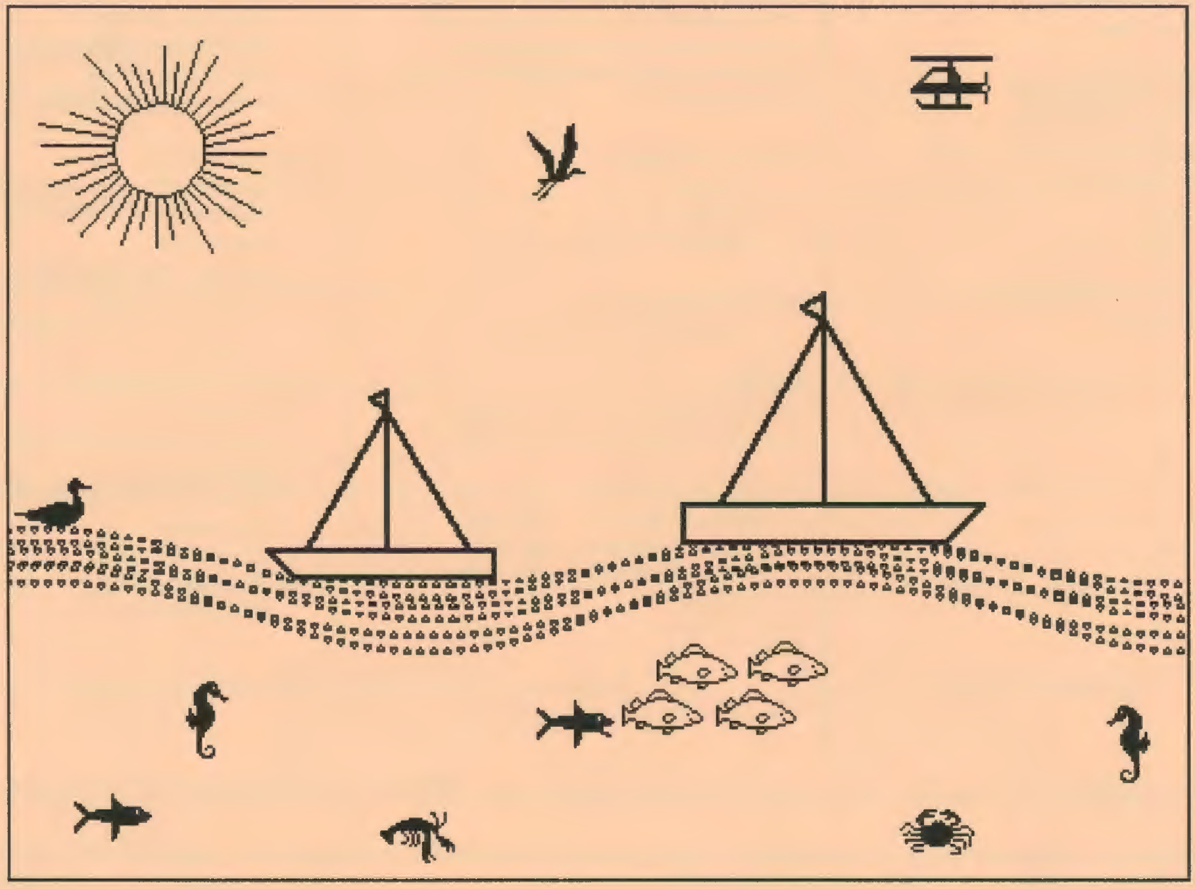
LOGO EXCHANGE

Volume 15 Number 4 Summer 1997

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



Logo Exchange **Volume 15 Number 4**

Editorial Publisher
International Society for Technology in Education

Editor-in-Chief
Dorothy M. Fitch

Copy Editor
S. M. Summerlight

Founding Editor
Tom Lough

International Editor
Dennis Harper

SIGLogo Officers
Chuck Friesen, President
Steve Sesko, Vice-President
Hope Chafiian, Secretary/Treasurer
Dorothy Fitch, Editor

Production
Kerry Lutz

Director of Advertising Services
Judy Stickney

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.

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 may be submitted for review if electronic copies are supplied upon acceptance.

Send surface mail to:

Anita Best
1787 Agate St.
Eugene, OR 97403–1923

Send electronic mail to:

Anita_Best@ccmail.uoregon.edu

Deadlines

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

Volume 16, Number 2	June 1, 1997
Volume 16, Number 3	Oct. 1, 1997
Volume 16, Number 4	Jan. 1, 1998
Volume 17, Number 1	Mar. 1, 1998

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®*.

ISTE Members may join SIG/Logo for $24.00. Dues include a subscription to *Logo Exchange*. Non ISTE Member subscription rate is $34.00. 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, 480 Charnelton St., Eugene, OR 97401-2626 USA. Periodicals 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 15 Number 4 Journal of the ISTE Special Interest Group for Logo-Using Educators Summer 1997

Contents

The cover is made from pieces of several articles in this issue. The sailboats are from Bill Spezeski's article on creating flexible shapes. The sun is drawn using the sea urchin code in John Gough's first article. The waves are made from graphing code in John Gough's second article, on algebra. The rest are shapes from Logo PLUS for the Macintosh, which the editor couldn't resist including! Have a great summer!

From the Editor: Parting Thoughts

by Dorothy M. Fitch

I write this editorial with sadness. As Tom Lough celebrates his 101st column, I write my last one. For personal reasons, I must step down as editor-in-chief of *Logo Exchange*.

I have truly enjoyed working on each issue of *LX*. What I have enjoyed most is my contact with all the authors and learning how they use Logo. I have met new friends, and gotten to know others even better. I now have faces to attach to many names.

I want to thank all the *LX* authors, ISTE staff, and SIG-Logo officers. It's been a pleasure to work with all of you. As this issue goes to press, a search continues for a new editor. Hopefully not a beat will be missed for the Fall issue. Continue to send in your submissions!

Logo saves the day!

A few weeks ago I had the delight of using Logo for a very practical purpose. In fact, Logo was the only computer tool I could think of that could do the job, or at least make it easy.

The task was to make a scale drawing of a large hayfield that is part of a nature center. The center had a trail map, but it was not as carefully drawn as we needed. We needed it to show sites for mist nets for a bird-banding station. The nets were to be at least 50 meters apart; most would be situated in the woods, just inside the edge of the hayfield. Our map needed to show the area and indicate net sites.

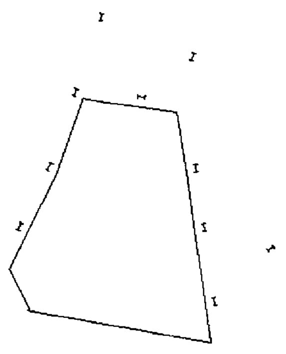
So, we gathered friends one day and set out with a 50-meter length of string, compass, 100-foot measuring tape, and notebook. We used the string to confirm that the net sites were far enough apart. While others selected net sites, two of us set out to survey the field.

