is created. Flip books in which an object in the corner of the page appears to move as pages are flipped rely upon this same principle. A number of programs such as *HyperCard* make use of this principle by flipping electronically from page to page or card to card on the computer screen.

Some of the first versions of Logo relied on *sprites* for animation. The first sprites were found in Texas Instruments (TI) Logo, and made use of graphics hardware found in the early TI microcomputer. A traditional Logo turtle moves forward the requested number of steps and stops. In contrast, a sprite remained in motion until asked to stop. This feature was so popular that a special version of Logo called *Sprite Logo* requiring a special hardware was also created for the Apple II.

MicroWorlds provides a clever way of emulating the effects of sprites in software without special hardware. If the shift key is pressed and at the same time the mouse is used to double-click on the turtle, a dialog box appears. This dialog box allows instructions to be assigned to the turtle.



An option of “Many Times” instructs the turtle to carry out the instruction repeatedly. The turtle will continue to carry out the commands in the background even

Animation

Glen L. Bull, Gina L. Bull, and Todd Kent

while other events are occurring. In effect, the turtle behaves like a sprite. (In UNIX there are processes called *daemons* that continually run in the background.) The turtle in *MicroWorlds* can assume the shape of other objects such as a sailboat or a train. Changing the shape of the turtle and instructing it to run an instruction “Many Times” provides a way of emulating “sprite-like” behavior.

Changing the Turtle’s Shape

To change the shape of the turtle, select the picture of a moon in the *MicroWorlds* toolbox to open the Shapes center.



The Shapes center displays a number of pre-made shapes. It is also possible to edit existing shapes or add new ones. Click the picture of the sailboat to select it.



After a shape in the Shapes center has been chosen, click on the turtle.



The turtle takes on the shape of a sailboat:



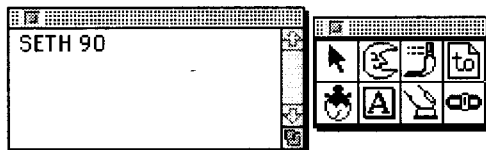
Animating the Sailboat

Before sending the sailboat across the screen, first choose the direction in which it will travel. The direction can be set using the *seth* (set heading) command. The turtle’s heading is specified in terms of the points of a compass: 0 degrees = North, 90 degrees = East, 180

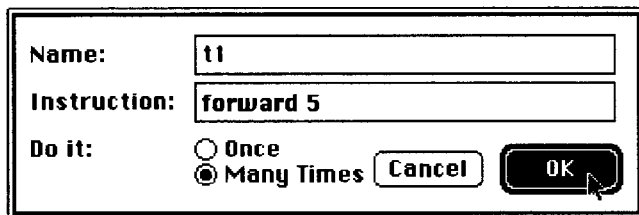
degrees = South, and 270 degrees = West. Click on the arrow pointer in the toolbox to open the *MicroWorlds* Command center. Then enter

```
seth 90
```

and press the Return key:



After setting the direction in which the sailboat will travel, hold down the Shift key and double-click on the sailboat to display the turtle's dialog box. Enter the instruction **forward 50**, select "Many Times," and click on "OK" to confirm the choice. You can also change the turtle's name from "t1" to "Sailboat" if you wish.



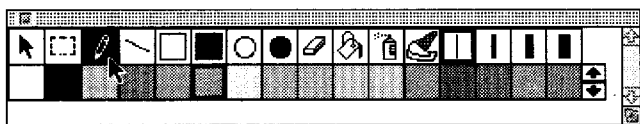
When you close the dialog box, clicking on the sailboat will cause it to move across the screen. While the sailboat is moving, its direction can be changed by entering the **seth** command with a direction between 0 and 360 degrees in the Command center.

The sailboat can be stopped by clicking on it once. If the sailboat is moving too fast to catch, entering the command **stopall** in the Command center will stop all turtles. You may want to experiment with the effect of using different numbers of turtle steps. How does changing the command in the Turtle's dialog box from **forward 5** to **forward 1** or **forward 10** affect the sailboat?

