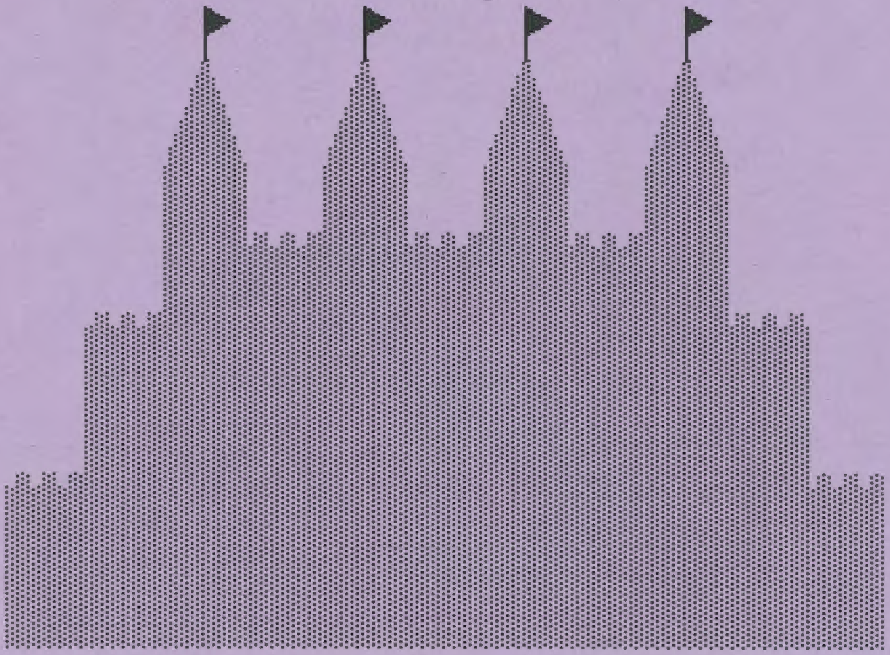
Journal
of the ISTE
Special Interest Group
for Logo-Using
Educators

LOGO EXCHANGE

Summer 1995

Volume 13 Number 4

In this issue: *Castles in the Sand*



Also—

You Gotta Have a Gimmick

Egyptian Multiplication: An Ancient Algorithm for the Modern World

International Society for Technology in Education

LX Volume 13 Number 4

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

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.

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

LOGO EXCHANGE

Volume 13 Number 4 Journal of the ISTE Special Interest Group for Logo-Using Educators Summer 1995

Contents

From the (old) Editor

From the (new) Editor

Quarterly Quantum

Musings...

Beginner's Corner

More Musings...

Logo: Search and Research

Windows on Logo

From the (old) Editor

A Long and a Short Journey

by Sharon Yoder

It's hard to believe. I'm sitting here in front of my computer writing my last *LX* editorial. At one moment it seems as though I wrote that first editorial only a short time ago; at another moment it seems ever so long ago.

Ah, but it has been a journey—an amazing journey. Much of the last 12 or 13 years of my life have been intimately tied up with Logo, and *Logo Exchange* has been an important part during the past seven years. So much has changed in those years.

When I first discovered Logo it was on an Apple II Plus and I was teaching math and computing at the junior high and high school level in Ohio. Our 48K Apples were miracles—expensive, sophisticated pieces of equipment that were few and far between in the schools. And we were so excited about the potential of Logo in our classrooms.

Building a Logo-based curriculum was an exciting adventure in the early 1980s, one that kept me busy for quite a few years. It was that adventure that put me in contact with Tom Lough, then editor of *LX* and led to my writing some articles for *LX* during its early years.

During those years my children grew to young adults and left for college. Amazing to think they were in grade school and junior high when Logo first appeared on the scene.

Along Came *LX*

I was living and working in New York City when I became editor of *LX*. Logo had just begun to change with the advent of *Object Logo*, *LogoWriter*, and *Logo PLUS*. My state of the art machine was a Macintosh SE without a hard drive, but with *two* floppy drives. I could run *Microsoft Word* on a single disk and keep the documents on that same disk.

The move to New York and my early work with *LX* and ISTE was the beginning of quite an odyssey in my life. Before that year was over, I was driving across the country with all of my possessions in a rented moving truck, with my cat for company. It was a trip I will never forget, seeing this beautiful country from ocean to ocean as I moved to my present home in Eugene.

And the odyssey continued here in Eugene. I spent a lot of time those first few years here growing to love Oregon—its beaches, its mountains, and the beautiful valley in which my husband and I live.

As the months and years went by, my responsibilities with the University of Oregon grew as I worked with graduate students preparing to be computer educators. Logo was a centerpiece of what I taught those students. At the same time, my own writing flourished, resulting in a whole series of Logo books focusing on various Logo dialects. But just about the time I became comfortable with my new job and new home, budget cuts hit Oregon and our graduate program was gone. It was a sad time for all.

There were some lean years—for awhile we only had students finishing up degrees in the canceled programs. Then our enrollments diminished even more as no new programs were started. We were all depressed and disheartened.

Looking to the Future

But, at long last, those lean years seem to be over. We have begun a new undergraduate program in education this year and are looking forward to developing the coursework for the upcoming years, which will include a specialization in technology. The future is looking very bright for us now.

And how the technology has changed over my years of involvement with LX. My (now) state of the art machine is a Power Macintosh 8100/100 AV, and even it occasionally struggles with software such as *PageMaker* and *Microsoft Word 6*. I now have a Gig drive, a CD-ROM, 32 Megs of internal memory, a scanner, a laserdisk player, and a portable hard drive that moves from home to school each day. My current equipment was science fiction back in those early Logo days.

And Logo...

But what of Logo? Unfortunately, my teaching of Logo has diminished with the completion of graduate programs in our old program. Logo will not be a part of our new program since our technology students will be going into business and industry, not schools. For the first time in more than a dozen years, Logo is no longer a central part of my work. Oh, Logo is still a good friend, but one that I only visit from time to time, and then mostly for old times' sake.

So, it's time to move on, to pass the torch to someone more intimately involved with Logo on a day-to-day basis; someone who has a better pulse on the reality of today's Logo.

That special someone to whom I am passing the editorship of *LX* is Dorothy Fitch. Many of you know Dorothy. She has been a part of the Logo community for many, many years. In more recent years you have had the opportunity to see her work in her Beginner's Corner column in *LX*. And you will certainly get to know Dorothy better as the weeks and months go by.

I know that Dorothy will bring a fresh, new perspective to *LX* and that you will enjoy her contributions tremendously. And, I suspect I will find myself "drafted" from time to time by Dorothy, so we may meet again in the pages of *LX*.

And So Farewell...

And so I bid a fond farewell to LX. A special thanks to all of you, dear readers. It has been a delight sharing my life with you, whether it was insights from my students, walks on the beach, or exciting new products. Enjoy your time with Dorothy. I will be watching!

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

From the (new) Editor

Greetings

from Dorothy Fitch

I am excited, honored, and, at the moment, overwhelmed at the thought of becoming the next *LX* editor. You may be wondering about my background and what I can bring to this position.

A Journey Begins

It is interesting how things work out. Being immersed in the development of Logo languages seems far removed from my first career of teaching elementary music. But the change came gradually and naturally. Indeed, I have found parallels between music and programming at many different levels.

To make a long story short, in 1980 the school where I taught got its first Apple II computer. Interested in what it could do, I took a course or two and found myself on the district's computer education curriculum committee. It was then that I discovered Logo. I found that computers and I got along well, and that not all teachers found that to be the case. In 1982 I took a leave of absence to try doing computer workshops for teachers. For five years I ran a small computer classroom for teachers and kids, presented teacher workshops throughout the state, and consulted with curriculum development committees in several districts.

Logo crept into everything I did. Language arts teachers found themselves exploring sentence structure using Logo. Math teachers found themselves writing simple function games. All the kids who came through the door "did" Logo. Meanwhile, a first-grade teaching colleague and I put together a set of programs and activities for young learners, called Kinderlogo. Later, I wrote a set of Logo programs for creating databases, charts and graphs, surveys, and matching games, which interested Terrapin, who published it as *Logo Data Toolkit* in 1987.

And Logo Becomes Central

Alas, the writing on the wall became clear. My little computer classroom, with its handful of Apple IIe's, was quickly becoming outdated. I couldn't afford to replace the machines with newer equipment. At that moment, a position at Terrapin opened up, and that's where I've been ever since, developing Logo materials, testing software, writing documentation, helping to design new versions of Logo, presenting workshops at conferences, and doing a little bit of everything. What a dream job, I've been told. And it's true. I can't imagine anything better!

Through my work and SIGLogo, I've gotten to know many members of the Logo community. In 1989 Sharon Yoder asked if I would write a Beginner's Column. So for the past six years (has it been that long?) I have worked with and learned a great deal from Sharon. I know I have a hard act to follow as editor of *LX*. Sharon has been very supportive during this transition, making things as easy for me as possible. I only get nervous when she says things like, "It's yours now. You're making the decisions."

I Need Your help

Now here's where you come in. I need your help to make the right decisions.

I want *LX* to have something for everyone. Some of our readers are just beginning with Logo; others are stimulated by its complexities. And with so many versions of Logo available, one challenge will be to provide useful information for all readers.

How can you help? I need your ideas, your stories, your articles, your questions, your input. No contribution is too small or too simple. Send me your quick ideas for Logo activities that anyone can try, larger-scale projects, classroom success stories, tips and tricks that you have discovered. I know you're busy, and I'll do whatever it takes to help you with your contribution. Perhaps most importantly, I need to know what you want from *LX*. This is your journal. I'm just the messenger.

When I started at Terrapin, friends asked if I didn't miss teaching. I told them that I was still in education, still working with teachers, and able to reach even more teachers and their students through my work than before. Now, as editor of *LX*, my goal is to bring together people with ideas that will influence and affect an even wider audience.

You are welcome to contact me anytime at

3 Derby Road
Derry, NH 03038
603/425-2010,
Compuserve: 71760,366
Internet: 71760.366@compuserve.com

I look forward to hearing from you!

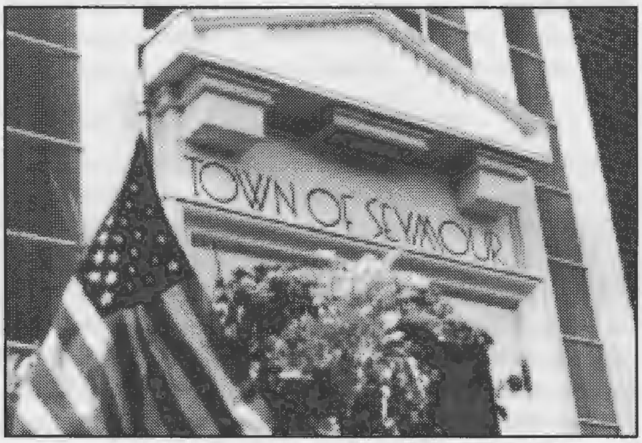
Dorothy M. Fitch

In Search of Seymour

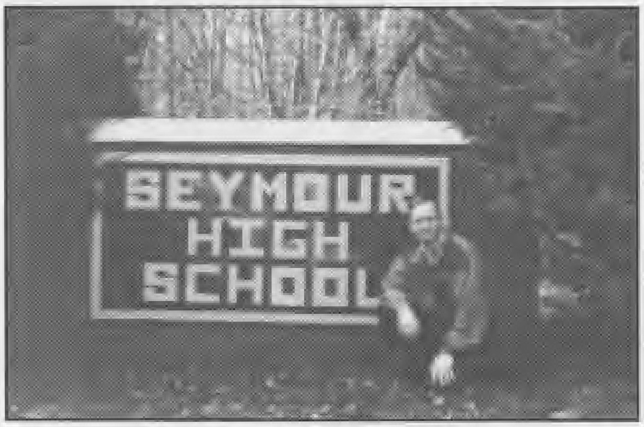
by Tom Lough

I always admired Charles Kuralt. His "On the Road" programs really spoke to me. I thoroughly enjoyed "accompanying" him to out-of-the way places to meet the extraordinary ordinary people he discovered.

Recently, while cruising the interstate, I passed a sign that said, "Seymour, Next Exit." Because I was on the way to an appointment, I had to pass that exit by. But on the way back I could not resist and turned off the highway in response to the invitation of that sign.



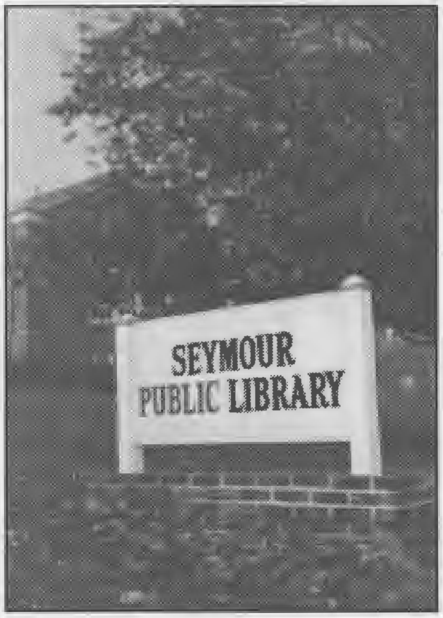
I found myself transformed from a somewhat harried traveler to a very curious and interested explorer. I slowed my car down to a crawl, totally absorbed in the scene unfolding before me. As I crossed the town line into Seymour, I somehow became a person on a quest. What was this town like? What kind of people live here? What can I learn here?



Suddenly, I realized what I could do here. I could search for Logo in Seymour! But first, I took a look around.

What fun to ride past the Seymour Town Hall and to stand on the platform of the Seymour Train Station! I delighted in driving by the Seymour Public Library and climbing the steps up to the Seymour Post Office. Other special stops included the Seymour Bargain Barn and, of course, Seymour High School.