My friend Hank took compass readings along each straight edge of the field and paced out each distance. He double-checked both measurements while I took notes. When we were done, I had a rough sketch, labeled with degree headings and distances for each leg. Hank measured his stride against the 100-foot tape several times to find the length of his average pace.

Someone asked if I needed to use his computer program to draw the map. I said I couldn't imagine a better tool for the job than Logo, which he didn't know about. Hank recalled that his kids had used it in school.

That evening, within a few minutes I had the outline of a perfectly scaled map. The measuring had been so carefully done that the last line came within a few dots of the starting position. All it took was a series of **setheading** and **forward** commands! Amazingly, I could use the number of paces as the input to **forward** with no scaling required.

I drew symbols for nets in the correct locations, and used Logo PLUS's **stampcircle** command to draw 50 meter circles around each one, having written a function to convert the pace measurement to meters. Here is a tiny version of the map (without the circles):



I challenge you to find another application to do this job—to handle distances, headings, and conversions—and let you make changes quickly and easily. Isn't Logo great?

Happy Logo adventures. 'Til we meet again!

Dorothy M. Fitch, LX Editor
3 Derby Road, Derry, NH 03038
E-mail: 71760.366@compuserve.com
Telephone: 603-425-2010 Fax: 603-425-6487

Quarterly Quantum: Logo 101

by Tom Lough

I just could not believe it! And yet figures do not lie (most of the time). How could it have happened so quickly? Amazing!

According to my calculations, I am now writing my 101st regular* *LX* editorial! Please permit me to ruminate on this realization for just a moment or two.

As many of you know, this journal began in September 1982 as an eight-page newsletter output from a dot-matrix printer. From the very first issue, everyone involved with *The National LOGO Exchange* (as it was known then) was enthusiastically dedicated to learning and communicating the full potential of this fascinating computer language.

In editorial number one, I tried to describe this dedication by explaining our motto, FORWARD 100.

> Here at *The National LOGO Exchange*, our motto is FORWARD 100! This reflects our enthusiasm toward LOGO and its role in education. We are very excited about the potential represented by LOGO. We want to press on in a FORWARD direction with our efforts to bring LOGO to a position from which it can influence our country's children. We want to give 100% support to you, the LOGO teachers and parents, as you work with LOGO in the classrooms and homes. Please feel free to adopt this motto as your own, for LOGO, and for life!
>
> —*The National LOGO Exchange*, Vol. 1, No. 1, September 1982, page 2.

Since then, the idea of FD 100 has spread worldwide. With the launching of *The International LOGO Exchange* newsletter in 1986 and its later combining with the *NLX* into the present *Logo Exchange* in 1987, the publication you are holding established itself as a global Logo clearinghouse. Logo ideas from the *LX* have influenced hundreds of thousands of children in countries all around the earth.

I did not set off on this journey with the goal of writing 100 editorials. Along the way, I have had the opportunity to work with dedicated educators such as Glen Bull and Steve Tipps, with enthusiastic colleagues such as Sharon (Burrowes) Yoder and Michael Tempel, and with gifted visionaries such as Seymour Papert. Inspirational co-workers such as these give wings to my feet (and to my typing fingers).

In actuality, this 101st editorial is a testimony to the work of you, our readers, as well as to the power of Logo to engage, to challenge, and to stimulate. You have brought this computer language to your children, you have nurtured them as they encountered amazing ideas and discoveries, and you have followed them as they built their own knowledge and then applied it in wonderful and unanticipated ways. On behalf of the children of the world, thank you!

In this 101st editorial, I would also like to thank Dorothy Fitch for the outstanding job she had done as *LX* editor for the past several years. Drawing upon her longtime involvement with Logo, she brought to the *LX* a sense of balance, providing both challenging articles for experienced readers as well as nurturing articles for those just getting started. It is always a pleasure to work with such dedicated people as Dorothy. Thanks, D!

Just like the turtle drawing the classic square, I feel as if I am turning a corner after going FD 100. What will the next 100 steps be like? One of the thoughts I had was to use the Roman numeral C and write a procedure.

```
to C
output 100
end
```

In this way, we can express our motto as FD C. We can think back over the past 100 editorials and reflect. Then we can look to the future and wonder what challenges, what triumphs, what realizations and discoveries it will bring. This suggests, then, that our motto might become

FD C more! ▲

Tom Lough
Founding Editor
20 Whitcomb Drive
Simsbury, CT 06070
70020.223@compuserve.com

P.S. Many of the articles in the early *National LOGO Exchange* newsletters contain ideas and suggestions that are still relevant and stimulating today. These newsletters are available in microfiche form through ERIC. See your media specialist for more information.

Volumes 1–3 (September 1982 to May 1985) ED 264 842

Volume 4 (September 1985 to May 1986) ED 279 305

* Astute *LX* diehards will quickly recognize that there have been more than 101 issues published. Just to set the record straight, guest editorials (Steve Tipps, December 1983; and Elaine Blitman, November 1985), the tribute to the Challenger crew (March 1986), and the spoof April Fool issue of 1991 are not included in this tally.

Logo Workshops: Summer 1997

Stonington Retreat — Maine

June 22–28 on Deer Isle

Led by founder Laura Allen, the Stonington Retreat is a week-long opportunity for educators to learn, play, and discuss ideas about technology, Logo, and learning. This experience combines hands-on time with high level discussions about the future of learning and technology. Facilitators include Fred Martin, Robbie Berg, and Wanda Gleason.

Registration: $1,400 per person (tuition, all breakfasts, three dinners, board, scenic boat ride)

For more information, contact Laura Allen at:
303 West 66th St. 19 FE
New York, NY 10023
Telephone: 212-873-3553
E-mail: laurallen@aol.com

Summer at Spence — New York

June 23–27 in New York City

The Spence School, an independent school for girls located in the heart of New York City, just steps from Manhattan's Museum Mile, is a leader in educational technology. Share ideas with the teachers here, where educational technology is being thoroughly infused into the curriculum from grades K–12.

Registration: $590 per person

For more information, contact:
Michael Tempel at the Logo Foundation
250 West 85th Street, Suite 4D
New York, NY 10024
Phone: 212-579-8028; Fax: 212-579-8013
E-mail: michaelt@media.mit.edu
http://el.www.media.mit.edu/groups/logo-foundation/

Logosium — Seattle

July 27, 1997 in Seattle, Washington

Site:	John Hay School, 201 Garfield Street	
Time:	9:00 a.m. – 4:00 p.m.	
Fees:	Adults (over 17)	$55.00
	Children (under 18)	$6.00

This miniconference (in conjunction with NECC '97) is for Logo practitioners and people interested in finding out about Logo. We will share our thoughts and expertise on building constructivist environments and on devising projects that are educationally rich and interesting. There will also be hands-on, how-to-do-it, and what's-new sessions. Participants should bring their own best projects and hardest questions. (Children under 12 must be accompanied by an adult.)

Logosium '97 is sponsored by ISTE SIG-Logo and The Logo Foundation. To register, see the NECC '97 materials.