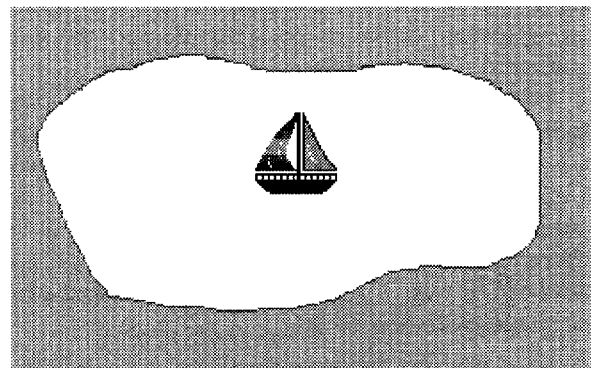
If the paintbrush in the *MicroWorlds* Tool Palette is selected,



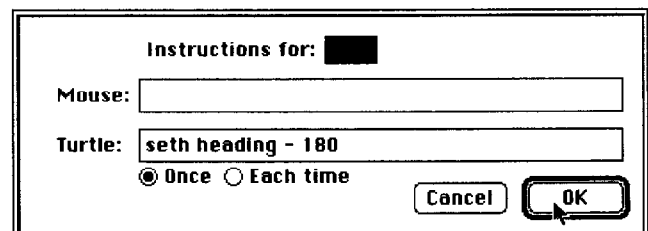
the Command center is replaced by the Drawing center.



The pencil tool can be used to draw a border around a lake. If the border does not have any breaks in it, the paint bucket can be used to fill the shore around the lake with a color.



Initially the sailboat sails out of the lake across the shore. Double-clicking on the color in the Drawing center that is the same shade as the shore produces a dialog box.



The color that is the same shade as the shore can be given instructions in the same manner that you give instructions to the turtle. This command will reverse the direction in which the sailboat is traveling when it reaches the shore.

Place the sailboat in the middle of the lake, and click once to start it sailing. When the sailboat reaches the shore, it will reverse directions and sail until it reaches the opposite shore. This continues indefinitely.

A more sophisticated strategy might be to vary the heading a bit each time the sailboat turns. For example, try replacing the color's instruction with the following:

```
seth heading - 170 + random 20
```

The sailboat will now tack back and forth across the lake.

Animation Using Multiple Shapes

Another type of animation involves shifting among two or more shapes. For example, when the watch hands on the Macintosh "wait" icon move around the face of the watch, the computer is actually shifting among several different icons that display the watch hands in different positions.

MicroWorlds supplies several different shapes in the Shapes center that assist in this type of animation. A pair of bird shapes named Bird1 and Bird2 have the wings of a bird in different positions, and a pair of dogs named Dog1 and Dog2 display the animal in two different phases of running. The **setsh** (set shape) command can be used to change the shape of a turtle from one shape to another.



The turtle-hatching icon can be used to create a new turtle if there is not already one on the screen. Click on the turtle-hatching icon



and then click anywhere else on the screen to create a new turtle.

Next, open the Shapes center. Move the pointer over the bird with its wings raised and hold down the mouse button until a balloon pops up.



This balloon gives the name of the bird's shape: bird1. The bird with its wings pointed down is called bird2. If the shape of the turtle changes repeatedly from one bird to the other while the turtle moves forward, the bird-turtle appears to fly.

To enter the appropriate commands, select the arrow pointer, hold down the Shift key and double-click on the turtle to display its dialog box. Enter these commands:

```
setsh "bird1  
setsh "bird2
```

Select "Many Times." Then click on the turtle to cause it to shift repeatedly between the two shapes. Click on the bird again to stop it.

Using only the above commands makes the bird look like a hummingbird that fans its wings as it hovers in one place. Adding a **forward** command causes the hummingbird to fly across the screen as it flaps its wings.

```
setsh "bird1  
forward 4  
setsh "bird2  
forward 4
```

When you are in the Shapes center, it is possible to edit a shape by holding down the Shift key and double-clicking on a shape. If you use the Shape Editor to change the color of the bird from red to gray, you might think of it as a "turtle dove." (Sorry about the pun!)

Procedures with Names

If animating the turtle involves only one or two commands, it is easiest to type the commands directly in the turtle's dialog box. As the commands become more complex, they may be easier to work with if they are put on the Procedures Page. To go to the Procedures Page, click on the Procedures Page tool.