I learned that the town of Seymour had gone through several name changes. In 1650, the area was originally



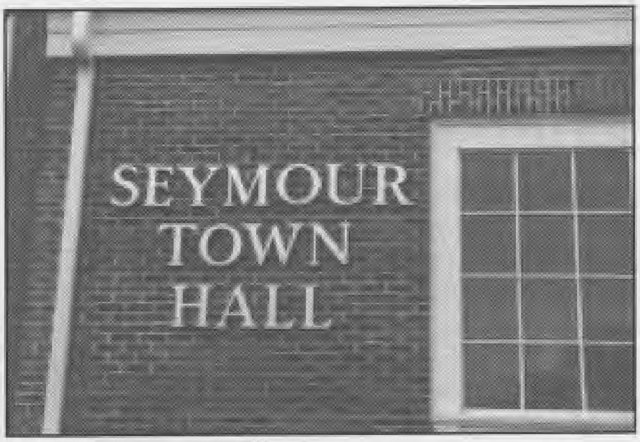
called Nawcatock, after a nearby Native American village. Then, in 1678, it was changed to Amaugsuk, meaning a fishing place where waters pour down. In 1738, it became Chusetown, to honor Chief Chuse, a scout in the French and Indian War. In 1803, the name was changed to Rimmon Falls, and again in 1805 to Humphreysville, to honor General David Humphreys, who had purchased land nearby. (Humphreys was a friend and aide to General George Washington.) Finally, in 1850, it was named Seymour, in honor of Thomas H. Seymour, who was elected the governor of Connecticut that same year.

I stopped at a Seymour craft sale that was raising money for a local museum to promote historical awareness, and began to chat with some of the people. I struck up a conversation with a young man who was selling beautiful wooden lawn ornaments. Naturally, I asked him if he had any ornaments in the shape of a turtle. He did not, but said he could make me one if I wanted.

I asked him about his school and if he had learned anything about computers. It turned out that he had. In fact, he had been taught Logo by Mr. Ted Zalenski in the sixth grade at Seymour's Anna Lopresti Elementary School. So there it was!

I later called Mr. Zalenski to chat with him a bit. He is a 26-year classroom veteran and enjoys teaching mathematics, science, reading, and spelling. He told me he had taught Logo to the young man with the TRS-80 version on a Radio Shack computer!

This odyssey had provided me with some compelling ideas and fascinating experiences. I enjoyed a relaxing diversion from a weary drive home. I had immersed myself in Seymour. And I had found Logo as well!



As teachers and as perpetual students, shouldn't we each allow ourselves an occasional diversion of whimsy? Isn't that what summers are for? Who knows what we will find ... especially if we look for it!
FD 100!

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

PS: Do you have a town named Seymour in your state or province? If so, I would love to hear about it!

PPS: This issue closes out the period of service for Sharon Yoder, who has served as your faithful editor since 1987. Sharon, thank you for your labor of love. You nurtured *LX* as it moved into its publishing home at ISTE, and you elevated it to the status of a refereed journal. We welcome Dorothy Fitch to the masthead and wish her well in the exciting years that lie ahead. FD 100, indeed!

Egyptian Multiplication: An Ancient Algorithm for the Modern World

by Robert Macdonald

The autumn of 1989 was much influenced by the ancient Egyptians. That October Judi Harris published a fascinating microworld in The *Computing Teacher* dealing with numerical hieroglyphics (Harris, 1989). She wrote a computer program that created Egyptian icons as output. These icons represented a numerical system lacking the concept of place value—although the icons could be interpreted as representing numbers that were multiples of 10. However, as Ms. Harris observed, ancient Egyptian numerals were representational not positional as our Arabic numerals are.

Further Egyptian Investigations

The Egyptian motif continued. In late October for a class Halloween party, one of my more creative students dressed as King Tutankhamen. Her costume was brilliantly designed.

Then, at the November book fair, the best selling item for my class was The Metropolitan Museum of Art's *Fun With Hieroglyphs*. An informative booklet with 24 rubber stamps of hieroglyphs, an ink pad, and a key to the hieroglyph alphabet was for us a rare find (Roehrig, 1989). Sending messages in an ancient and mysterious language occupied the class for weeks.

All was set for undertaking an intriguing adventure with an enthusiastic group of fourth graders who had *just* gained some control over Roman numerals. As the students worked through Ms. Harris's program and investigated hieroglyphs with a set of stamps, I began to formulate an activity based on Egyptian multiplication.

The Algorithm Explained

The algorithm about to be described has been attributed to numerous ethnic groups: Maygar, Hungarian, Peasant, Russian, even Egyptian. I prefer the term Egyptian as it conjures up a civilization that intrigues us all (Macdonald, 1991).

An algorithm has been defined as "some special process of solving a certain type of problem, particularly a method that continually repeats some basic process" (James, 1976, p. 8).

The patterning that takes place in this process requires that students be able to half factors, double factors, recognize odd and even numbers, and add with competence. After working through the algorithm, many students have expressed gratitude for the algorithm currently in use.

An Ancient Algorithm For Multiplying

When being introduced to the activity, students initially should be allowed to use calculators. Because the algorithm applies the skills of duplation (doubling) and mediation (halving), calculators may produce more accurate results.

Let's multiply 24 X 35.

1. Record each factor at the top of a column.
2. Divide the number to the left by 2: 35/2 = 17.5. Discard the remainder. Write 17 below 35.
3. Double the factor to the left: 24 X 2 = 48. Write 48 below 24.
4. Continue to double the right column until the quotient in the left column equals 1.
5. Place an asterisk by each odd number in the left column. (Not all will be odd).
6. Circle all numbers in the right column that are paired with the odd numbers in the left column.
7. Add all the circled numbers in the right column: (24 + 48 + 768 = 840)
8. Be certain that you consider whether your *initial* factor is odd or even. If it is odd, add its pair.

Half	Double
*35	24
*17	48
8	96
4	192
2	384
*1	768
	840

Now you can check your work by reversing the process. Remember to half (divide by two and throw away any remainder) and to double (multiply by 2).

Half	Double
24	35
12	70
6	140
*3	280
*1	560
	840

If you add the numbers in the right column paired off with the odd numbers 3 and 1 (280 + 560), the sum will equal 840. The work checks out. It might be prudent the first few days of working on this algorithm to keep the halving and doubling on the same side. No need to confuse unduly.

Practice the Algorithm

A worksheet found at the end of this article can be used to provide practice with this algorithm. A *LogoWriter* computer program is also included. It will output the algorithm in its entirety. This will allow students a way of checking their own work while building up their skills with little interaction from the teacher.

The command to run the computer program is **choose**. You will be asked which side you wish to half and double. Then you are asked to input your first factor (touch the Return key), and then to input your second factor (touch the Return key). The entire algorithm will be reproduced on the page. If the work scrolls off the screen, simply use an the Up key to gain access to the page and then use the appropriate arrow key to move up to bring the scrolled material into view. Remember to use the Down key to return to the Command Center.

Odd factors that are halved are indicated by asterisks. The doubled factors paired with those odd factors will be added by the computer to provide the product.

Because the ancient Egyptians did not use a zero as a factor, the program will not accept such an unauthentic entry. (The concept of nothing—or zero—was not available in many older civilizations. This is a related issue that might be fun to investigate as a class.) The Command Center will then provide directions for continuing the program.

In writing the program, procedures such as **integer** and **intquotient** had to be provided for *LogoWriter*. The procedures test for odd numbers in either column. In halving the factors, a test for an integer was necessary in order to discard any remainder in the quotient.

The Computer Program

Start the *LogoWriter* program with the command **choose**.

```
to choose
clearpage
repeat 4 [pr []]
(insert [To half your first factor
   and double the second, reply with
   (Y)])
(print [.])
print []
(insert [To double your first factor
   and half the second, reply with
   (N)])
(print [.])
ifelse readchar = "Y [egypt]
   [egypt1]
end

to egypt
clearpage
initialize
enter
test
math.figure
final
end

to egypt1
clearpage
initialize
enter
test1
math.figure1
final1
end

to clearpage
if not front? [flip]
rg
ht
ct
cc
end

to initialize
make "factor1 0
make "factor2 0
make "count1 0
make "count2 0
end

to enter
print [Enter your first factor.]
make "factor1 readnumber
print [Enter your second factor.]
make "factor2 readnumber
if :factor1 < 1 [caution ]
if :factor2 < 1 [caution ]
print []
end
```

```
to caution
print []
print [*******************]
insert [To be authentic, each factor
   should be greater than 0 ]
print [.]
print [*******************]
print []
continue
stopall
end

to readnumber
output first readlist
end

to math.figure
if :factor1 < 1 [stop]
(print [First factor is] :factor1
   [;the second] :factor2)
make "factor1 intquotient :factor1
make "factor2 :factor2 * 2
test
math.figure
end

to math.figure1
if :factor2 < 1 [stop]
(print [First factor is] :factor1
   [;the second] :factor2)
make "factor1 :factor1 * 2
make "factor2 intquotient :factor2
test1
math.figure1
end

to intquotient :number
output integer :number / 2
end

to test
ifelse even? :factor1
   [make "count1 :count1 + :factor2
   insert [-]]
   [make "count2 :count2 + :factor2
   insert [*]]
end

to test1
ifelse even? :factor2
   [make "count1 :count1 + :factor1
   insert [-]]
   [make "count2 :count2 + :factor1
   insert [*]]
end

to even? :number
output member? (last :number)[0 2 4
   6 8]
end
```

```
to integer :number
output ifelse member? ". :number
   [integer butlast :number]
   [:number]
end

to final
print []
(print [The product of your problem
   is] :count2)
print []
print [The Egyptians added all
   second factors]
print [in lines that begin with
   asterisks.]
continue
end

to final1
print []
(print [The product of your problem
   is] :count2)
print []
print [The Egyptians added all first
   factors ]
print [in lines that begin with
   asterisks.]
continue
end

to continue
cc
show [To continue type the command
   CHOOSE and touch the RETURN key.]
end
```

References

Harris, Judi (1989). Place value — place/hieroglyphics AHA!'s, *The Computing Teacher* (XVII, No. 2), 40-42.

James, Glenn (1976). *Mathematics Dictionary.* (4th ed.) New York: Van Nostrand Reinhold Co.

Macdonald, Robert (1991). An old algorithm for multiplication," *Clime News* (Autumn, 1991), 12-13. This article is a revised and expanded version of that originally published by *Clime News.* Our thanks are expressed to Mr. Ihor Charischak, the editor of *Clime News,* for granting permission for publishing the revision.

Roehrig, Catharine (1989). *Fun with hieroglyphs.* New York: The Metropolitan Museum of Art and Viking.

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

WORKSHEET

Name______________________________________ Date ________________

Two factors have been selected in each of the following problems. You are to half one, double the other until you reach 1 in the halved column. In halving forget about remainders. But do keep track of whether the halved factor is an odd (*) or even (-) number. You will sum all the paired doubled factors of the odd-numbered halved factors to get a product. Never use 0 as a factor.

1.

Half	Double
16	32
8	64
4	128
2	256
*1	512

The product is ______________________

2.

Double	Half
43	18
86	9*

The product is ______________________

Double	Half
56	34

The product is ______________________

Half	Double
*9	8

The product is ______________________

Half	Double
312	59

The product is ______________________

Double	Half
10	10

The product is ______________________

A Monster of the Day

by Laura M. Halleen

The Worst
When singing songs of scariness
Of bloodiness and hairyness
I feel obliged at this moment to remind you
Of the most ferocious beast of all:
Three thousand pounds and nine feet tall—
The Glurpy Slurpy Skakagrall—
Who's standing right behind you!
—Shel Silverstein

The above poem is a reminder of what children love the most (or the worst)—something scary, silly, and unexpected. Something resembling the Glurpy Slurpy Skakagrall may be soon present in your classroom when you use Logo in the way described in this article. This lesson can be used for the third or fourth grade. It can be integrated with a whole language approach for language arts, or used as an enhancement for reading (syllabication), science, art, and even for "snack time." It can be fun and educational for a number of reasons:

- It is well-suited for group or individual work.
- It is equally appropriate for Logo beginners and experts.
- It allows the learners to gain mastery in syllabication.
- It provides an opportunity for personal expression.
- It provides practice in looking at three main parts of animals—the head, the body, and the legs.
- It allows the learner to see his or her perception of the "monster" in written and illustrated form.
- It provides practice in answering questions asked by the computer to "discover" the monster of choice.