For more information, contact:

	Marian Rosen	mbrosen@icon-stl.net
or	Michael Tempel	michaelt@media.mit.edu

Colorado Logo Institute

July 27–31 in Grand Junction, Colorado

Mesa State College hosts its first Logo Summer Institute. Join us for four days of learning and relaxation in beautiful western Colorado. Meals and recreational activities, such as rafting on the Colorado River, are included in your registration.

Registration: $470 per person

For information, contact Michael Tempel at the Logo Foundation (see address information listed for **Summer at Spence**).

For information about housing, contact:
Ben Keefer, Continuing Education Center
Mesa State College
1175 Texas Avenue
Grand Junction, CO 81501
E-mail: keefer@wpogate.mesa.colorado.edu
Phone: 970-281-1476

Logo St. Paul — Minnesota

August 18–22 in St. Paul

Over the past 16 years, the St. Paul Logo Project has provided a comprehensive professional development program for hundreds of elementary and secondary school teachers. A limited number of places has been set aside for people from outside the St. Paul Public Schools. Here you will interact with teachers who have experience implementing Logo in a wide variety of urban school settings. Optional graduate credit is available from Hamline University.

Registration: $590 per person; $126 for university credit

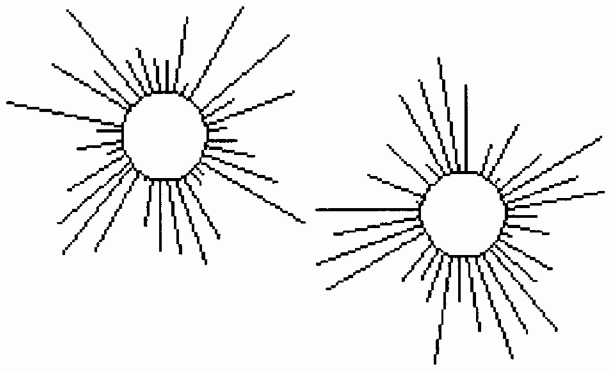
For more information, contact Michael Tempel at the Logo Foundation (see address information listed for **Summer at Spence**). ▲

Sea Urchins and Random Memories

by John Gough

I am continually trying to find new things to do with LogoWriter. Recent seaside experience suggested drawing sea urchins. It's easy to draw a circle. And it's easy to make spikes stick out from the circle as the turtle moves—that is, it's easy if the spikes are all the same length.

But what if the sea urchin has spikes of random length?



We can turn the turtle so it faces perpendicularly away from the circumference, and then move the turtle forward some random number. But how will the turtle know how far to move back so that it can turn back and resume drawing the circle? We need a way to remember the random number, or some alternative.

At first I thought of creating a global variable using a **make** command.

```
to sea.urchin
repeat 36 [fd 3 spike rt 10]
end

to spike
make "s random 50
lt 90 fd :s bk :s rt 90
end
```

But then I wondered if it could be done without using a global variable such as **"s**. A sudden inspiration suggested this alternative.

```
to sea.urchin.1
repeat 36 [fd 3 spike.1 pos rt 10]
end

to spike.1 :posit
lt 90 fd random 50
setpos :posit rt 90
end
```

This stores the turtle's current position by using the command **pos** to provide a value for the local variable **:posit** in the modified procedure **spike.1 :posit**.

Here is another way.

```
to sea.urchin.2
repeat 36
  [fd 3 spike.2 random 50 rt 10]
end

to spike.2 :size
lt 90 fd :size
bk :size rt 90
end
```

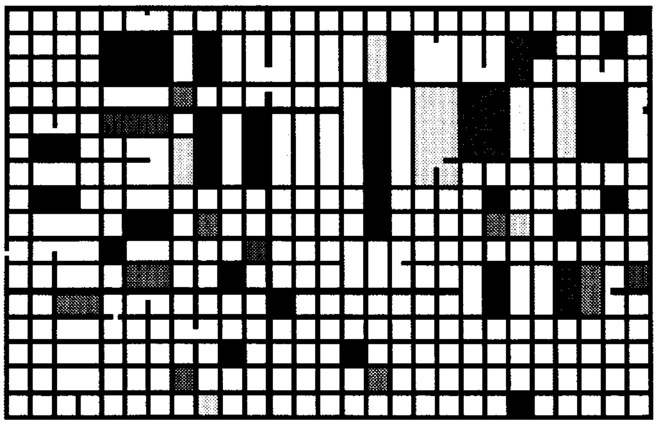
In this case, the command **random** is used once to provide the initial value for the length of the urchin spike when the modified procedure **spike.2** is called within the larger procedure **sea.urchin.2**, and this is remembered through **spike.2** as the value of the local variable **:size**.

Drawing random spikes around a circle suggested the following "tartan":

```
to tartan :unit
pu setpos [130 -80] pd
repeat 2
  [repeat 160 / :unit [fd :unit spike.2
  random 270] lt 90
  repeat 270 / :unit [fd :unit spike.2
  random 160] lt 90]
repeat 50 [spotty]
end
```

```
to spotty
pu setpos list (random 250) - 125 (random
  140) - 70
ifelse colorunder = 1 [stop] [pd setc 2 +
  random 6 fill]
end
```

Do you like this Scottish modern art?



For ***John Gough***'s biography, refer to his other article in this issue, "Seeing Algebra: My Turtle Plots Graphs."

NATIONAL • EDUCATIONAL • COMPUTING • CONFERENCE

POTLATCH

@NECC97.seattle.wa.edu

Don't miss your opportunity to experience one of the foremost K–16 educational technology conferences in the country!

featuring the US' largest national ed tech exhibit hall!

Join your national and regional colleagues this summer at the eighteenth annual National Educational Computing Conference (NECC) for three days of top-notch sessions, networking opportunities, and internationally known Keynote and Spotlight speakers—plus two days of preconference workshops, June 28 and 29.

Seattle is a breathtaking city, and we hope that in addition to enjoying all that the conference has to offer professionally, you'll take time out for a ferry ride, take in the view from the Space Needle, and dine at some of the finest restaurants and cafés in the Northwest.

For the latest information on attending or exhibiting, check us out on the Web or call us toll-free today!

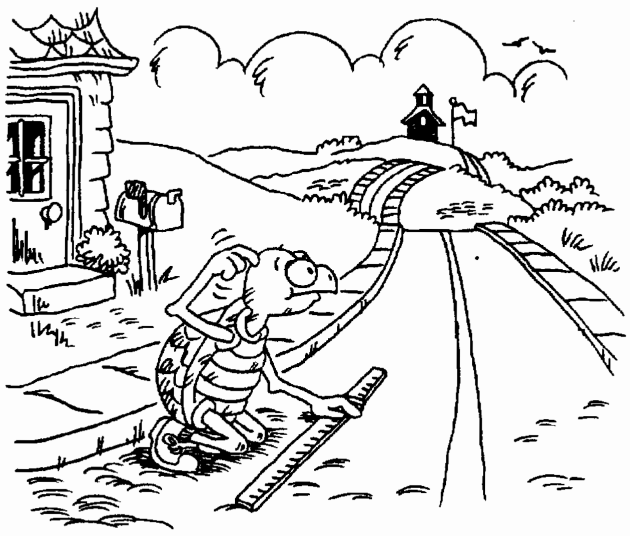
NECC '97 is hosted by Western Washington University (WWU) and is sponsored by the National Educational Computing Association (NECA).