The Procedures Page is used to create procedures, thus giving a name to a series of instructions

```
To fly  
setsh "bird1  
forward 4  
setsh "bird2  
forward 4  
end
```

Click on the Procedures Page tool to return to the page with the turtle. Double-click on the turtle to see its dialog box and enter the name of the procedure that has just been defined on the Procedures Page.

Name:	<input type="text" value="t1"/>
Instruction:	<input type="text" value="Fly"/>
Do it:	<input type="radio"/> Once <input checked="" type="radio"/> Many Times <input type="button" value="Cancel"/> <input type="button" value="OK"/>

Test the change by clicking on the bird once. It should fly across the screen as before. However, now it will be easier to introduce more complex and subtle changes into the procedure. For example, can you think of a way to cause the turtle to move diagonally across the screen, or perhaps fly in circles. (Hint: Try adding the command `right 5` at the end of each line of the procedure.)

Controlling Several Objects (Nearly) Simultaneously

The Shapes center contains the shapes of an engine and car. Use the turtle-hatching icon to hatch three turtles.



Then return to the Shapes center, and change the first turtle into a train engine. Change the next two turtles into train cars.



Double-click on the train engine to produce its dialog box. Enter "Engine" as the name of the turtle. Click the "OK" button to confirm the choice and close the dialog box.

Name:	<input type="text" value="Engine"/>
Instruction:	<input type="text"/>
Do it:	<input type="radio"/> Once <input checked="" type="radio"/> Many Times
<input type="button" value="Cancel"/> <input type="button" value="OK"/>	

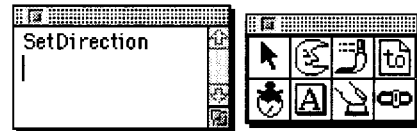
Follow the same procedure to name the second and third turtles "Car1" and "Car2" respectively.

Then go to the Procedures Page. Add the following procedures.

```
to train
engine,
  forward 1
car1,
  forward 1
car2,
  forward 1
end
```

```
to setdirection
engine,
  seth 90
car1,
  seth 90
car2,
  seth 90
end
```

Return to the Page. Enter the command `setdirection` in the Command center and press Return.



Open the engine dialog box. Enter the name of the procedure `train` as the instruction in the dialog box as shown below, and select "Many Times."

Name:	<input type="text" value="Engine"/>
Instruction:	<input type="text" value="Train"/>
Do it:	<input type="radio"/> Once <input checked="" type="radio"/> Many Times
<input type="button" value="Cancel"/> <input type="button" value="OK"/>	

Click on the engine once to start the train. The engine and both cars should move across the screen.





The movement may be a bit jerky, depending upon the speed of your computer, but this matches the movement of an actual train. Try changing `forward 1` to `forward 4` in the `train` procedure. The train should travel faster, but the movement may be jerkier.


Glen Bull is an associate professor in the instructional technology program of the Curry School of Education at the University of Virginia. Gina Bull is a computer systems engineer in the information technology and communication organization at the University of Virginia. Todd Kent is a graduate instructor in the Curry School specializing in instructional uses of technology.


Internet Addresses: GBull@Virginia.edu,
Gina@Virginia.edu, TKent@Virginia.edu

What if your students could

 **BUILD** upon the skills and concepts that LOGO teaches.

 **COMMAND** the latest computer technology with a new programming language designed specifically for kids.

 **Reach** beyond the LOGO paradigm of controlling a "turtle" to explore and control the actual interworkings of a computer.

 **Discover** a creative tool as powerful as their imaginations.

 **Orchestrate** their own graphics and sounds to produce multimedia book reports, research projects, learning games, and more!

They can. Introducing...

FUND^{DA}MENTALTM

The New Computer Programming Language for Kids

For a FREE brochure on FUNdaMENTALTM and how you can integrate it into your creative classroom, write (be sure to include your name, school and address) to:

FUNdaMENTAL Brochure
KartoffelSoft Inc.
Suite 217
Town and Country Village
Palo Alto, CA 94301

Looking Back and Looking Forward

by Douglas H. Clements and Julie Sarama

Richard Noss and Celia Hoyles have worked with Logo and mathematics longer and more than almost anyone around. They stepped back and looked at this area in an insightful piece that we will review today (Noss, 1992 #1255). [Mostly, we wish to share *their* ideas with you, but we couldn't resist including some of our own comments, enclosed in brackets.]