The lesson is introduced by explaining to the class that today is Friendly Monster Day. The class then brainstorms as many animals as they can think of whose names have three syllables. The names can be one word or two. The names are then divided into three syllables:

croc/o/dile
el/e/phant
cock/a/too
jack/rab/bit
bil/ly/goat
sea/li/on
kan/ga/roo
chim/pan/zee
pan/da/bear

The students then use the three syllables of each animal to create a new animal. The new animal is called Friendly Monster. This monster will have three main parts that correlate with the three syllables: the first syllable is the head, the second syllable is the body, and the third syllable is the legs. The following Logo commands are used to create a "Friendly Monster."

```
TO FRIENDLY.MONSTER
HT
OUTPUT (SENTENCE HEAD BODY LEGS)
END

TO HEAD
OUTPUT PICK [CROC EL COCK JACK BIL
   SEA KAN CHIM PAN]
END

TO BODY
OUTPUT PICK [O E A RAB LY LI GA PAN
   DA]
END

TO LEGS
OUTPUT PICK [DILE PHANT TOO BIT GOAT
   ON ROO ZEE BEAR]
END

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

To get a list of 15 mixed up animals, type

```
REPEAT 15 (PRINT FRIENDLY.MONSTER)
```

in the Command Center.

The following is an example of 10 random friendly monsters.

pangatoocockpanzee
billytoochimlibear
sealibitjackrabbit
bilagoatbilpantoo
cockezeeellytoo

Nine out of ten of these monster names can be very odd looking. But, now is when the fun can really begin. Each child is invited to each pick one animal to have as his or her friendly monster. The children divide the new animal into three syllables and figure out which three animals their animal came from. This gives them an idea of what it will look like. The students then give their monster a name and draw a picture on poster board. It is always more fun for the students to choose the name themselves—"I choose 'seadagoat' and named it Sammy Seadagoat—alias: Sammy 'See the Goat.'"

After they have all learned what their new friend looks like and are on a first name basis with it, the students need to get to know it a little better.

Next, have the children read the following poem:

If the World Was Crazy
If the world was crazy, you know what I'd eat?
A big slice of soup and a whole quart of meat,
A lemonade sandwich, and then I might try
Some roasted ice cream or a bicycle pie,
A nice notebook salad, an underwear roast,
An omelet of hats and some crisp cardboard toast,
A thick malted milk made from pencils and daisies,
And that's what I'd eat if the world was crazy.
If the world was crazy, you know what I'd wear?
A chocolate suit and tie of eclair,
Some marshmallow earmuffs, some licorice shoes,
And I'd read a paper of peppermint news.
I'd call the boys "Suzy" and I'd call the girls "Harry"
I'd talk through my ears, and I always would carry
A paper umbrella for when it grew hazy
To keep in the rain, if the world was crazy.
If the world was crazy, you know what I'd do?
I'd walk on the ocean and swim in my shoe,
I'd fly through the ground and I'd skip through the air,
I'd run down the bathtub and bathe on the stair.
When I met somebody I'd say "g 'bye, Joe,"
And when I was leaving — then I'd say "Hello."
And the greatest of men would be silly and lazy
So I would be king... if the world was crazy.
—Shel Silverstein

After having some fun with this poem, each student has a chance to add another stanza to the poem by describing their friendly monster. They would start the stanza with:

If the world was crazy, you know what I'd see?
A friendly monster staring right at me!

Students then get help from Logo. The students are given an explanation of how Logo will help them get colorful ideas and creative thoughts about their monsters. The following is sample text with potential responses from the students.

HELLO! I would like to learn about your friendly monster you have created. Please answer the following questions with incomplete sentences. After answering the first question, you may type in any of the following number words: one through seven. You may need to clear text on each page before going to the next question.

PLEASE TYPE YOUR NAME TO BEGIN.

Laura Halleen
Have a fun adventure with your monster,
Laura Halleen

What is your monster's full name?

Sammy Seadagoat
That's a very clever name, Sammy Seadagoat.

If I met your monster on the street, how would I know it was your friendly monster? Please tell me what animal parts it has: head, body and legs.

A sealion head, a panda bear body, and goat legs
WOW! That is really an interesting combination, a sealion head, a panda bear body and goat legs.

Where does your monster live?

In a deep, dark cave in the bottom of the ocean

That sounds lovely. I'm sure your monster is very comfortable there in a deep, dark cave in the bottom of the ocean.

What does your monster like to eat?

Fish with spinach sauce and bamboo shoots for dessert

Mmmm! Sounds very delicious. I would like to try some fish with spinach sauce and bamboo shoots for dessert.

What is your monster's favorite thing to do?

Sleep during the day and chase anything that moves during the night

I sure hope it gets a chance to do that often. It is always fun to sleep during the day and chase anything that moves during the night.

What is your monster afraid of?

Cute kitty cats and homework

Yes, sometimes I am afraid of cute kitty cats and homework.

Who is your monster's best friend?

Suzy Sealibit

Best friends are very special and so is Suzy Sealibit.

Now the students are ready to finish their poem.

Here is a program that was developed to produce output like that given above.

```
TO BLANK LINE
PRINT [ ]
END

TO HELLO
HT
PRINT [HELLO! I would like to learn
   about your friendly monster you
   have created. Please answer the
   following questions with
   incomplete sentences. After
   answering the first question, you
   may type in any of the following
   number words: one through seven.
   You may need to clear text on
   each page before going on to the
   next question. PLEASE TYPE YOUR
   NAME TO BEGIN]
BLANK.LINE
PRINT SENTENCE [Have a fun adventure
   with your monster] SENTENCE
   READLIST ".
BLANK.LINE
END

TO ONE
PRINT [What is your monster's full
   name?]
BLANK.LINE
PRINT SENTENCE [That's a very clever
   name,} SENTENCE READLIST ".
BLANK.LINE
END

TO TWO
PRINT [ If I met your monster on the
   street, how would I know it was
   your friendly monster? Please
   tell me what animal parts it has:
   head, body and legs.]
BLANK.LINE
PRINT SENTENCE [WOW! That is really
   an interesting combination,]
   SENTENCE READLIST ".
BLANK.LINE
END

TO THREE
PRINT [Where does your monster
   live?]
BLANK.LINE
PRINT SENTENCE [That sounds lovely.
   I'm sure your monster is very
   comfortable there] SENTENCE
   READLIST ".
BLANK.LINE
END

TO FOUR
PRINT [What does your monster like
   to eat?]
BLANK.LINE
PRINT SENTENCES [Mmmm! Sounds very
   delicious. I would like to try
   some] SENTENCE READLIST ".
BLANK.LINE
END

TO FIVE
PRINT [What is your monster's
   favorite thing to do?]
BLANK.LINE
PRINT SENTENCE [I sure hope it gets
   a chance to do that often. It is
   always fun to] SENTENCE READLIST
   ".
BLANK. LINE
END

TO SIX
PRINT [What is your monster afraid
   of] SENTENCE READLIST ".
BLANK.LINE
PRINT SENTENCE [Yes, sometimes, I am
   afraid of] SENTENCE READLIST ".
BLANK.LINE
END

TO SEVEN
PRINT [Who is your monster's best
   friend?]
BLANK.LINE
PRINT SENTENCE [Best friends are
   very special and so is] SENTENCE
   READLIST ".
BLANK.LINE
END
```

As these procedures demonstrate, endless questions may be asked to help the children explore their monster, and they can develop very creative answers.

Many other follow-up activities can be done with their friendly monster. Below are a few suggestions:

1. Illustrate each monster on Logo using turtle graphics.
2. Each student (or small groups) uses a different story starter, makes up their own story using the word processor, and illustrates it. Make books from the completed stories and drawings and display them.
   Story Starters
   - Once a big, purple, furry monster followed me to school. It wanted to come to my class so that it could....
   - My friend and I were walking to the store one day when we heard a strange noise. In a tree overhead there was...
   - On my birthday, my parents gave me a huge box. It was almost as big as my room. When I opened it, inside there was...
   - When I visited the zoo, I suddenly heard a zookeeper shout, "Watch out! It's behind you!" When I turned around, I saw...
3. Write a language experience story as a class. Write one of the story starters on the board and read it to the class. Ask for a volunteer to tell the next one or two sentences of the story. Write the child's contribution on the board (or use a flat panel display and a word processor). Keep asking for volunteers to continue the story until the children lose interest or you feel that the story is finished. Each child can illustrate whatever scene from the story he or she chooses. Let each child read a sentence from the story into a tape recorder. Play the tape and have the children point out their illustrations at appropriate times.
4. Create a friendly monster model using various craft material (especially recyclable). Display the finished monsters on a table.
5. Make a Monster Menagerie. Using magazines, have each child find three animals that make up their monster and cut it out. Spread butcher paper out on the floor and sketch a zoo scene on the paper and color it. Glue the monster pictures to the paper in appropriate arrangements.
6. Make Monster Cookies. Enjoy them at the end of the Monster Day with lights off, "monster music", and some "ugly" juice. The following recipe makes 25 cookies.

Ingredients:
3/4 c. margarine
1 3/4 c. sugar
2 eggs
1/2 t. vanilla
1/4 t. almond extract
1 1/2 t. salt
6 t. baking powder
5 c. flour
1/2 c. milk
Red, yellow, blue , and green food coloring
Different colored sugar sprinkles
Small chocolate chips.
Raisins

Steps:

1. Beat together the margarine and sugar. Add eggs, vanilla, and almond extract. Mix well.
2. Sift salt, baking powder, and flour together. Add dry ingredients and milk alternately in thirds to sugar mixture, ingredients are combined.
3. Divide dough into at least four parts. Add 1/2 t. of a different food coloring to each. If dough is sticky, add flour until it is easy to handle.
4. Preheat the oven to 350 degrees. Have the children take teaspoonfuls of different-colored dough and mold it into monster shapes.
5. The children sprinkle cookies with sugar sprinkles and use chocolate chips and raisins as eyes and other features.
6. Bake cookies on lightly greased cookies sheet for 8 to 10 minutes, or until lightly browned.

GRRRR...EAT !!!!

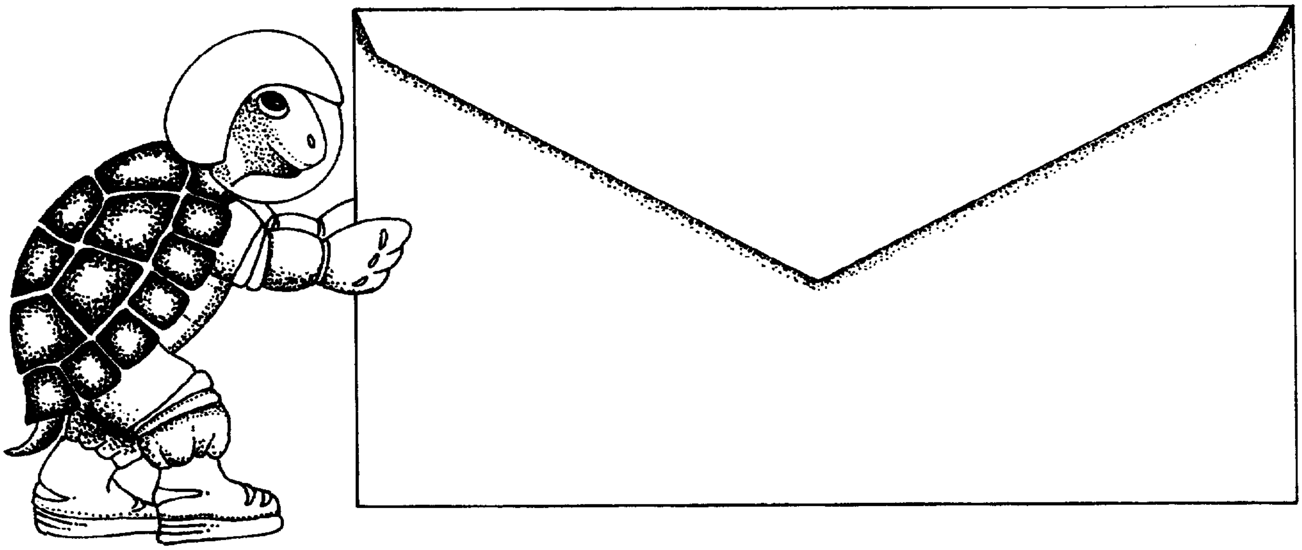
The ideas can be endless for a "monstrosity" of a day. Who knows...this day may be so exciting that you may never get a poem written quite like this one.

It's Dark in Here
I am writing these poems
From inside a monster,
And it's rather dark in here.
So please excuse the handwriting
Which may not be too clear.
But this afternoon by the monster's cave
I'm afraid I got too near.
And I'm writing these lines
From inside a monster,
And it's rather dark in here.

—adapted from Shel Silverstein
"Where the Sidewalk Ends"

Reference

Silverstein, Shel (1974). *Where the sidewalk ends.* Harper & Row: N.Y. Special Days. Macmillan Seasonal Activity Packs (1985). Macmillan Educational Co., 10, 11, 12, 17

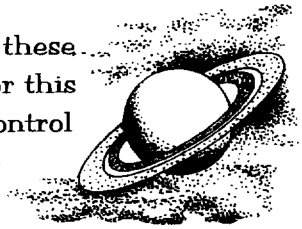


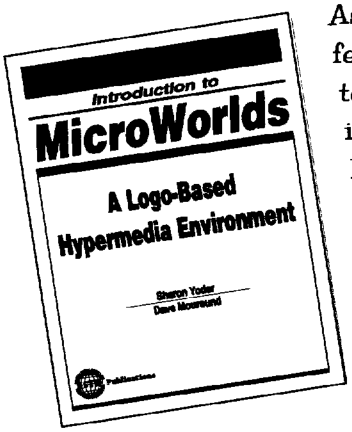
Push the envelope with MicroWorlds

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.





Introduction to
MicroWorlds
A Logo-Based
Hypermedia Environment
Sharon Yoder
Dave Moursund

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

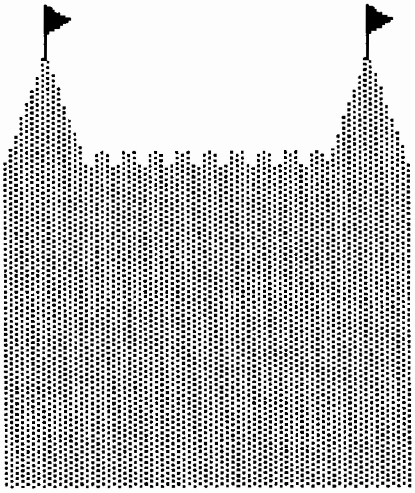
Castles in the Sand

by Dorothy Fitch

Beginner's Corner

Summertime...and the living is easy. Take some time at the beach this summer to build a sandcastle. It will get you ready for a fun Logo project to start off your school year. In this column we'll build some Logo sandcastles—no fuss, no mess, no waves.

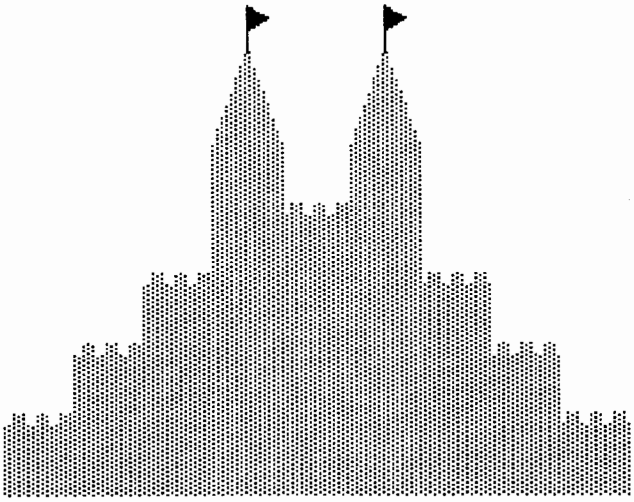
The program we'll explore is not very complicated. Use it "as is" with younger students or have older ones add their own procedures. Gear up your creativity and your nimble fingers—leave your bucket and shovel behind.



The Concept

The idea is to let kids construct a castle of their own design using a set of building blocks. Using single keystroke commands, they can create a castle out of their own building block shapes or the ones provided.

This project can serve as a beginning computer activity for almost any age, an introduction to Logo, or a bridge from a single keystroke Logo program to the real thing. Like all "instant" Logo programs, it has its limitations. However, when children do feel restricted, it's a sign that they are ready for a complete Logo environment, where they can achieve exactly what they want.



The Program

This program is written using Terrapin's Logo Plus for the Macintosh. Most of the commands used here are standard, though you may need to twiddle an instruction here and there for other versions of Logo.

The Main Procedure

The CASTLE procedure sets up colors and patterns. First we clear the screen, set the pencolor to brown (to look like sand) and the penpattern, for drawing and filling, to look granular. Then we call the procedure that does all the work, KEY. The input to KEY is a keystroke you type at the keyboard, reported by the primitive RC. The command UPPERCASE ensures that "s" and "S" are treated the same.