June 30–July 2, 1997 • Seattle, Washington • 1.800.280.6218 • http://www.wce.wwu.edu/necc97

Robotics Fun: What's Your Best Guess?

reprinted with permission from LEGO DACTA CONNEXION™, Vol. 3 No. 1, Jan. 1990.

When you estimate you make calculated guesses. For example: If someone asks you how far it is to school, you might say, "about two miles." You would not be expected to give an exact answer—"two miles, three hundred fifty-seven yards and two inches!" You would be expected to estimate.



In this project you are going to use a robotic turtle to estimate distances and degrees of rotation.

This project was originally written for use with the LEGO® TC logo program. Feel free to adapt it for use with your robotics program and device.

Materials

- LEGO® TC logo turtle (or other robotic device that you can move and turn by controlling motors)
- Ruler
- Masking tape (or something to mark distance on the floor or table)
- Inventor's Notebook or sheet of paper

Setup

1. Build a robotic turtle, using LEGO activity card 9700–8, for example.
2. You must create a new page with the **turtle** procedures on it. First get the **turtle** page.
3. Then create a page called estimate. Type **namepage "estimate** in the Command Center.
4 You will need special procedures to operate the turtle. Add the procedures below to your **estimate** page.

```
to startup
cc
reset
end

to reset
alloff
recycle
talkto [A B]
setpower 5
end

to setup.l          (prepares motors to turn left)
recycle
talkto "A
seteven
talkto "B
setodd
talkto [A B]
end

to setup.r          (prepares motors to turn right)
recycle
talkto "A
setodd
talkto "B
seteven
talkto [A B]
end

to rl :degrees      (rotates motors to the left)
setup.l
onfor :degrees/4.7
reset
end

to rr :degrees      (rotates motors to the right)
setup.r
onfor :degrees/4.7
reset
end
```

For these procedures to work correctly, be sure to connect the RED plug to port A and the BLUE plug to port B. To test your connections, type **talkto [A B] onfor 20** in the Command Center. If the turtle moves forward, everything is connected properly. If the turtle turns to the right instead of going forward, turn the plug in port A around. If it goes left, turn the plug in port B around. If it goes backward, turn the plugs in both ports A and B around.

Estimating distance in inches

You are going to estimate the number of inches the turtle moves. Follow the steps below:

Step 1: Mark your turtle's starting position with a piece of masking tape.

Step 2: Use the turtle forward command (e.g., **tfd 30**) to try to move the turtle six inches.

Step 3: Mark the turtle's stopping point.

Step 4: Estimate how far the turtle moved.

Step 5: Now measure how far the turtle moved.

Step 6: Record the result in your Inventor's Notebook.

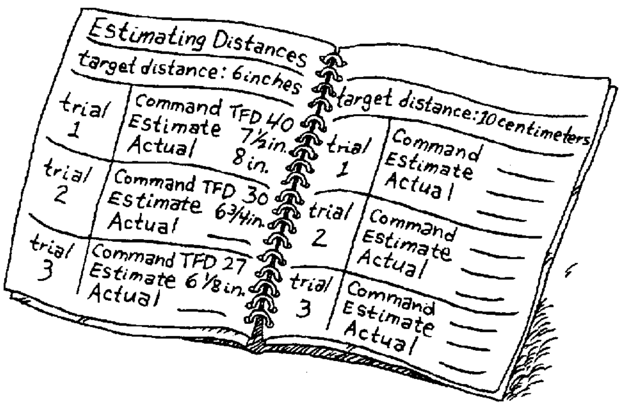
Step 7: Return the turtle to its starting position. Repeat the exercise using one **tfd** command.

Do three trials for six inches and record your results. Then try three trials for each of the following distances: 13 inches; 20 inches; your choice.

Remember to record your results!

Estimating distance in centimeters

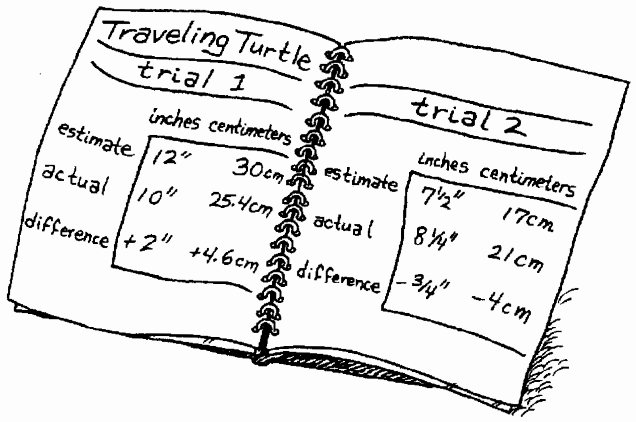
Centimeters are a unit of measure in the metric system. Follow the steps you used for estimating distance in inches to estimate the distance your turtle moves in centimeters. Do trials for the following distances: 17 centimeters; 35 centimeters; your choice.



Traveling turtle

1. Mark the turtle's starting point.
2. Have one member of your team use **gofd** or **gobk** to move the turtle forward or backward a desired distance. (The command **ot** turns off the turtle.)
3. Mark the turtle's stopping point.
4. Have everyone on the team make two estimates of the distance the turtle moved—one estimate in inches and one estimate in centimeters. Team members record each of their estimates in their own Inventor's Notebook.
5. Measure the actual distance in inches and in centimeters. Which team member came closest to estimating the actual distance the turtle traveled?
6. Record the actual distance the turtle moved.

Try at least three trials. Estimate both in inches and in centimeters. Remember to record all the data in your Inventor's Notebook.



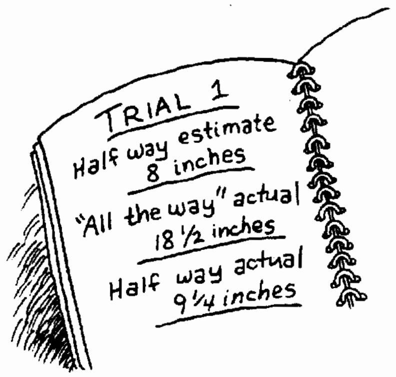
All the way there and halfway back

Here's a harder challenge!

1. Follow steps 1–3 in "Traveling Turtle."
2. Now measure the distance from the starting point to the stopping point. Let's call this the "all the way distance."
3. Record the "all the way" distance.
4. Estimate the "half way" distance. Mark your estimated distance.
5. Now measure the distance from the "all the way" stopping point to your half-way mark. Record your result.
6. Calculate exactly half of the "all the way" distance. (Divide by 2.) Record your result.

How close was your estimate to the actual "half way back" distance?

Do a few more trials. Let everyone on the team move the turtle.