Noss and Hoyles argue that Logo is not an "all-purpose learning environment," believing that it gives rise to unrealistic expectations concerning the development of problem-solving skills. [We partially disagree. The research on Logo and problem solving is much more positive than the authors imply.]

What Logo *does* provide is a computer environment in which mathematics can take place; one in which certain mathematical ideas not attainable in other settings can be attained, students can pose their own questions, and in which they find pleasure working mathematically over extended periods of time. [Being picky: The authors seem to imply that the mathematical ideas are floating around somewhere and that students somehow grasp more of them with Logo. Instead, we believe that Logo can help students invent their own mathematical ideas.]

That's the ideal. Noss and Hoyles say that we must look back and ask hard questions. Were the students really "doing mathematics" or was that just a projection of the researchers' hopes? Their "hard look" revealed three big obstacles to learning mathematics in an unstructured Logo environment.

1. Unreflective use of tools and the play paradox

We want students to play with ideas before these ideas are presented formally, so they are not learned devoid of any experiential meaning. However, if students really play, they are unlikely to play with the ideas that we had in mind. That's a paradox. It is easy to use ideas in Logo to get an effect without thinking and reflecting deeply on these ideas.

2. Bypassing of Logo tools

Students do not use tools of which they are unaware. "Doing" Logo does not necessarily encourage the use of its most powerful ideas, such as recursion.

3. Avoidance of mathematical analysis

Students often prefer strategies that do not involve analysis. For example, they "home in" (e.g., rt 90 rt 20 rt 20 lt 20 rt 5 rt 5) rather than work out a solution mathematically. The authors note that people behave

similarly in everyday life. [And some "situated cognition" researchers have not only documented this fact but, we believe, celebrated it perhaps excessively.]

The authors suggest that, done correctly, Logo can help provide experiences from which students can build an intuitive mathematics. These experiences are not like "using temperature for negative numbers." This latter approach tries to hide mathematics behind real-world experiences. Instead, the authors argue that seeing interrelationships among the symbols of mathematics can *itself* be meaningful. Understanding mathematics, in other words, does not occur only through grounding the ideas in real-world experience. Understanding can also involve understanding mathematical forms and appreciating mathematics as a discourse.

So, the activities in which Logo tools are used help determine whether students will use mathematical ideas.

Students need activities that expand the circle of situations with which they can play and of which they can make sense. They need to have tools for bringing forth the mathematics in these situations. They need to connect their own mathematical inventions to traditional mathematics. Noss and Hoyles argue that Logo provides a unique setting for all these.

Why is Logo unique in this regard? First, Logo does allow students to use intuitive mathematical strategies, rather than avoid mathematics altogether. Students can solve problems at their own level of sophistication. Second, these intuitive strategies can lead to more analytic work. Students can and do debug and expand their intuitions. Logo provides an observable record of thinking that can be acted on and revised.

Let's consider an example, involving the idea of variables. Students often use multiple variables in their Logo procedures, one for each parameter that varies, ignoring relationships among them. For example, they may have four separate variables for the lengths of the sides of a parallelogram, even though opposite sides must be the same. They then run these procedures with values that satisfy the relationships. This suggests that they recognize the relationships—at least implicitly. "Putting them out there" with variable names helps them think about and recognize what they implicitly know. Often with—but sometimes without—teacher intervention, students can change their procedures to show these relationships explicitly.

Noel was working on his parallelogram procedure. He gave it four inputs for every attribute he thought he had to vary. (He understood the equality of opposite sides, but he used two variables for the adjacent turns.) After using his procedure, he brought together the meaning he had given to his code and the turtle's visual output. This play between the code and the picture helped him revise his procedure to have only one input for a turn. (The second turn was computed inside the procedure as `180 - :turn`). He did this without the teachers. So, students can do some tasks with a computer that would be impossible without one.

Symbols in Logo can allow students to erect a scaffolding around the solution of a problem. With time, they can fill in more and more parts of the solution. Switching between code and picture helps them proceed. The code captures just enough of their intended goal—even if that goal is not thoroughly analyzed—to allow students to take “mind-sized bites.” One limitation of Logo is that students have to work on the symbols. [Newer versions, such as *Turtle Math*TM, do not have this limitation.]

The authors sum up the unique contributions of Logo to mathematics learning.