```
TO CASTLE
CLEARGRAPHICS
SETPC BROWN
SETPPATTERN 23
KEY UPPERCASE RC
END
```

The Workhorse Procedure

The procedure KEY looks at our keystroke, stored as the input variable :LETTER, and decides what to do with it.

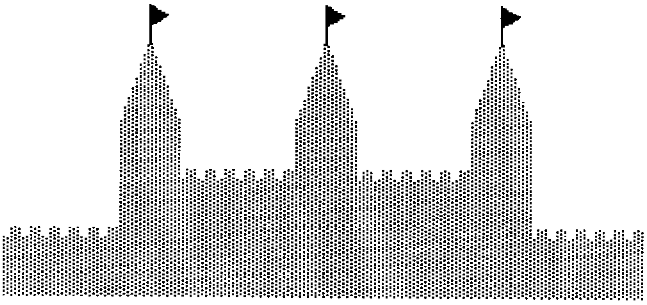
```
TO KEY :LETTER
IF :LETTER = "S [SQUARE]
IF :LETTER = "T [TOWER]
IF :LETTER = "F [FLAG]
IF :LETTER = "P [PARAPET]
IF :LETTER = "X [CLEARGRAPHICS]
IF :LETTER = "E [PENERASE]
IF :LETTER = “D [PENPAINT]
IF :LETTER = CHAR 27 [STOP]
IF :LETTER = CHAR 28 [PENUP SETX
   XCOR - 30 PENDOWN]
IF :LETTER = CHAR 29 [PENUP SETX
   XCOR + 30 PENDOWN]
IF :LETTER = CHAR 30 [PENUP SETY
   YCOR + 30 PENDOWN]
IF :LETTER = CHAR 31 [PENUP SETY
   YCOR - 30 PENDOWN]
KEY UPPERCASE RC
END
```

Pressing S, T, F, or P will draw a castle part. The letter X clears the screen, E sets the pen to erase mode, and D stops the penerase mode so the turtle will draw again. Stop the program by pressing <Esc>, character 27.

Each key on the keyboard has an ASCII (American Standard Code for Information Interchange) character code. We can use this code to access special keyboard keys. To find the ASCII code for a key, type PRINT ASCII RC and then press the key whose code you want to know.

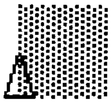
Characters 28-31 are the arrow keys. Pressing the Up arrow key (CHAR 30) moves the turtle with its pen up toward the top of the screen. SETY positions the turtle 30 steps above its current position, which is reported by YCOR. (SETY YCOR + 30 could have been written as SETHEADING 0 PENUP FORWARD 30 PENDOWN.) Similar instructions make the other three arrow keys useful for moving the turtle around.

Finally, the KEY procedure is run again (an example of recursion). The next key you press becomes the input to the next copy of the procedure KEY.



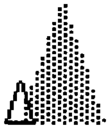
Sandcastle Building Blocks

Here are four simple shapes to get you started on your building project.

SQUARE 

This is the basic building block, a simple filled-in square with no border. (Your version of Logo may use the command FILL.)

```
TO SQUARE
HIDETURTLE
FILLSH [REPEAT 4 [FORWARD 30 RIGHT
   90]]
SHOWTURTLE
END
```

TOWER 

You could use a regular triangle shape, such as REPEAT 3 [FORWARD 30 LEFT 120] instead of the more involved instruction below, but I thought that a taller shape might look better on our castle.

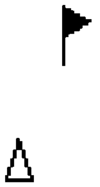
```
TO TOWER
HIDETURTLE
FILLSH [RIGHT 20 FORWARD 44 RIGHT
   140 FORWARD 44 RIGHT 110 FORWARD
   30 RIGHT 90]
SHOWTURTLE
END
```

PARAPET
4 parapets in a row:

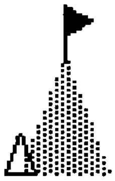
The turtle begins at the left edge of the parapet and ends at the right edge. This is the only procedure that is not state-transparent (where the turtle starts and ends in the same position and heading). However, as I played with the program, I found it much easier to make a row of parapets when the turtle was already in place to draw the next one.

```
TO PARAPET
RIGHT 90
FORWARD 2.5
LEFT 90
FILLSH [REPEAT 4 [FORWARD 5 RIGHT
   90]]
RIGHT 90
FORWARD 10
LEFT 90
FILLSH [REPEAT 4 [FORWARD 5 RIGHT
   90]]
RIGHT 90
FORWARD 10
LEFT 90
FILLSH [REPEAT 4 [FORWARD 5 RIGHT
   90]]
RIGHT 90
FORWARD 7.5
LEFT 90
END
```

FLAG

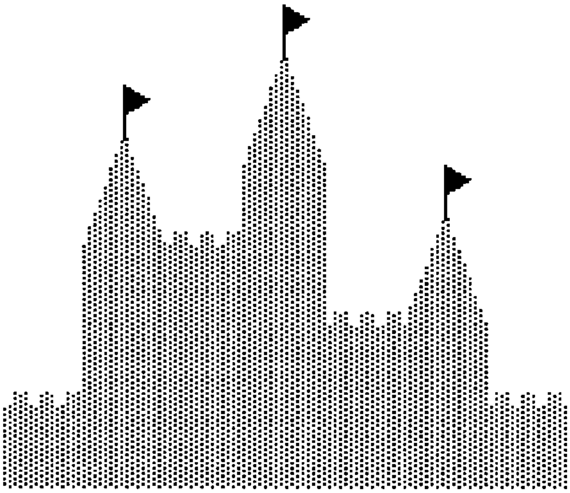


TOWER and FLAG



This tiny flag is designed to go on top of the tower. The turtle starts at the same point to draw both the flag and the tower. The turtle draws the flag with a bright red pen in a solid penpattern.

```
TO FLAG
PENUP
RIGHT 90
FORWARD 15
LEFT 90
SETPPATTERN 1
SETPC RED
FORWARD 40
PENDOWN
FORWARD 10
FILLSH [REPEAT 3 [FORWARD 10 RIGHT
   120]]
PENUP
BACK 50
LEFT 90
FORWARD 15
RIGHT 90
PENDOWN
SETPPATTERN 23
SETPC BROWN
END
```



Castle Activities

So now that you have all the tools ready, your students can build a sandcastle. Have them start the program by typing CASTLE. They'll probably want to begin by trying out all the building blocks.

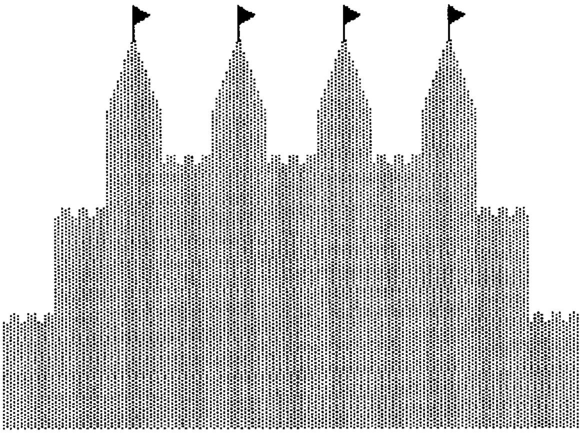
Have each student build a sandcastle and then print out the design. This gives you a whole collection of sandcastles to use on bulletin boards or for the first activity below.

Here are some more activities to try. Have your students:

- try to recreate sandcastles designed by other students;
- make a 3-D model of their castle using sugar cubes, building blocks, or solid shapes they make themselves;

- create other building block procedures and help them add the instructions to the KEY procedure that will run their new procedures;
- run the procedures from Logo, not using the single keystroke program (they'll have to move the turtle around the screen without using the arrow keys);
- study pictures of real castles; (Can they use the sandcastle program to build same design in Logo? If not, why not? What other building blocks do they need?);
- enhance their pictures by adding a sun, water, boats in the background, gulls, and so on.

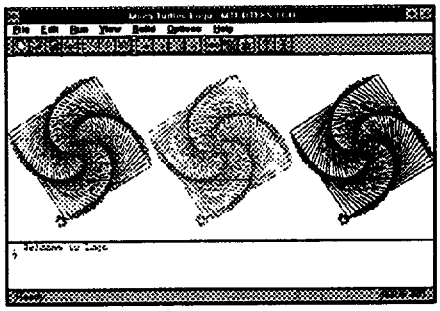
Happy Logo adventures!



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 in-service 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. She will be the editor of *LX* beginning with the Fall 1995 issue.

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

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



LogoLand

by Peggy Walkey

Once upon a time, not so long ago, there existed a magic land called LogoLand. It was into this new and scary land, a girl named Peggy was thrown. One minute she had been at HOME and the next moment she was in LogoLand. She knew she was not alone here, that others had been here and had come back safe, but she did not think she would be so lucky. Everything was so different here, even a language she did not know was spoken. How would she ever ESCAPE and return HOME again?

Peggy was so frightened she was afraid to do anything for awhile, even open her eyes. When she finally did move, there before her stood a large white turtle. She had never seen a white turtle before, so she became more curious than scared. Peggy decided that she would talk to the turtle. It did not look mean, and everyone knows how slow turtles are, so she could always run away if this turtle became nasty. Peggy went up to the turtle and said, "Hi, my name is Peggy, what is your name?" The turtle did not move or even look at her. It just said,

```
I don't know how to hi.
```

Peggy tried shouting hello, but the turtle only responded,

```
I don't know how to hello
```

Peggy thought maybe the turtle could speak French and said, "Bonjour," but the turtle only replied,

```
I don't know how to Bonjour
```

Peggy began to get angry so she pushed and she pulled at the turtle, but still it sat there. Peggy worried how she would ever go home if she couldn't make the turtle understand. Peggy was so tired from all this shouting, pulling, pushing, and worrying that she sat down beside the turtle to rest. Soon she fell into a deep sleep.

While she was asleep, Peggy had a very strange dream. Perhaps because this was a magical place—she dreamed she was a captain in an army and that she was in COMMAND of many soldiers. Peggy was startled when she saw that all the soldiers had turtle faces. "What an awful dream, I am not watching any more Ninja Turtle cartoons when I get home...if I ever get home," she cried.

Peggy soon became bored, so she thought she would play a game with the turtle. She would be an army captain and show that turtle who was the boss. First she told the turtle, that she was in COMMAND and he had to obey her. "Stand up," she shouted, "sit down," she cried, but the turtle still just sat there and kept saying

```
I don't know how to stand
I don't know how to sit.
```

"FORWARD march, you silly turtle," she cried out in anger, "this is a dumb game if you will not play."

Just then the turtle began to shake, ever so slightly, and then stopped and said,

```
I don't know how to march
```

Peggy jumped back in surprise, "You can move you silly turtle, but what was it was I said? FORWARD march," she cried. The turtle shook again, but just as suddenly stopped and replied,

```
I don't know how to march
```

Peggy knew then that the magic word must be FORWARD. "FORWARD TURTLE," she said, but the turtle just shook a little bit and then said,

```
I don't know how to turtle
```

Peggy thought and thought, then she suddenly remembered that game, the one she had played when she was little, "Mother, May I." In that game "Mother" always told you exactly what you could do in a very precise manner. Peggy told the Turtle "FORWARD 2." The turtle began to shake and then took two steps forward.

"I did it, I did it," Peggy jumped up and down, she was so excited. She wondered if she could make the turtle do more things. "BACK 2," she said. The turtle when BACK two steps. "Jump 2 steps Turtle," Peggy said, but the turtle did not move. Peggy began to get mad again, "If you do not do what I say I'll turn you on your FLIP side and CONTROL you. You won't ever be able to move again."

No sooner had Peggy said that, she was then flying through the air, turning and turning and finally crashing down onto the ground. The turtle was nowhere to be seen. Peggy was all alone in LogoLand and scared all over again. "Oh Mr. Turtle I'm sorry, I wouldn't really turn you on your FLIP side. I didn't mean it!" Peggy once again was flying through the air, and then she fell, landing right beside the turtle. She was so glad to see the turtle, she almost hugged him.

"I am so glad to see you, Mr. Turtle, I would not hurt you, you know. I was just mad and so scared. This is a very scary place for me." Peggy was getting very sleepy. Just before she dozed off she promised the turtle, "When I wake, Mr. Turtle, I will teach you a new game called 'HIDE and seek'."

Peggy fell fast asleep and did not hear the turtle laugh and say, "No my friend I will teach you a game called, HIDETURTLE, and many more games that are lots of fun. I will SHOW you how wonderful and exciting my world can be. There are many surprises for you to discover, now that you are here, and you will love them all. One day there might come a time, when you do not want to go HOME at all, but will want to stay and play with me forever, in LogoLand ."

The END

Peggy Walkey
33 Iroquois Crescent
Lethbridge, Alberta
T1K 5J3
403/381-0532

More Musings...

A Magnificent Figure-Making Machine

by Robert Macdonald

Probably the first aspect of Logo most of us encounter is turtle graphics. Making the turtle go forward or back, turn right or left, introduces us to our first Logo primitives. Either in the Command Center or on the flip side of the page in *LogoWriter*, we give commands that produce designs.

Continuing our work we quickly move into the construction of polygons—triangles, squares, rectangles, pentagons, etc. We juxtapose these figures as we please on a screen. It is all part and parcel of initial Logo experimentation.

The encounter described in this column evolved over a long period and was frequently used early in the year with students having limited experiences in Logo. It was an easy way to have students produce some rather intricate designs with a minimum of effort. The work of these beginning students was displayed at early fall open houses where parents were provided an educational outline of the academic year.

Join me as we recreate the experience of making a Magnificent Figure-Making Machine.

The Problem and a Solution

What is needed to begin is a program that will build up a larger figure from a group of smaller, similar figures. The product of this program can then be displayed on a computer screen or in printed form.

First a subprocedure that can produce a multitude of differing regular polygons is needed. The following "classic" procedure with two variable inputs satisfies that need:

```
to polygon :sides :length
st
repeat :sides [forward :length right
   (360 / :sides)]
ht
end
```

The subprocedure is fairly straightforward. The command **polygon** requires two inputs: the first designates the number of sides desired in the regular polygon, the second provides the size (length) of each of those sides in turtle steps. Note particularly in the body of the procedure the manner in which the exterior angles of the polygon are determined: 360° is divided by **sides**. Examples of this sort are well worth remembering when undertaking a study of the Total Trip Theorem. Try the following examples:

```
polygon 3 50
```

An equilateral triangle is drawn. Each side is 50 turtle steps in length.