Turtle turns

In this activity, you are going to estimate degrees of rotation.

1. Place a small object (LEGO brick, piece of chalk, etc.) near the turtle, but off to one side.
2. Use the **rr** (rotate right) procedure to turn the turtle so that it points directly at the object.
3. Record the command you typed.
4. If you don't reach the object or if you pass it, use **rr** or **rl** (rotate left) to try again.
5. Remember to record each command you type.

You can try the same activity using the **rl** command first.

Putting it together

1. Position your turtle in one location and a target object (LEGO brick) in another location.
2. Use the **rr**, **rl**, **tfd**. and **tbk** commands to move your turtle as close to the target as possible without hitting it. Try to use as few commands as possible.
3. Record each command as you go.
4. If you hit the target, record a "hit" and count the hit as one extra move.
5. Move the target object to another position and try again!

What other rotation/distance activities can you design to give yourself more practice?

Teaching Notes

Key ideas

1. Estimating and measuring in inches.
2. Estimating and measure in centimeters.

3. Estimating and measuring degrees.
4. Learning about acute and obtuse angles.

Materials

In the procedure **reset**, you could change the input to **setpower** to any number between 1 and 7. But remember to recalibrate angles in **rr** and **rl**.

Excess friction on gears and variations in the speed of the turtle's motors affect measurements. The surface on which the turtle travels affects its speed.

Estimating distance in inches

It is very important that students record the results of each trial as it takes place. Students should take turns controlling the turtle and marking the starting and stopping points.

Estimating distance in centimeters

This activity is a good opportunity to address students' working knowledge of the metric system, specifically regarding the relationship between a centimeter and an inch. For example, once they have measured the distance in inches, they could calculate the distance in centimeters, rather than measuring separately (1 inch = 2.54 cm).

Which was easier: estimating in inches or in centimeters? Why?

Were longer distances harder to estimate than shorter ones?

Can you think of instances when you had to estimate distance?

Traveling turtle

Students will find that it takes practice to control the turtle with the **gofd**, **gobk**, and **ot** (off turtle) commands. It may be helpful to have the practice before actually doing the activities. **Gofd** and **gobk** simply turn on the motors. **Tfd** and **tbk** turn on the motors for a specified time.

The number of estimates recorded for each trial in "Traveling Turtle" will depend on the number of students per team. You may want to have each student make estimates in both inches and centimeters. Or, if you have four students per team, you may want to have two students give estimates in inches and two give estimates in centimeters.

All the way there and halfway back

"Halfway back" provides a good opportunity to review fractional quantities and basic arithmetic skills.

As an extended or advanced activity, students could move the turtle a different fractional quantity (e.g., one third the original distance).

Students can use the **show** command to calculate the exact half distance. For example: If the turtle travels 11.5 inches, type **show 11.5/2** in the Command Center to calculate half the distance.

Which was easier: estimating the original distance (in "Traveling Turtle") or half the distance back?

For what occupations are estimation skills important?

Can you think of any situation in which estimation isn't enough and accuracy is necessary?

Turtle turns

In this activity, students are required to use a specific command first so that they will increase their ability to rotate more than 180 degrees. They will observe that turning a large angle in one direction can be equivalent to turning a small angle in the opposite direction.

You may find that the **rr** and **rl** commands are not very accurate. If that is the case, you will need to change the calculations in one or both procedures. (This is necessary because not all models are built exactly the same, and may rotate at different speeds.)

If the turtle doesn't rotate far enough, change the number 4.7 in the procedure to 4.9 or 5.1. If the turtle rotates too far, change the number 4.7 to 4.3 or 4. Continue to adjust these numbers until the turtle approximates closely the actual rotation in degrees.

You may want to make a practice page with different angles drawn on it (e.g. 45°, 90°, 135°, 180°, and so on). Then you could test your turtle's accuracy by putting it in the center of the practice page, typing **rr 45** to see if the turtle turns 45 degrees.

Rr and **rl** may not work with numbers less than 5 or 6, depending on your calibration. The numbers may simply be so small that the motors turn on and off too quickly to move the turtle.

Once the results have been recorded, they provide valuable discussion material. Students should see that their accuracy increased with successive trials; their first trial was probably always less accurate than their third trial.

Discussion

Which do you estimate more accurately, rotation or distance? Why?

Do you think the people who make the LEGO building materials estimate or use exact calculations to make LEGO bricks?

What other kinds of estimations do we use besides rotation and distance? ▲

A Challenging Game of Nim

by Jim Muller

The game of Nim is an old favorite that you may remember from your youth. When you first began to play, you could never win. However, after you learned the secret, you could never lose.

The game is easy enough. Two players start out with a small pile of toothpicks or stones. The players decide how many pieces each can take on a turn and how many turns each will have. The whole idea is not to get stuck taking the last piece.

One nice thing about Nim is how easy it is to write in Logo. Here is the game in PC Logo.

```
to nim
cleartext textscreen
repeat 5 [pr "]
pr [Welcome to the game of nim!]
pr "
pr [This is a game where you and the]
pr [computer take turns picking stones]
pr [from a pile. The challenge is not]
pr [to get stuck with the last stone.]
pr "
pr [You decide. How many stones will]
pr [each player be able pick on each turn?]
make "pick first readlist
make "picks :pick + 1
pr [How many turns will we have to pick?]
make "turns first readlist
make "total :picks * :turns + 1
game
over
end

to game
(pr [There are] :total [stones in a pile.])
(pr [Take from 1 to] :pick [stones.])
make "key get.number
pr "
(pr [I take] word :picks - :key ".)
pr "
make "total :total - :picks
if :total > 1 [game]
end

to get.number
make "key rc
if :key > :pick [pr " pr [That's too many.
  Try again!] output get.number]
if :key < 1 [pr " pr [Pick at least one.]
  output get.number]
output :key
end

to over
repeat 3 [pr "]
pr [There is one stone left. I win again.]
end
```

Take a good look at this procedure. Do you see why the computer always wins?

The answer is right there in the simple formula that follows, "I take" in the **game** procedure. The computer always takes **:picks - :key**. Do the math yourself. You will soon see why there is no way the computer can lose.

PC Logo uses classic Logo, even in its Windows package. The procedure shown below, however, was written using the freeware package, MSW Logo.