- Intuitive actions arising from perceptual rather than analytical thinking can be captured in symbols.
- Logo's commands and structure are consistent with mathematics (e.g., procedures behave like mathematical functions; `fd` and `rt` reflect differential geometry).
- They can thus become objects to think about.
- Feedback allows students to compare the outcome to their intended outcome.
- Actions are documented. They can be put together and named. The symbols can aid generalization.
- Logo objects have measures associated with them that are visible, quantifiable, and formalizable.
- There is a need to make relationships explicit.
- The intuitive and the reflective exist side-by-side.

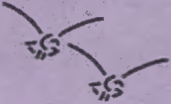
Noss and Hoyles conclude with two important observations. First, too many curriculum and research projects involve hierarchical sequences of Logo tasks. They often find no effects. This approach seems fundamentally misguided. It prevents real change and marginalizes Logo's potential for innovation. Second, time to learn programming is an even better investment if we think of programming as a medium for *expressing* mathematics, rather than as a tool for learning it.

Time to prepare this material was partially provided by the National Science Foundation under Grants No. MDR-8954664 and MDR-9050210. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

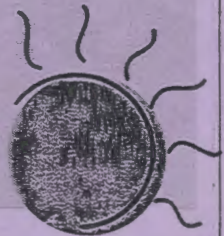
Douglas H. Clements, professor at the State University of New York at Buffalo, has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. Through a National Science Foundation (NSF) grant, he developed a K-6 elementary geometry curriculum, *Logo Geometry* (published by Silver Burdett, & Ginn, 1991). He is currently working with several colleagues on a second NSF-funded project, “Investigations in Number, Data, and Space,” to develop a full K-6 mathematics curriculum featuring Logo. With Julie Sarama, he is coauthoring new versions of Logo for learning elementary mathematics. One, *Turtle Math*, is available from LCSi.

Julie Sarama is a mathematics education doctoral student at the State University of New York at Buffalo. She has taught secondary mathematics and computer science, gifted math at the middle school level, and mathematics methods courses. Along with Douglas Clements, she is co-author of *Turtle Math* and is currently designing and programming new version of Logo for the NSF-funded Investigations project.

Douglas H. Clements and Julie Sarama
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall
Buffalo, NY 14260
Internet: clements@ubvms.cc.buffalo.edu



Introducing the next best thing to two months off in the summer.



Turtle Math and MicroWorlds Math Links from LCSi

After years of development and consultation with teachers like you, LCSi introduces **Turtle Math** and **MicroWorlds Math Links**: two math tools for teachers who want to make math exciting.

Turtle Math and **MicroWorlds Math Links** provide a true advantage over any other math software. Each easy-to-use package provides students with an invaluable exploratory environment plus dozens of activities that help them think mathematically. So they learn more about math. And that means increased satisfaction for you.

Turtle Math, designed for students in grades 3 – 6, lets students use a collection of activities and challenges in which measurement and geometry is the context for exploring various math concepts.

Aimed at students in grades 4 – 8, **MicroWorlds Math Links** is an interactive learning environment that gives students concrete ways to explore abstract ideas and visualize answers to mathematical questions.

Both packages support the NCTM Standards, and are available for Macintosh computers.

If you're interested in exploring a new standard in math teaching tools, why not call us today for a free demo disk at:

1-800-321-5646.



*And bring a little more sunshine
into your classroom*



ISTE BRINGS THE WORLD OF TECHNOLOGY CLOSER TO YOU.

By drawing from the resources of committed professionals worldwide, ISTE provides support that helps educators like yourself prepare for the future of education.

ISTE members benefit from the wide variety of publications, specialized courseware, and professional organizations available to them.

They also enjoy exciting conferences, global peer networking, and mind-expanding in graduate level distance education courses.

So if you're interested in the education of tomorrow, call us today.



International Society for Technology in Education

1787 Agate Street, Eugene, OR 97403-1923 USA

Phone: 503/346-4414 Fax: 503/346-5890

Order Desk: 800/336-5191

America Online: ISTE

AppleLink: ISTE

CompuServe: 70014,2117

Gopher: iste-gopher.uoregon.edu

Internet: iste@oregon.uoregon.edu

WE'LL PUT YOU IN TOUCH WITH THE WORLD.