```
polygon 4 40
```

A square is drawn with sides equal to 40 turtle steps.

```
polygon 5 45
```

This will produce a regular pentagon, each side of which has a length of 45 turtle steps.

Continue experimenting with some of the following:

```
polygon 2 60
polygon 7 50
polygon 30 40
```

Creating a single polygon isn't particularly exciting. What if we were to repeat the same figure around a central point as many times as we wished? Incorporating the polygon procedure in a twirling procedure, things instantly become more intense.

```
to twirl :sides :length :repetitions
repeat :repetitions [(polygon :sides
   :length) right (360 /
   :repetitions)]
end
```

The command **twirl** requires three inputs. The first two replicate the inputs to polygon. The third and final input provides the number of times you wish the polygon distributed around a central point. Again in the body of the procedure note 360° divided by **repetitions**. Try some of the following examples:

```
twirl 3 50 5
twirl 4 40 8
twirl 5 45 4
twirl 2 60 8
twirl 7 50 6
twirl 30 40 3
```

Enhancements

LogoWriter has some additional features that may enhance these figures: background color and pencolor. Depending on the computer, you have a differing number of available colors. The Apple IIe, IIc may provide telling effects with six colors. The Apple GS has an array of sixteen. The Macintosh has an enticing 256.

To select a background color, use the primitive **setbg** with a numerical input representing the selected color. Depending on the potential of your computer, try some of the following:

```
setbg 7
setbg 15
setbg 134
```

Check the documentation of *LogoWriter* for your computer (Yoder, 1990).

Those using a Macintosh may bring up a color screen with the command **help "colors.** It is a useful command if you have as much difficulty as I remembering 256 colors and their numerical identifiers. You can clear this screen with the Escape or Clear key.

Not only can you change the background color, but with the primitive **setc** followed by a numerical input identifying the color, you have control over the color with which your turtle will draw your figure. Depending on your computer, experiment with some commands:

```
setc 4
setc 14
setc 24
```

Note how your turtle changes color. This color is a reflection of its pen color.

Let's put all of this together. First write a **clearpage** procedure:

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

Go to the front page of *LogoWriter*. Be aware of what happens when you enter each line below in the Command Center.

```
clearpage
setc 15
setbg 8
twirl 8 20 10
```

Be sure that the inputs to **setc** and **setbg** differ. If not, you will not be able to see your figure.

A Computer Display of Twirl

Experimenting with pen and background colors in conjunction with the twirl procedure may produce some pleasing or perhaps some jarring presentations. Record them. (See the Worksheet accompanying this article.) It is feasible to gather your more interesting displays in procedures. You may give the procedure any name you wish. For example:

```
to one
setc 15
setbg 8
twirl 8 20 10
end
```

The command to run is **one.** Perhaps creating such a simple procedure could be an introduction to Logo programming. Providing a simple template with an explanation of such reserve words as **to** and **end** would be appropriate.

Let's continue:

```
to two
setc 2
setbg 11
twirl 5 35 8
end

to three
setc 1
setbg 15
twirl 3 40 20
end

to four
setc 42
setbg 11
twirl 9 35 15
end
```

Perhaps members of a class would like to indicate who has programmed the procedure. Hence, introduce some print statements within the procedure.

```
to five
print [Programmed by]
print [Joel Rodgers]
setc 6
setbg 10
twirl 4 50 20
end
```

```
to six
print [Programmed by]
print [Bobby Ammond]
setc 42
setbg 67
twirl 3 60 14
end
```

```
to seven
setc 7
setbg 14
twirl 2 70 30
end
```

It is possible to impose one twirl procedure on top of another while changing the pencolor. Note the following procedure:

```
to eight
setc 2
setbg 29
twirl 5 35 8
setc 5
twirl 5 40 8
setc 11
twirl 5 45 8
setc 228
twirl 5 50 8
end
```

For an open house in which you might want to produce a moving display of each student's work on a computer screen, the following two procedures will prove helpful:

```
to hold
wait 100
end
```

```
to separate
hold
clearpage
end
```

Now incorporate all of this into a superprocedure. Don't forget the use of recursion (see Salter, 1993). You will want the display to repeat itself for some time. Simply place the name of the procedure just prior to end.

```
to display
clearpage
one
separate
two
separate
three
separate
four
separate
five
separate
six
separate
seven
separate
eight
separate
display
end
```

If you desire to stop such an eternal running procedure, discover how it is done in the **sample** superprocedure later in this article. Note, especially, the subprocedure **curtain**, which contains a counter initialized outside of the procedure.

Some Additions

In the above activities we have used only one figure per screen. It is possible to place a number of figures on each screen. The easiest way of doing this is by using the turtle-move mode to travel from one location to another. Experiment in the Command Center. On the Macintosh we can click on the turtle with the mouse and drag it where we wish. However, be certain that a **st** (show turtle) command is given, because the **twirl** procedure hides it. If the turtle is hidden you will not be able to click with the mouse.

When you have placed a turtle, you need to discover its location. To discover its Cartesian Coordinates use the command **show pos.** The coordinates will appear in brackets. For example: [-127 -47]. If you wish to use the coordinates, enclose them in a command as follows:

```
setpos [-127 -27]
```

Remember *LogoWriter* possesses four turtles. Each figure on the screen could have its own individual pencolor and be drawn with a different turtle. The background color could change frequently. The possibilities of enhancements to the basic program seem to multiply. Here is a sample. However, only one turtle is used. The command, naturally, is **sample**.

```
to sample
clearpage
central.figure
first.sunburst
second.sunburst
third.sunburst
fourth.sunburst
curtain 0 <—An example of initializing an
            input outside of a procedure.
end
```

```
to central.figure
setbg 8
setc 5
twirl 3 35 8
twirl 3 45 8
twirl 3 55 8
end
```

```
to first.sunburst
st
pu
setpos [143 40]
setc 106
pd
twirl 2 40 30
end
```

```
to second.sunburst
st
pu
setpos [143 -54]
pd
setc 11
twirl 2 40 30
end
```

```
to third.sunburst
st
pu
setpos [-144 -58]
pd
setc 10
twirl 2 40 30
end
```

```
to fourth.sunburst
st
pu
setpos [-144 46]
pd
setc 14
twirl 2 40 30
end
```

```
to curtain :counter <—Using a counter to
                      stop a procedure.
if :counter = 4 [stop]
hold
setbg 15
hold
setbg 20
hold
setbg 120
hold
curtain :counter + 1
end
```

Dividing a program up into small parts, as we have done here, makes programming so much easier to follow. This is particularly true for children. A pattern establishes itself in the subprocedures sunbursts. The use of Cartesian Coordinates (paired numbers) is no great problem for a fourth grader. Many third grade math texts introduce the concept using the first quadrant. When children discover how useful the coordinates are, expanding into the second, third, and fourth quadrants is an easy task. It is especially helpful when the computer can give you the precise location. The author worked this program out very quickly in the Command Center. The material there was copied onto the clipboard and pasted on the flip side of the page in *LogoWriter*. It was then a simple task to organize the superprocedure and formalize the procedures thereafter. This is a good way to have students proceed because they can see what they are programming as it occurs.

A User-Friendly Program

The following program was written in *LogoWriter* for the Macintosh. It may be adjusted for the Apple IIe by giving a six-color distribution of 0 through 5. Although the Macintosh has 256 available colors, the writer used the Apple GS distribution of 0 through 15. By merely entering a reply to questions using the keyboard, a somewhat complex figure may be produced with a minimum of effort. Remember that the command **help "colors** will provide the Macintosh user with the full spectrum available to that computer. If you are programming spacing for an Apple IIe, replace the vertical bars (| |) in the type statements with a backslash (\ \). The GS follows the style of the Macintosh.

```
to startup
directions
end
```

```
to directions
direction1
direction2
end
```

```
to direction1
clearpage
type [What background color do you
   wish? (0 through 15) | |]
make "color readnumbercc
setbg :color
type [What pencolor do you wish to
   use? (0 through 15) Make certain
   it differs from background color.
   | |]
make "pentint readnumbercc
setc :pentint
type [How many sides do you want on
   your figure? (3 or 4 or 5, etc.)
   | |]
make "sides readnumbercc
end

to clearpage
if not front? [flip]
rg
ht
ct
cc
end

to readnumbercc
output first readlistcc
end

to direction2
type [What is the length of each
   side in turtle steps? (Below 10
   is rather small, larger than 60
   may wrap it around the screen.) |
   |]
make "length readnumbercc
type [What number of repetitions of
   the figure do you want around one
   point? (1, 2, 3, etc.) | |]
make "repetitions readnumbercc
twirl :sides :length :repetitions
type [Do you wish a printout of this
   figure? (Y or N) | |]
make "answer readchar
if :answer = "Y [printscreen]
type char 13
type [Do you wish to do the program
   again? (Y or N) | |]
make "answer readchar
if :answer = "Y [directions]
wait 100
clearpage
end
```

```
to polygon :sides :length
st
repeat :sides [forward :length right
   (360 / :sides)]
ht
end

to twirl :sides :length :repetitions
repeat :repetitions [(polygon :sides
   :length) right (360 /
   :repetitions)]
end
```

References

Yoder, Sharon (1990). *Introduction to programming in Logo using* LogoWriter. Revised Edition. Eugene, Oregon: ISTE.

Salter, Mary (1993). *Exploring recursion with* LogoWriter. Eugene, Oregon: ISTE.

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

WORKSHEET

Name ______________________ Date _______________

Keep a record of your more important discoveries on the worksheet. Start with polygon then go on to **twirl**. With **twirl** keep a record of successful background and pencolor combinations. Imposing one twirl procedure on another may also lead to some surprising results.

	:sides	:length
polygon	______________	________________
polygon	______________	________________
polygon	______________	________________
polygon	______________	________________
polygon	______________	________________

setbg #	setc #	Twirl	:sides	:length	:repetitions
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________
________	________	Twirl	________	________	________

You Gotta Have a Gimmick

by Jody Krupski

Although most people would agree that using grammar accurately is admirable and even necessary, many people cringe at the thought of studying the specifics of grammar. In the words of Gypsy Rose Lee, "You gotta have a gimmick." Using *LogoWriter* text commands is a way to reinforce sentence structure and parts of speech in an enjoyable way.

After a student has been introduced to the concept of subject/verb agreement, the program described in this article could be used at any level of foreign language study. The level of difficulty can be increased as the teacher increases the number of specific grammar elements required in sentences.

The beginning stages of this activity are done away from the computer. Students working individually or in small groups are asked to create a list of subjects in the third person singular. Each subject must be a living being. This could include a possessive adjective, definite and indefinite article, or even a proper name—specific to the target language, of course. The student is then asked to create an equal number of action verbs conjugated in the third person singular.

Here are some examples in French.

Subjects:	Ma soeur, Monique, Jean-Claude
Verbs:	montre, indique, donne

At this point the program listed at the end of this article can be used. However, to increase the level of difficulty and the possibility for amusing, nonsense sentences, the teacher can instruct the student to make lists of direct objects, indirect objects, prepositional phrases, adverbs, etc. Here are some examples:

Direct objects: l'os, la photo, le sac
Indirect objects: 'a ekkem 'a Luc, aux amis

Once the necessary elements are ready, each group enters their vocabulary, following the teacher's instructions. Most students derive great pleasure from seeing their own words on a computer screen. On a more practical level, it forces the student to think about how he/she is classifying each word. It also gives the added impact of stressing the importance of spelling. If this is done with a group, all members should be responsible for typing an equal number of words or phrases.

Logo procedures that use the speech parts have the following structure:

```
TO SUBJECT
OUTPUT PICK [[ ] [ ]]
END

TO VERB
OUTPUT PICK [[ ] [ ]]
END
```

and so forth.

Students are asked to enter their words inside the inner brackets ([ ]) and to add as many lists as they wish. They must remember that their list of lists begins and ends with a double bracket ([[or]]).

The students should type the words using all lower case letters except for the first letter of the subject. On most keyboards it is also possible to add accent marks when using lower case letters. On an IBM computer, hold the Alt key while using the numeric key pad to type the appropriate ASCII code number. For example, to get an **à** the student would type Alt + 133. The most common accented letters in French are:

é è

The type of vocabulary the teacher expects to be used depends on the purpose of the exercise. The teacher could assign each student a chapter for which the vocabulary should be stressed. The students could be given identical sets of prepositions and be told to use them all. When direct and indirect objects are being reviewed, the students could be asked to make a list of verbs that use direct and/or indirect objects.

Completed procedures in French might look like this:

```
TO SUBJECT:
OUTPUT PICK [[ Le president] [Mon
   frere aine] [Le prof] [Ma soeur]
   [Olivier] [Le chien] [La
   Parisienne] [Yves] [La serveuse]]
END

TO VERB
OUTPUT PICK [[donne] [vend] [montre]
   [explique] [indique][enseigne]
   [dit] [lit] [ecrit]]
END
```

```
TO DIRECT
OUTPUT PICK [[la lecon] [le fromage]
   [le journal] [ la verite] [l’os]
   [le menu] [le Metro] [le billet
   doux] [les actions] [la tour
   Eiffel]]
END
```

```
TO INDIRECT
OUTPUT PICK [[aux eleves.] [a la
   souris.] [a l’infirmiere.] [aux
   amis.] [au grand chien.] [au
   client.] [au touriste.] [a
   l’amant.] [au businessman.] a
   l’excursioniste.]]
END
```

The Program

The student's input is then used by the sentence-generating procedures that have been prepared in advance of the student activity. The first procedure, FRENCH, directs the order of output for the parts of speech. The second procedure, the PICK procedure, allows for random word choices from each student's list of words for a particular part of speech.