This package is unique in that it adds a full range of Windows features to explore. A few of these are included in the version of the game listed below:

```
to nim
ct
pr [Welcome to the Game of NIM!]
pr "
pr [This is a game where you and the]
pr [computer take turns taking stones]
pr [from a pile.  The challenge is not]
pr [to get stuck with the last stone.]
pr "
pr [You decide. How many stones will each]
pr [player be able to take on each turn?]
selectbox [Pick a number.]
make "pick :key
make "picks :pick + 1
pr [How many turns will each player have to
  TAKE stones?]
selectbox [Pick a number.]
make "turns :key
make "total :picks * :turns + 1
game
over
end
```

```
to game
(pr [There are ] :total [ stones in a
  pile.])
(pr [Take from 1 to ] :pick [ stones.])
selectbox [Pick a number.]
if or (:key > :pick)(:key < 1) [correct
  stop]
(pr [I take] word :picks - :key ".)
make "total :total - :picks
if :total > 1 [game]
end

to over
pr [There is one stone left. I win again.]
make "ans yesnobox [nim] [Want to play
  again?]
ifelse :ans = "true [nim] [ct pr [Bye for
  now!]]
end

to selectbox :question
pr :question
dialogcreate "root "Selector "SelectBox 250
  20 100 180 [selectbox.setup]
end

to selectbox.execute
if radiobuttonget "one [make "key 1 del
  stop]
if radiobuttonget "two [make "key 2 del
  stop]
if radiobuttonget "three [make "key 3 del
  stop]
if radiobuttonget "four [make "key 4 del
  stop]
if radiobuttonget "five [make "key 5 del
  stop]
if radiobuttonget "six [make "key 6 del
  stop]
if radiobuttonget "seven [make "key 7 del
  stop]
if radiobuttonget "eight [make "key 8 del
  stop]
if radiobuttonget "nine [make "key 9 del
  stop]
end

to selectbox.setup
groupboxcreate "Selector "nim 10 10 80 140
radiobuttoncreate "Selector "nim "one [One]
  20 15 60 15
radiobuttoncreate "Selector "nim "two [Two]
  20 30 60 15
radiobuttoncreate "Selector "nim "three
  [Three] 20 45 60 15
radiobuttoncreate "Selector "nim "four
  [Four] 20 60 60 15
radiobuttoncreate "Selector "nim "five
  [Five] 20 75 60 15
radiobuttoncreate "Selector "nim "six [Six]
  20 90 60 15
radiobuttoncreate "Selector "nim "seven
  [Seven] 20 105 60 15
radiobuttoncreate "Selector "nim "eight
  [Eight] 20 120 60 15
radiobuttoncreate "Selector "nim "nine
  [Nine] 20 135 60 15
radiobuttonset "one "true
radiobuttonset "two "false
radiobuttonset "three "false
radiobuttonset "four "false
radiobuttonset "five "false
radiobuttonset "six "false
radiobuttonset "seven "false
radiobuttonset "eight "false
radiobuttonset "nine "false
buttoncreate "Selector "game "OK 35 150 25
  20 [selectbox.execute]
end

to correct
messagebox [nim] [That number won't work.
  Try again!]
game
end

to del
dialogdelete "Selector
end
```

Challenges

Before exploring the opportunities of Windows programming, why not consider some of the challenges open to you regardless of the Logo dialect you are using?

Why not make it a graphical game? Create pictures of the pieces used in the game—things like stones, toothpicks, rabbits, or cars. This really doesn't change the game. It just makes it easier to look at.

How can you change the procedure to make it more of a challenge? In other words, can you change it so that the computer doesn't always win?

The easy way is simply to add a random element. The computer takes a random number of pieces on each turn. If you do your arithmetic correctly, you have a good chance of turning the tables on the computer. Of course, many games are fun because they have a degree of randomness that keeps things interesting. However, randomness does not do much to develop strategic thinking.

How can you make it tough on both the player and the computer?

One way to add a strategic element to the game is to distribute the pieces on multiple rows. Have the players select pieces from just a single row on any one turn. For example, a real challenge can be developed by using pieces placed on three rows. However, as you can guess, you will get to the point where you have one or two pieces on each row. Your original algorithm does not work in such situations, so you'll have to add new rules for the end of the game.

Writing game procedures is fun, regardless of the Logo dialect you use. Games are fun to play, but the novelty can soon wear off. This is where a good game design challenge can end up being more fun than the game itself.

Why windows programming

Windows programming supports two modes, Modal and Modeless. Modal programming is similar to non-windows programming. When running Logo, the procedure controls the action. For example, during processing, you (as the programmer) decide to ask the user for information using **readlist**, **readword**, or **readchar**. Everything stops until the user enters the requested information.

In Modeless programming, the tables are turned. The window (actually the user) is in control. This takes some getting used to; however, it is a very important concept to learn. Using the example above, rather than halt everything while waiting for the user to respond, Logo is merely idle.

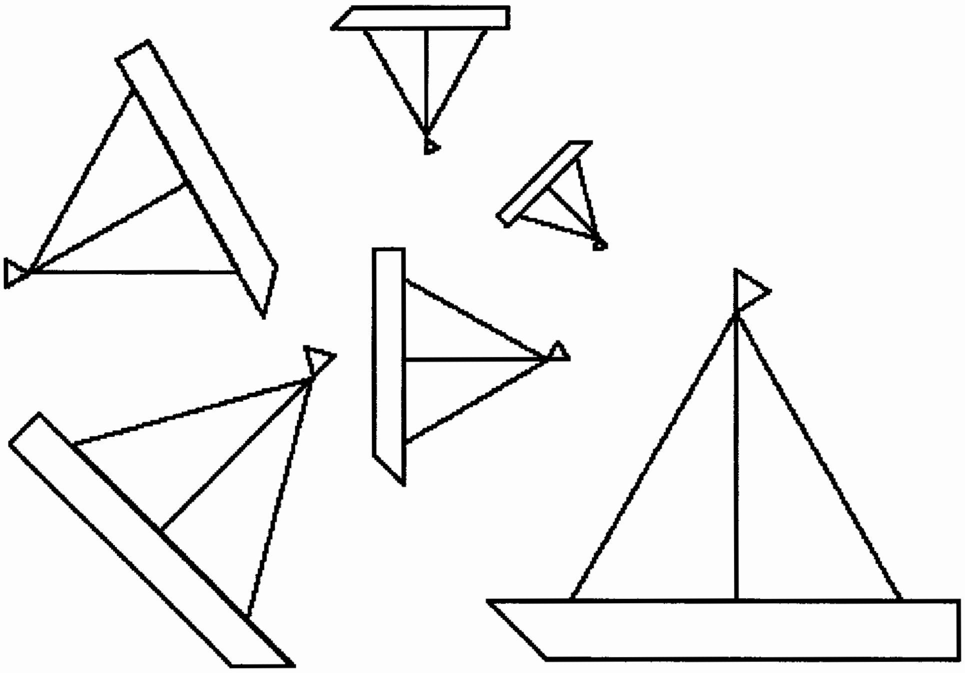
While Logo is idle, you can change the colors being used, for example. You can even write an execute another procedure. The original procedure continues to wait until you trigger an appropriate event such as pushing a button. For example, if the game of Nim was done using Modeless programming, you go off and test out a turn before entering it in the real game.

Why program with windows? Try to imagine what a child's imagination could do in such a multi-event environment. ▲

Jim Muller has had a lifelong interest in translating various technologies into understandable and persuasive programs. In 1981, Muller and his son organized the first Logo users group, the Young Peoples' Logo Association, which eventually grew into a worldwide 6,000-member organization. In 1985, the YPLA merged with CompuServe, where it became The Logo Forum. Today, Muller is a computer training and marketing consultant in the Dallas/Fort Worth metroplex.
E-mail: 76703.3005@compuserve.com

Drawing Objects for Flexibility

by William J. Spezeski



Consider the sailboats shown above. They appear in different sizes and orientations. But most important, these sailboats were drawn using a single Logo drawing procedure. Writing procedures that allow this kind of flexibility is the subject of this article.

Making objects movable

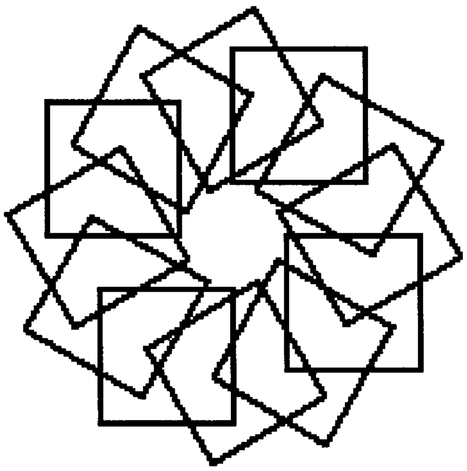
Creating objects that can be positioned anywhere on the screen is relatively easy. The secret is to restrict the commands in the drawing procedure to those that result in relative turtle movement (i.e., **fd**, **bk**, **rt**, **lt**). Avoid commands that result in absolute turtle movement such as **setxy, seth,** and **home**. The effects of these two categories of commands can easily be seen in the two procedures below:

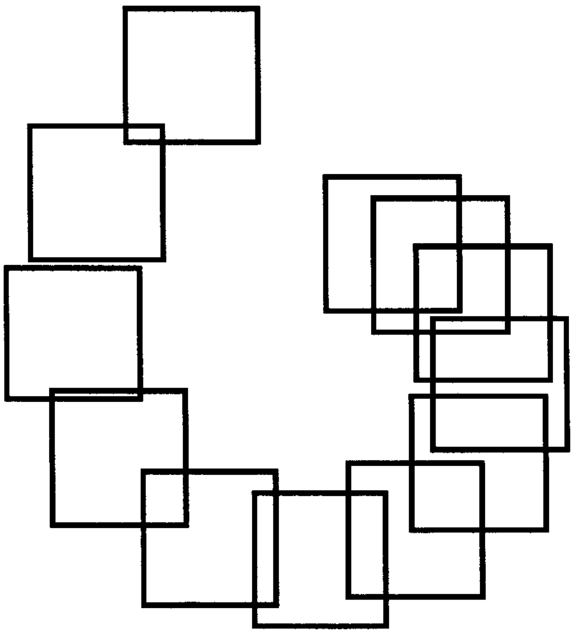
```
to square
; relative turtle movement
repeat 4 [fd 50 rt 90]
end
```

```
to square1
; absolute turtle movement
pu home pd
setxy [ 0 50]
setxy [50 50]
setxy [50  0]
home
end
```

Both procedures draw squares of side length 50. Procedure **square** uses only relative turtle commands, while procedure **square1** uses absolute commands to accomplish the task. But only procedure **square** has the flexibility to draw the square at various locations and alignments on the screen.

Once an object is created by a procedure using relative turtle movement, it can be repositioned anywhere on the screen by simply changing the initial drawing coordinates of the turtle. Likewise, the object can be rotated to any heading by selecting a new drawing heading for the turtle. The following two pictures show the same square procedure drawn at various positions and headings.





Another sound practice to follow when drawing objects is always to return the turtle to its original starting position in order to complete the drawing. This is not a requirement, merely a strong suggestion. But it certainly makes life easier to know where the turtle is when the task of drawing the object is complete. Returning the turtle to its starting position becomes particularly important when you work with animation. In the interest of brevity, the procedure that draws the sailboat in this article does not return the turtle to its starting position.

Making objects scalable

Objects that are movable have a useful level of flexibility. But there are other levels as well. Creating objects whose size can be changed is another desired drawing feature. This can be done in a straightforward manner by adding a parameter to the drawing procedure that controls the size (scale) of the figure being drawn. Let's first take a look at the technique for scaling a figure and then discuss why it works. The process is easy and mechanical and done with the Logo editor.

To add a parameter to a procedure that draws an object so that the object can be effortlessly resized, do the following:

Step 1. Add a parameter to the title line (say **:x**)

Step 2. Append the expression *** :x** to each number in the body of the procedure that is related to either forward or backward turtle movement. For example, **fd 20** would become **fd 20 *:x**.

Example:

```
to example      becomes      to example :x
  fd 50         becomes        fd 50 * :x
  rt 45                        rt 45
  bk 20         becomes        bk 20 * :x
end                          end
```

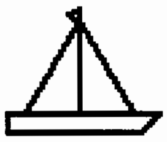
Explanation:

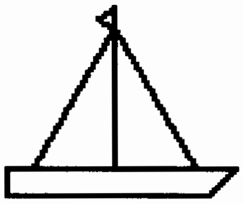
The parameter **:x** is actually a multiplier. It will modify the linear turtle movement by a factor of **:x**. If **:x** has a value of 2, the figure will be drawn twice as large as when originally created. If **:x** has a value of .5, the figure will be half of its original size.

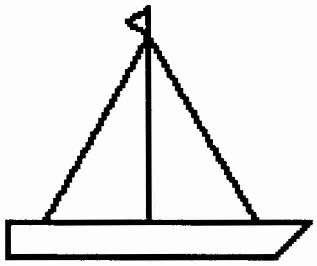
The key element of this technique is that it preserves all of the proportions of the original figure. Once the figure has been created, the hard work has been done. Modifying the corresponding procedure is an exercise in using the Logo editor. Note that all angles must remain the same. If they are changed, then the figure will be changed and in some cases totally unrecognizable. An example of a procedure that draws a sailboat and its modified version that scales the size of the boat are shown below.

```
to sailboat
rt 30 fd 80 rt 120 fd 80
lt 240 fd 80 bk 40 rt 90
rt 90 fd 60
rt 135 fd 20
rt 45 fd 100
rt 90 fd 14
rt 90 fd 114 bk 60 lt 90
fd 80 bk 10 lt 60
fd 10 rt 120 fd 10
end
```

```
to sailboat :x
rt 30 fd 80*:x rt 120 fd 80*:x
lt 240 fd 80*:x bk 40*:x rt 90
rt 90 fd 60*:x
rt 135 fd 20*:x
rt 45 fd 100*:x
rt 90 fd 14*:x
rt 90 fd 114*:x bk 60*:x lt 90
fd 80*:x bk 10*:x lt 60
fd 10*:x rt 120 fd 10*:x
end
```


SAILBOAT .5


SAILBOAT .75


SAILBOAT 1

Flipping: changing the direction of an object

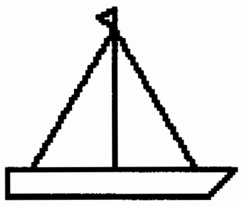
Many objects, like a sailboat or an airplane, tend to point in a specific direction, and it is often desirable to draw a similar sailboat or airplane that points in the opposite direction. This change of direction is sometimes referred to as a horizontal flip. The object is flipped horizontally about an imaginary vertical axis.