```
TO FRENCH
OUTPUT (SENTENCE SUBJECT VERB DIRECT
   INDIRECT)
END
```

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

Directions

Once the students have contributed their input, they can run the program independently The procedure, COMMENCE, includes instructions for using the program.

```
TO COMMENCE
BONJOUR
INSTRUCT
GET.GOING
BEGIN.AGAIN
END
```

```
TO BONJOUR
HT
CT
PRINT [COMMENT VOUS APPELLEZ -
   VOUS?]
PRINT []
(PRINT [ENCHANTEZ DE FAIRE VOTRE
   CONNAISSANCE,] READLISTCC)
END
```

```
TO INSTRUCT
WAIT 30
CT
PRINT [YOU ARE GOING TO CREATE A
   GRAMMATICALLY ACCURATE FRENCH
   SENTENCE. YOUR GOAL IS TO THEN
   TRANSLATE THAT SENTENCE INTO GOOD
   ENGLISH.]
WAIT 125
END
```

```
TO GET.GOING
PRINT []
PRINT [TO BEGIN, TYPE "PRINT FRENCH"
   (AT THE END OF THE
   INSTRUCTIONS.)]
WAIT 75
PRINT []
PRINT [AFTER THE SENTENCE APPEARS,
   TYPE "INSERT [TRANSLATION]".]
WAIT 100
PRINT [YOU CAN CONTINUE THIS PROCESS
   AS LONG AS YOU WANT]
WAIT 50
PRINT []
PRINT [WHEN YOU ARE FINISHED TYPE
   "FINI"]
WAIT 75
END
```

```
TO BEGIN.AGAIN
PRINT []
PRINT [ANYTIME YOU CANNOT REMEMBER
   DIRECTIONS, TYPE "COMMENCE".]
WAIT 75
PRINT []
PRINT [REMEMBER, TO BEGIN TYPE
   "PRINT FRENCH".]
WAIT 50
PRINT []
PRINT[AMUSEZ - VOUS BIEN!]
WAIT 30
CT
END
```

Having the students type the translations in upper case helps the teacher and the student to follow his/her progress more easily.

As a means of checking students' progress and to give them something concrete to hold on to, the students can print a copy of their sentences. The instructions are included in the "Fini" procedure.

```
TO FINI
PRINT [NOW THAT YOU HAVE FINISHED
   CREATING AND TRANSLATING
   SENTENCES IT IS TIME TO MAKE A
   COPY FOR YOUR TEACHER.]
PRINT []
WAIT 30
PRINT [TO PRINT YOUR SENTENCES TYPE
   "PRINTTEXT".]
PRINT []
WAIT 50
PRINT [MAKE SURE THE PRINTER IS ON!]
PRINT []
WAIT 30
PRINT [HAND IN YOUR COPY TO THE
   TEACHER.]
PRINT []
PRINT [A BIENTOT!]
WAIT 75
END
```

Sample Output

Here is an example of output using the words and procedures given earlier in this article:

```
Le president ecrit l'os a
l'excursioniste. THE PRESIDENT WRITE
THE BONE TO THE SIGHTSEER. La
Parisienne ecrit le journal aux
eleves. THE PARISIAN WRITES THE
NEWSPAPER TO THE STUDENTS. Le
president indique le fromage aus
eleves. THE PRESIDENT SHOWS THE
CHEESE TO THE STUDENTS. Le prof
ecrit le billet doux au client. THE
TEACHER WRITES THE LOVE NOTE TO THE
CUSTOMER. Mon frere aine ecrit le
journal a l'excursioniste. MY OLDEST
BROTHER WRITES THE NEWSPAPER TO THE
SIGHTSEER. Yves explique la tour
Eiffel au businessman. YVES EXPLAINS
THE EIFFEL TOWER TO THE BUSINESSMAN.
Le chien donne le Metro au touriste.
THE DOG GIVES THE METRO TO THE
TOURIST. Ma soeur indique la verite
au grand chien. MY SISTER SHOWS THE
TRUTH TO THE BIG DOG.
```

Project Modifications

- The foreign language teacher who teaches more than one language can easily adapt this program to another language.
- Once a student, or group of students, has completed translating their own sentences they can attempt to translate those of another group.
- The input of each group can be saved and then reused at the end of the semester or quarter to review for an exam.
- This program can work well in either a cooperative learning environment or individually.
- The teacher can ask the student to change verb tenses.
- The program can be useful in reviewing subjunctive sentences at the upper level. The students can be required to use only subjunctive moods in their introductory clause.

The usefulness of this program is limited only by the ideas generated by teachers and their students. Through the procedure described, Logo provides drill and practice "gimmick," a useful tool to challenge and inspire students when reviewing grammar. Students laugh at silly sentences they create while being confined to uses of accurate grammar. Logo assists in proving that grammar really is fun!

A Research-based Logo in the Classroom

by Douglas H. Clements and Julie Sarama

Logo: Search and Research

Way back in 1993 (*Logo Exchange, 12*(2),1993-94) we discussed "A research-based Logo for mathematics." We also promised to tell you what happened when we worked with teachers to use this new version in the classroom. We're fairly sure you weren't holding your breath, so we'll skip past the apologies and start right into it.

A Quick Reminder: *Geo-Logo*™ and *Turtle Math*™

Our version of Logo has two names. It's called *Geo-Logo* when used in the new elementary curriculum, *Investigations in Number, Data, and Space.* A more complete version is available as *Turtle Math,* from LCSI (we'll simplify things and use the latter name here). *Turtle Math* is a version of the Logo computer language tailored specifically for investigations in geometry and other mathematical topics. What is different about *Turtle Math*?

1. Turtle Math *provides measurement tools.* For example, an on-screen protractor, placed at the turtle's position and heading, measures turns. One arrowhead shows the turtle's heading. The other follows the cursor, which students move with the mouse. When they click this arrowhead it "freezes" and a turn command is displayed. Rulers and other measurement tools are also available.
2. Turtle Math *helps students close ties between representations.* In *Turtle Math,* students enter commands in "immediate mode" in a Command Center. Any change to commands in either location, once accepted, are reflected automatically in the drawing. The dynamic link between the commands in the command center and the geometry of the figure is critical.
3. Turtle Math *allows easy editing.* A Step tool allows students to walk through commands to find errors or explore mathematical properties. Other tools allow easy editing and erasing.
4. Turtle Math *facilitates children's learning of mathematical ideas.* In several other ways, *Turtle Math* encourages students to build solid ideas about mathematics. For example, the Turn Rays option shows rays during turns. If you type rt 120, a ray is drawn to show the turtle's initial heading. Then as the turtle turns, another ray turns with it, showing the change in heading throughout the turn. A ray also marks every 30° of turn.

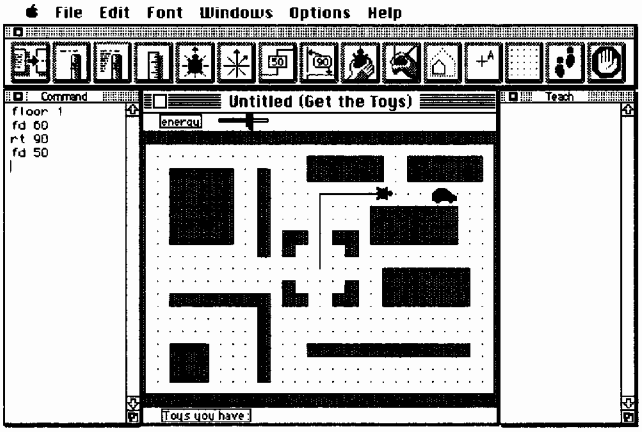
The Curriculum Unit

Turtle Paths (Clements et al., 1995) engages students in a series of combined geometric and arithmetic investigations. The unit teaches about geometric figures such as paths and triangles; geometric processes such as measuring, turning, and visualizing; and arithmetic computation and estimation. Throughout the unit, students explore paths and the lengths of paths.

Investigation 1—Paths and Lengths of Paths

1. Students walk, describe, discuss, and create paths. They give Logo commands to student-robots to specify movements that create paths, which starts the development of a formal symbolization that is built up during the remainder of the unit
2. In an off-computer game, "Maze Steps," students count steps in a maze to find certain paths, such as a path that is 14 steps in length and has 2 corners, and play a similar Maze Paths game. Problems with more than one possible solution are emphasized.

3-4. Students give movement commands to the Logo turtle to create paths in a "Get the Toys" game. Only 90° turns are used. The computer activity is designed to promote thoughtful use of the commands (as opposed, for example, to nonreflective trial and error). A on-screen battery's limited energy decreased with each command used.



```
floor 1
fd 60
rt 90
fd 50
```

Investigation 2—Turns in Paths

5. Students turn their bodies, discuss ways to measure turns, and learn about degrees. They are introduced to another computer game, "Feed the Turtle," in which they must use their knowledge of turns to direct the turtle through channels of water (turns in this game are multiples of 30°).
6. Students play the "Feed the Turtle" game and thus continue to estimate and measure turns. They discuss the nature of triangles and build their own descriptions of that class of geometric figures.
7. Students write Logo procedures to draw equilateral triangles.

8-9. Students find the missing measures (lengths and turns) to complete partially-drawn paths. The students must figure out that the turtle must move fd 10 (forward 10 turtle steps) to finish this side, and so on. In this and more complex problems, they must analyze geometric situations and use mental computation in a meaningful setting.

Investigation 3—Paths With the Same Length: Isometric Exercises

10-11. Students write as many procedures as possible to draw different rectangles of a certain overall length or perimeter—200 steps. As with the missing lengths and turns problems, they must analyze geometric situations and use mental computation.

12-14. Students design and program the computer to draw a picture of a face, for which each part (e.g., ear, mouth) has a predetermined length or perimeter; the shape (rectangle, square, or equilateral triangle) is the students' choice.

In the Classroom

Paths, Perimeter, and Length

Combining commands. Given the focus of this unit, we looked at what students learned about paths, perimeter, and length using *Turtle Math.* One striking result was how much students combined and decomposed measures of length, one of our educational goals. The first computer activity, the "Get the Toys" game, achieved its purpose of motivating combinations. The falling energy of the battery was especially salient to the students and motivated them to combine old commands to create new, more efficient ones. Early in their play, Anne told her pilot test partner, "You should go back 50. Because 30 + 10 + 10 is 50. It's less commands to type one bk 50 than three backwards, 30, 10, and 10."

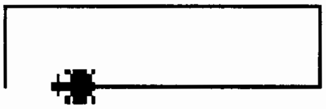
Growth was evident for most students; during their second round of the game, these girls began to combine commands before the second command was entered (e.g., "We need 20 more; 80 plus 20; change fd 80 to fd 100").

Oscar and Peter combined commands in a different manner. They appeared to need to use successive approximation, with a series of forward and back movements, to get the turtle located correctly. Only then would they edit the commands to produce a more efficient solution. For example, they once entered fd 80 and when the turtle hit a wall, they started over, entering fd 50 fd 10 fd 10. To save energy, they combined these three commands to fd 70. On other occasions as well, they needed to see all the commands enacted before they grasped what the situation required.

Monica and Nina became adept at combining commands fairly quickly; even away from the computer they could answer questions such as "If you had gone forward 40 and back 10, how would you combine the two commands?" correctly, justifying that in this case, one "needed to minus because the turtle is coming back, it is not going on." They were also competent at decomposition and reversals in these situations. In "Feed the Turtle," they had gone forward 60, at which point the turtle had eaten berries. They saw they had farther to go. Monica suggested that they change the command to forward 70. They tried it and found that the turtle went over the berries without eating them. So they erased the fd 70 and entered fd 60 and, afterwards, fd 10. So, Monica and Nina were at a level of reasoning above Oscar and Peter because they could think about the change as part of the new command before they saw it. They can think in terms of part-wholes, maintaining a 70-step length as a whole and 60- and 10-step lengths as parts of this whole. Oscar and Peter could not. They combined numbers after the fact, not lengths.

Combining lengths. What many students did to determine lengths should, perhaps, not have been surprising. The teacher presented the first problem.

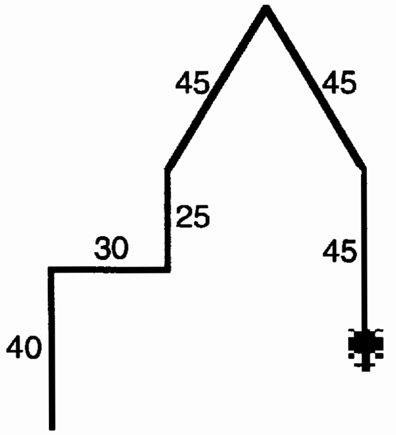
The inventor gave you only this much.



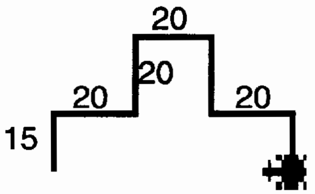
How could you finish the procedure?

```
to rectangle1
fd 20
rtf
fd 80
rtf
fd 20
rtf
fd 60
...
end
```

Anne protested, "But there's no dots! [Such as those on the "Get the Toys" game.] How do we know how far to make the turtle go?" They figured out that another forward command was needed. Charles indicated that its input should be 20, "Because the space looks about the same as that side that is 20." David said, "Because 20 plus 60 equals 80." They got better at combining lengths as they worked at the computer. For example, explaining his solution to the "building" problem, Charles said, "I thought the missing part on the right side was 10, because it was such a little space. But now I figured it out: 40, 25, so...65. So, it has to be a 20, not a 10. Because, 20 plus the 45 would be 65."



All students came to invent good strategies. For many, *Turtle Math*'s feedback alone prompted them to review their choices. Nina, for example, typed fd 50 for the final side of the figure below, but after seeing that the figure did not close, calculated the missing length by looking at the labeled measures on the opposite sides.



Once they had a strategy, all of the students finished the activity successfully. Monica and Nina stated enthusiastically that the activity was fun; they seemed to enjoy the mental arithmetic and its power to determine "just the right move."

Connections between number and geometry. After gaining experience playing the "Get the Toys" game, Anne said, "Try bk 50. Because we put fd 40 in [pointing at the Logo code], but then we had to put in a fd 10 [pointing first to the code, but immediately then to that section of the path], so it's got to be bk 50 to get back [gesturing back along the section of length 50]." Anne had integrated the geometric form of the turtle's path and the symbolic-numeric Logo commands that created it. How common was such integration?

Our observations indicate that the answer to this question differs by task and student. Charles, for example, gave signs of simultaneously applying numerical and geometric schemes (e.g., spatial images). On a missing length problem, he had entered fd 40 for the missing bottom side, but had come up short. He mentioned that he forgot the "20 part" at the top.

Teacher: Do you have to go back and do the fd 20 before the fd 40?

Charles: No! I can just add the 20 now. It doesn't matter. I should have put 60. My command was wrong. 'Cause the 40 only takes care of this part on top.

For the 200-step rectangle task, Charles told about his plan: "First I wanted 10 and 10, then I couldn't get it, so I made it 20 20 80 80."

Teacher: What if you had wanted it 10 and 10?

Charles: Well, then that would have to be...90 and 90.

Teacher: How did you do that?

Charles: I just added the 10 on to the two long sides...because you can't leave it out.

Charles implied that the total perimeter must be maintained; the difference of 10 between the old side length (20) and the one proposed by the teacher (10) must be maintained, and the only way to do this was to add the length to the other sides. Charles deliberately added one measurement to the length of two sides, as opposed to operating (80 + 10) with no referents. So, Charles was connecting the numbers and the lengths.

Not all students made such connections immediately. Barb and Anne initially tried to "get the numbers to add up" to 200, drawing rectangles but without establishing a connection between the numbers and the figures they drew. For example, they drew a rectangle and labeled the sides 70 30 60 40.

Role of Turtle Math. There are five characteristics of the *Turtle Math* computer environment that aided construction of units for these students.

1. *Changes problem situation.* The small steps (and thus larger numbers) on the computer, compared to the students' actual steps or the steps in the "Maze Paths" game, encouraged students to work on larger units, such as tens (it is difficult to even imagine a single turtle step).
2. *Provides feedback that students can use to reflect on their thinking.* As one example, recall the rectangle with sides labeled 70 30 60 40. When Barb and Anne tried this solution on the computer, however, the resulting open path immediately led them to connect the side lengths to the attributes of a rectangle.

Anne: It's not a rectangle!

Barb: Because it's like this [draws in the air]. Because one side was not as long as the other one. So, we have to do that again.

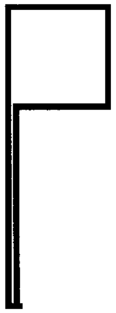
3. *Motivates students.* Oscar and Peter were, in the teacher's opinion, the "lowest ability" students in mathematics. On paper and pencil, they often solved the problems with visually based guessing, but on the computer, they were convinced that only the deliberate use of arithmetic was adequate. They needed pencil and paper for almost all the calculations. (As an afterthought in the "Get the Toys" game, for example, they did not use such strategies in missing measures problems.) They were, however, highly motivated by this computer activity, and their teacher had to stop them from fighting between themselves as to who would get to do the arithmetic for each figure (to her, a happy incident that she reported to us eagerly).
4. *Helps make dynamic connections.* Anne and Barb never combined their commands on the "Get the Toys" game when they wrote Logo commands by hand, but did combine them when they typed the commands into the computer. Later, they began to combine commands on paper as well.
5. *Integrates the geometric and numeric.* When they successfully integrated geometry and number ideas, most students talked about *movement*. For example, Nina and Monica used arithmetic to solve problems, but their language was about the turtle: "It's going on 50, then 30 more, so that's...." The emphasis here is not on the geometric figure, as much as it is on the turtle's movements. Thus, the emphasis on physical action and the dynamic connections between the symbolic and graphic representations in *Turtle Math* facilitated students' development of such connections for themselves.

Shapes

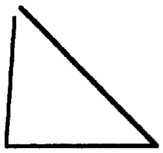
The school teachers originally questioned the time spent on the concept of triangle. They believed that their students had that idea "down" for years. During the unit, they expressed disbelief that their students had such limited and rigid ideas about triangles. The dominant belief in both classes was that only isosceles or only equilateral triangles were "triangles." A few children contended that the base of any triangle had to be horizontal. Further, they knew little or nothing about the measures of triangles.

We discovered that certain attributes of shapes seemed especially conspicuous for the students. So, teachers can use these to help children become more aware of shape classifications.

One such conspicuous property was the notion of path, upon which the unit was built. Students used this notion to understand such basic concepts as a "closed figure." For example, to create a path that is 40 steps long, closed, and simple, Anthony drew a path that almost doubled back on itself, but he was careful to keep the path from tracing over itself.



Students used this concept to reject nontriangles such as



which often confuse students.

Other properties were less conspicuous. For example, a teacher pointed out all the equilateral triangles and asked what distinguished them. Stephanie thought they were the same because they were "all not pointed up." Ryan thought they were "all fat."

Teacher: The same amount of fat?

Beth: Yes, the same amount.

The teach gave prompts; for example, asking which side was longest.

Stephanie: All three of these are the same amount long.

Beth: I mean that each triangle in the 3 are the same (as she traced the sides with her hands).

It was not until they had programmed these shapes and used the Label Lengths tool to "prove" they were equilateral triangles that the children understood "equilateral." These experiences extended, rather than limited, their ideas about triangles. Gina was asked if the triangles she was drawing in Logo were the only kind of triangles.

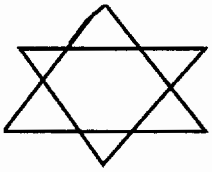
Gina: Triangles don't have to be equilateral triangles. But if you don't have an equilateral triangle, you have to turn ... not 120.

Teacher: How much would you turn?

Gina: Sometimes more, sometimes less.

Thus, Gina not only accepted several subcategories of triangles, she spontaneously linked this to the measure of the turns necessary to draw them with the turtle.

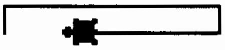
Turtle Math experience also helped children understand that certain attributes were not defining properties of geometric figures, though this conflicted with their previously-developed concept images. One example is a figure's orientation. Susan wanted to draw a star.



Gina asked her how she possibly could draw something like that. Susan explained, "First, I would draw my procedure 'tri' and that would be this and then you have to turn the turtle and draw one 'tri' that is upside down." This episode also speaks to Susan's developing ability to compose and decompose shapes.

Using the turtle also helped students link geometry and arithmetic. One teacher, Sue, wrote turtle commands and drew the figure they made on the board.

```
fd 10 rt 90
fd 80 rt 90
fd 10 rt 90
fd 50
```

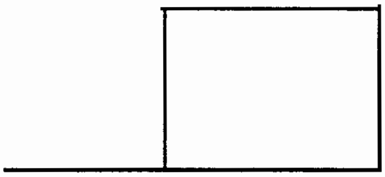


Sue explained that this procedure was not finished and was supposed to make a closed figure. The class was quite sure that the "bottom line must also be 80 turtle steps." Several immediately said that the turtle needed to go 30 steps forward. Susan's explanation was, "50 plus 30 is 80. 80 and 80 would be equal. 10 and 10 are equal. And it is a rectangle."

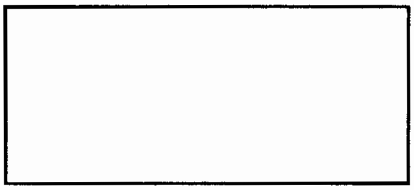
Most observed children completed these tasks successfully. Enough so, in fact, that the teacher presented more difficult challenges, such as writing the commands fd 80 rt 90 fd 20 rt 90 fd 20 fd 60 rt 90 fd 10 fd 10 rt 90 and asking if they would generate a rectangle. Initially, some students, such as Susan, were confused and were not sure "because it has to have four forward commands." However, as she worked it through with the class she seemed to agree that it would draw a rectangle.

When Sue presented a new problem—fd 40 rt 90 fd 65 rt 90 fd 20 fd 20 rt 90 fd 50 fd 15 rt 90—both Susan and Gina agreed that it would be a rectangle, until they tried drawing it (Susan remarked, "You can't really say how it would look like without drawing.").

Susan



Gina



Gina explained that 20 and 20 added up to 40, which is equal to the other 40 and that Sue was only tricking them. Susan quickly realized that 50 and 15 is actually 65.

Thus, after solving several of these problems, most students were competent at this task. Susan and Gina went off to design some of their own "puzzlers" to give to each other.

Final Words

We believe that *Turtle Math* encourages and supports the use of mathematical ideas and processes.

Students' integration of numbers and geometry was especially potent in the *Turtle Math* environment, which provided meaningful tasks. The geometric setting provided both motivations and models for thinking about number and arithmetic together. The motivations included game settings and the desire to create shapes and designs. The models included length and turns as settings for building a strong sense of both numbers and operations on numbers, with measuring and labeling tools supporting such construction. Conversely, the numerical aspects of the measures provided a context in which students had to attend to certain properties of geometric forms. The measure made such properties (e.g., opposite sides equal) more concrete and meaningful to the students. In addition, the change in problem situation encouraged the use of larger number units. The dynamic links between these two domains structured in the *Turtle Math* environment (e.g., a change in code automatically reflected in a corresponding change in the geometric figure) facilitated students' construction of connections between their own number and spatial schemes. Finally, *Turtle Math* provided feedback that students used to reflect on their own thinking.

Reference

Clements, D. H., Battista, M. T., Akers, J., Woolley, V., Meredith, J. S., & McMillen, S. (1995). *Turtle Paths.* Cambridge, MA: Dale Seymour Publications.

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 Sarama, he is co-authoring new versions 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 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

MathWorlds

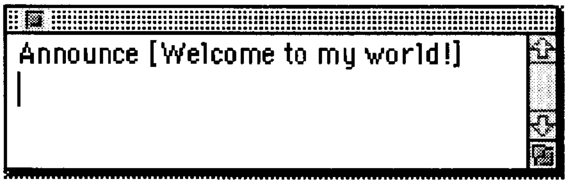
A Dialog with a Computer

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

The very first Logo developed in the 1970s had no turtle. It was implemented using a Teletype as an output device. The addition of graphic displays and screen turtles added an entire new dimension that colors public perception of the programming language. To this day turtles and Logo are forever inseparable in the public mind. Initially text and turtle graphics could be intermixed on the same portion of the screen only in a cumbersome way, if at all. Newer versions of Logo such as *LogoWriter* and *Logo Plus* addressed these limitations, allowing turtles and text to be together on the screen.

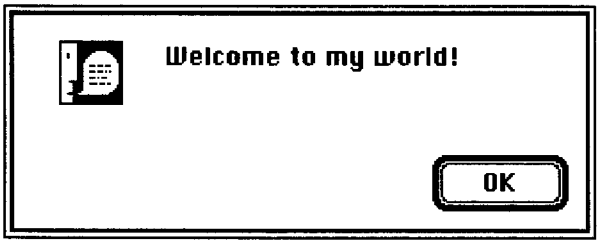
MicroWorlds is one of the first versions of Logo designed for a graphical interface with windows. Developed initially for the Macintosh and now available for DOS systems as well, *MicroWorlds* extends the text commands previously developed for *LogoWriter*. ANNOUNCE, QUESTION, and ANSWER are new Logo commands created for *MicroWorlds* that permit use of the dialog boxes now commonly found in most graphic user interfaces such as the Macintosh and *Microsoft Windows*.

A dialog box presents information to the user. The ANNOUNCE command is the *MicroWorlds* extension to Logo that permits information to be presented through a dialog box. For example, entering the following command in the *MicroWorlds* Command Center:



```
Announce [Welcome to my world!]
```

produces the following dialog box in *MicroWorlds*:



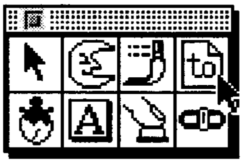
Welcome to my world!

The ANNOUNCE command generally would not be typed directly in the *MicroWorlds* Command Center. More often a procedure will be created that might be accessed through a button, for example. This consists of a two-step process:

1. Create a procedure with the name that you wish to appear in the button.
2. Enter the name of procedure as an instruction in the button's dialog box.

Creating an Announce Procedure

To create an ANNOUNCE procedure, click once on the icon that resembles a sheet of paper with the word "to" on it to access the Procedures page:



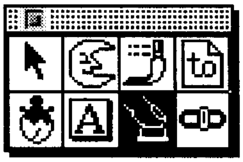
The name of the procedure should be the name that you want to appear on the button. In this instance, the editor will be used to create a **Welcome** procedure. (All Logo procedures begin with the word "TO" and end with the word "END".)

```
TO Welcome
ANNOUNCE [Welcome to my World!]
END
```

Click the editor icon on the Tools palette a second time to return to the previous *MicroWorlds* page when you have finished entering the procedure.

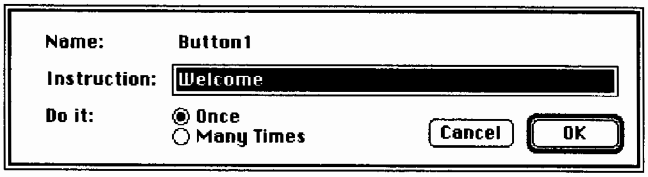
MicroWorlds Buttons

MicroWorlds provides the facility for creating a button that will run the WELCOME procedure when clicked. The button tool is just to the left of the slider tool on the *MicroWorlds* Tools palette:



After the button tool is selected, a hand will appear:

Use the hand to point to the location on the *MicroWorlds* page where you want the button to appear, and click the mouse button once. A dialog box will appear. Enter the name of the procedure—in this instance, "Welcome"—that you wish the button to run, and then click "OK". (If you need to change the button's name or instruction at a later time, hold down the shift key and double-click the button to recall its dialog box.)

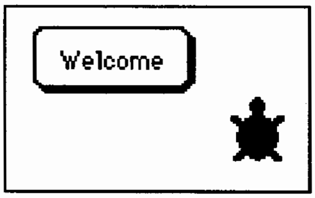


Name: Button1

Instruction: Welcome

Do it: ● Once ○ Many Times

After the button has been created, clicking it will run the procedure and produce the "Welcome" dialog box.



Dialog Boxes that Ask Questions

The ANNOUNCE command only produces a dialog box that makes an announcement. The QUESTION command can be used to create a dialog box that asks a question. When the question is answered, the result can be accessed through the ANSWER reporter. Logo's SENTENCE command can then be used to construct a sentence that contains the answer as part of a response from the computer.

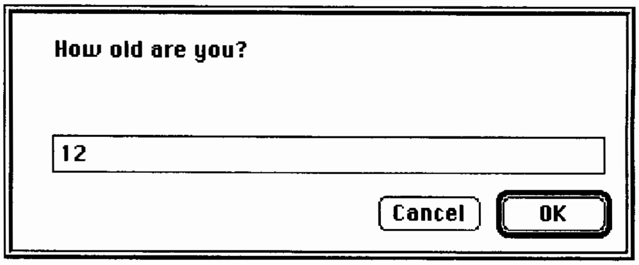
To explore how this works, enter the following procedure in the procedures editor:

```
TO Age?
QUESTION [How old are you?]
ANNOUNCE (SENTENCE [I think] ANSWER
   [is a very good age.])
END
```

Troubleshooting Tip: Be careful to distinguish between brackets "[ ]" (which are square) and the parentheses "( )" (which are curved) in the procedure or the procedure will not work. (A more detailed explanation of conditions that require parentheses is provided in a subsequent section titled, "When Parentheses are Required.")

After the AGE? procedure has been entered, create a button with this name:

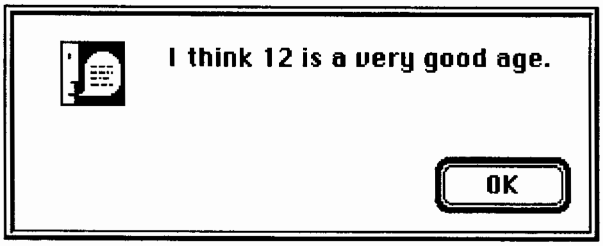
Clicking the button will produce the following dialog box:



How old are you?

12

In this instance, the answer "12" has been entered in response to the question. The procedure will respond with the reply that 12 is a very good age:



I think 12 is a very good age.

In this case, the QUESTION, ANSWER, and ANNOUNCE commands have been used to construct a dialog between the user and the computer. Logo is a dialect of a list- processing language known as "LISP," which has often been used for construction of artificial intelligence programs designed to conduct man-machine dialogs, so it is unsurprising that Logo also has good facilities for creation of such dialogs.

Evaluating Answers

Although such programs may create the illusion of conversing with another entity under the proper conditions, a nonsensical answer can quickly reveal the limitations of the program. For example, if the user answers "rutabaga" in response to the question, "How old are you?," the computer will cheerfully answer:

```
"I think rutabaga is a very good age."
```

In some instances it is possible to check for answers that do not make sense and provide an appropriate response. To illustrate how this might be done, a program will be described below that asks for the user's favorite color, and sets the background of the *MicroWorlds* screen to that color.

The SETBG (Set Background) command sets the background color of the *MicroWorlds* screen. Either the name of a color or a number can be used as inputs to the SETBG command. For example, the following command:

```
SETBG "RED
```

will set the background of the *MicroWorlds* screen to the color red. (*Troubleshooting Tip:* Note that a quotation mark must be placed before the name of the color specified.)

A range of numbers can be used to specify more precise shades; for example, the numbers 10 through 18 produce backgrounds that range from pink to deep red. (To determine the number of any color, first open the Drawing Center by clicking on the paintbrush in the *MicroWorlds* toolbox. Then choose the Question Mark under the Help Menu. Place the Question Mark on a color in the Drawing Center and click to display the number associated with any color.) Typing SETBG 15 will change the background to a bright red in the current version of *MicroWorlds*:

```
SETBG 15
```

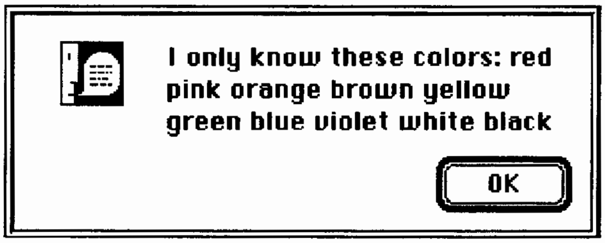
Two procedures below, COLOR? and VALIDCOLORS, ask for a favorite color and set the *MicroWorlds* background to that color. It is important to check for valid colors; otherwise the program might respond to answers such as "rutabaga" with error messages like "SETBG does not like rutabaga as an input."

The COLOR? procedure uses the QUESTION command to ask for a favorite color. The second line of the COLOR? procedure uses the MEMBER? reporter to check the ANSWER to see whether it is a member of a list of valid colors. (To determine this, the COLOR? procedure calls the VALIDCOLORS procedure, which responds by outputting a list of valid colors known to Logo.) If the answer is a valid color, the procedure sets the background to that color and stops.

```
TO Color?
QUESTION [What is your favorite
   color?]
IF MEMBER? ANSWER ValidColors [SETBG
   ANSWER STOPME]
ANNOUNCE SENTENCE [I only know these
   colors:] ValidColors
END

TO ValidColors
OUTPUT [red pink orange brown yellow
   green blue violet white black]
END
```

If the answer is not a valid color known to the "Set Background" command, the third line of the COLOR? procedure announces that it only knows certain colors.



I only know these colors: red pink orange brown yellow green blue violet white black

The COLOR? procedure illustrates how a dialog with a computer might be constructed and extended. The MEMBER? command is particularly useful in determining whether an answer is a member of a valid set of responses in such dialogs.

When Parentheses Are Required

There is one other noteworthy aspect of the COLOR? procedure: the SENTENCE command does not have the curved parentheses before and after it that were required in the AGE? procedure. That is because SENTENCE normally expects two inputs. In the AGE? procedure, three inputs are supplied to SENTENCE:

1. [I think]
2. ANSWER
3. [is a very good age.]

When more than two inputs are supplied to the SENTENCE command, curved parentheses are used to mark the beginning and end of the sentence. This allows the Logo interpreter to parse the command correctly:

```
ANNOUNCE (SENTENCE [I think] ANSWER [is
   a very good age.])
```

Because the SENTENCE command in the COLOR? procedure only has two inputs:

1. [I only know these colors:]
2. ValidColors

parentheses are not required:

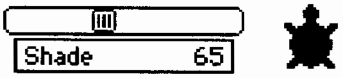
```
ANNOUNCE SENTENCE [I only know these
   colors:] ValidColors
```

Programs that Learn

The hallmark of an artificial intelligence program is an ability to learn. In this instance, an obvious weakness of the COLOR? program is that Logo may not know the names of all the colors that an average user might want to employ. For example, the version of Logo supplied with the current version of *MicroWorlds* knows the color VIOLET but does not know the term PURPLE.

The COLOR? program could be enhanced by allowing it to learn new names. Classic programs written in various dialects of LISP and Logo exhibit this characteristic. For example, the classic ANIMALS program (which has been constructed in both LISP and Logo) has the ability to associate animals with their characteristics, and can learn new animals taught to it by users.

The COLOR? program could be redesigned to ask the user to indicate the correct shade when the name of a color the program does not know is encountered. There are several ways this strategy might be implemented, depending upon the dialect of Logo employed. For example, the COLORUNDER command in *MicroWorlds* can be used to determine the current color under the turtle. The user might be asked to move the turtle to the square with the correct color in a color grid. Another input method available in *MicroWorlds* consists of a slider control that could be employed to allow a user to select a color by moving a slider to change the color of the turtle.



Shade 65

Regardless of the input method, once the program learned the color that a user associated with a particular name, it could then add this color to the list of known colors for future use. Typically a Logo list maintained by the program would be used to allow it to remember new colors.

Logo Adventures

MicroWorlds combines Logo with the card-and-stack metaphor originated by Bill Atkinson for *HyperCard*. (*MicroWorlds* employs the metaphor of pages in a project in place of the *HyperCard* analogy of cards in a stack.) Each page in a *MicroWorlds* project can consist of a different location within a virtual world.

Dialog boxes can be used to conduct conversations with adventurers as they tour these mini-universes and "micro-worlds." The correct answer to a riddle, for example, might convey adventurers to a locked room in a castle. These activities have been possible with the earliest versions of Logo, but the newest capabilities of *MicroWorlds* facilitate construction of such worlds.

Summary

Even before the turtle was added to Logo, the language contained facilities for working with lists of words and sentences and processing them to construct conversations and dialogs. The addition of dialog boxes to dialects of Logo found in programs such as *MicroWorlds* updates this capability to provide an interface similar to those found in modern graphical user interfaces.

If you would like to know more about creating Logo programs that can learn, we recommend a series of three books by Brian Harvey titled *Computer Science: Logo Style*. If you would like to learn more about Logo and natural language, we recommend *Exploring Language with Logo* by Paul Goldenberg and Wallace Feurzeig.

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

ISTE Books & Courseware Order Form

To order ISTE products advertised in this publication, find the product title in the following list and enter it on the form below.
To receive a free catalog with a complete listing of ISTE products and services, please call our toll-free number, 800/336-5191.

Product (• Indicates an ISTE-published title)	*Member Price*	*Nonmember Price*
• *Introduction to MicroWorlds—A Logo-Based Hypermedia Environment*	22.45	24.95
• *Way of the Ferret—Finding Educational Resources on the Internet*	22.45	24.95
The Whole Internet—User's Guide & Catalog	24.95	24.95
Education on the Internet—A Hands-on Book of Ideas, Resources, Projects, and Advice	25.00	25.00
Hands -On Internet—A Beginning Guide for PC Users	29.95	29.95
How the Internet Works	24.95	24.95
How to Use the Internet	17.95	17.95
NetPower—Resource Guide to Online Computer Services	39.95	39.95
Realizing the Information Future—The Internet and Beyond	24.95	24.95
Internet Companion—A Beginner's Guide to Global Networking	12.95	12.95
The Mac Internet Tour Guide—Cruising the Internet the Easy Way	27.95	27.95
Windows Internet Tour Guide—Cruising the Internet the Easy Way, 2nd Ed.	29.95	29.95
Zen and the Art of the Internet—A Beginnner's Guide	23.95	23.95

Receive an additional 18% discount when ordering 10 or more of the same title of ISTE-published products.

Name Membership #

School/Business

Address

City State Zip/Postal Code

Country Phone

Shipping & Handling

$0-$15.99 (subtotal)	add $3.50
$16-$45.99 (subtotal)	$5.00
$46-$75.99 (subtotal)	$6.00
$76-$99.99 (subtotal)	$7.00
$100 or more	8% of subtotal

Do not include *additional site license fees or subscription costs* when computing shipping rates.

GST Registration Number 128828431

Code LX4

ORDER

Quantity	Title	Member Unit Price	Nonmember Unit Price	Total Price

SUBTOTAL	
Deduct 18% on ISTE-published titles if ordering quantities of 10 or more	-
SUBTOTAL	
Shipping and Handling (see box above)	+
Add Additional 5% of SUBTOTAL if shipped to PO Box, AK, HI, or outside U.S.	+
Add 7% of SUBTOTAL for GST if shipped to Canada	+
If billed with purchase order, add $4.00; If COD, add $4.50	+
TOTAL	

PAYMENT OPTIONS

□ **Payment enclosed.** Make checks payable to ISTE—
International orders must be prepaid with U.S. funds or credit card.
□ VISA □ MasterCard □ Discover Card Expiration Date
□□□□□□□□□□□□□□□□

□ **Purchase Order enclosed.** Please add $4.00 for order processing—
P.O. not including $4.00 fee will be returned.

□ **C.O.D.** for U.S. book orders only. You will pay UPS the total upon delivery by check or cash—
ISTE will add $4.50 order processing.

□ **Airmail.** International orders for Books & Courseware are sent surface mail—
ISTE will bill you the additional shipping charge for airmail.

□ Send me ISTE membership and subscription information.

□ Send me a free ISTE catalog.

ISTE • 480 Charnelton Street, Eugene, OR 97401• Order Desk 800/336-5191• Fax 503/302-3778

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.

LCSI®

And bring a little more sunshine into your classroom