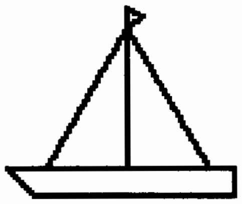
The ability to flip an object adds to its versatility. It can be accomplished in one of two ways. One technique is to take the drawing procedure and replace all of the **lt** commands with **rt** and replace all of the **rt** commands with **lt**. Thus the instruction **rt 60** would be replaced by **lt 60,** and the instruction **lt 45** would be replaced by **rt 45.**

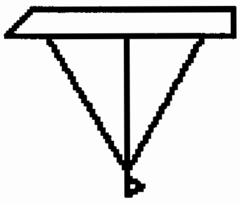
Alternately, we can multiply the inputs of all **rt** and **lt** instructions by –1. Using this technique, we would replace the command **rt 60** with **rt –60** and **lt 45** with **lt –45.** The second technique lends itself nicely to using a parameter to handle the multiplication (change of +/– sign).

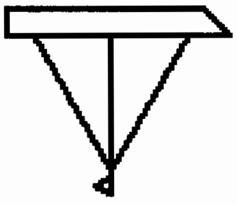
Shown below is procedure **sailboat**, modified with a second parameter that can be used to flip the boat horizontally.

```
to sailboat :x :y
rt  30*:y fd  80*:x rt 120*:y fd 80*:x
lt 240*:y fd  80*:x bk  40*:x rt 90*:y
rt  90*:y fd  60*:x
rt 135*:y fd  20*:x
rt  45*:y fd 100*:x
rt  90*:y fd  14*:x
rt  90*:y fd 114*:x bk  60*:x lt 90*:y
fd  80*:x bk  10*:x lt  60*:y
fd  10*:x rt 120*:y fd  10*:x
end
```


SAILBOAT 1 1


SAILBOAT 1 –1


SAILBOAT –1 1


SAILBOAT –1 –1

The orientation of these sailboats can be summarized by the following:

:x > 0,	:y = 1	original figure
:x < 0,	:y = 1	figure rotated 180°
:x > 0,	:y = –1	horizontal flip
:x < 0,	:y = –1	vertical flip

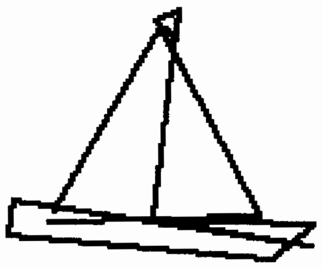
When positive scaling values of **x** are selected, the original sailboat (:y = 1) or a horizontally flipped sailboat (:y = –1) is drawn, as shown above.

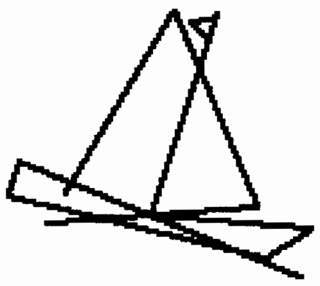
Using negative values for **x** produces a couple of interesting side effects. When negative scaling values of **x** are selected, the sailboat is turned upside-down from its original orientation, producing either a rotated sailboat (:y = 1) or a vertically flipped sailboat (:y = –1).

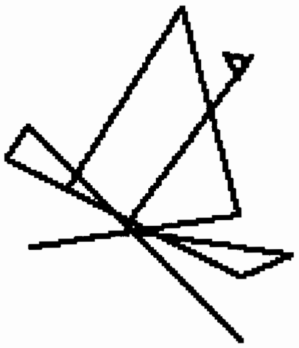
With very little effort, we have crafted a powerful procedure. The only "hard work" involved the creation of the initial sailboat.

Deforming an object

There is yet another interesting side effect of using parameters. When values of **y** other than 1 or –1 are selected, the figure becomes distorted. That is because the object's angles are not intended to be scaled. The distorted figures that are produced can be very interesting. In some cases, the deformed figures resemble a Cubist's interpretation of the figure. Some examples of this phenomenon are shown below.


SAILBOAT 1 1.02


SAILBOAT 1 1.05


SAILBOAT 1 1.10

Conclusion

The more flexible a procedure is, the more useful and valuable it is. Procedures that draw objects can be easily modified to enable their objects to be scaled and flipped. These added levels of flexibility not only make many more Logo creations easier but also possible in some cases. The net result is much more enjoyment with Logo. Try some of these ideas and see if you don't agree. ▲

William J. Spezeski is an Associate Professor in the Computer Science Department at North Adams State College in North Adams, Massachusetts. The department offers an elective course in problem solving that is built around Logo. The course features a variety of problems for computer solution emphasizing top-down design and modular programming. The text for the course is his book, "Logo: Models and Methods for Problem Solving," published by Harvard Associates, Inc.
E-mail: wspezesk@nasc.mass.edu
Computer Science Department
North Adams State College
North Adams, MA 01247
413-662-5591; 413-662-5010 (fax)

Musings: Language, Grammar, and Logo

by Robert Macdonald

In his 1973 series of six Charles Eliot Norton lectures at Harvard University, Leonard Bernstein examined Noam Chomsky's generative theory of language, in which the nature of linguistic universality holds a central position (Chomsky, 1957). Reduced to simplicity, Chomsky espoused that all mankind is aware of the generative principles concerning language at birth. He claims that these principles exist in every language and thus constitute a universal grammar.

Bernstein then decided to investigate musical universality by means of a linguistic analogy. He outlined his approach:

> Now, luckily, the study of linguistics is a threefold science, and so provides us with three handy categories in which these lectures can be conceived: Phonology, Syntax, and Semantics. These are the three departments of linguistics, and they point the way for our musical investigation as well. In this first lecture we'll be oriented phonologically, examining both language and music from the most fundamental point of view—that of sound itself, the stuff of which verbal and musical utterances are made. That should give us a solid base to build on, so that in our next session we can plunge into syntax, the actual structures that arise from that phonological stuff. From then on, the remaining four lectures will confront the challenges of semantics, that is *meaning,* both musical and extramusical meaning. Semantics can be seen as the natural result of adding phonology and syntax together.... (Bernstein, 1976 p. 9)

Bernstein's approach enthralled me for several weeks as I worked through the six lectures named *The Unanswered Question*—a title borrowed from a remarkable composition by Charles Ives. Then I kept reverting not to the musical illustrations but to the original linguistic analogy as I reflected on my own language experiences with children. However, we will find it beneficial to use Bernstein's outline as offered above in the following article, moving from phonology to syntax to semantics.

World languages provide us an alluring array of linguistic similarities and differences. We have two options open to investigate them: We may opt to investigate features that most languages have in common, or we may opt to investigate their differences. Chomsky's approach of commonality is more appealing. And this article will deal with an amateur's play on the philosophical investigations of the great.

Phonology

To begin our study of language we turn to the basic materials—sounds. One of the easiest ways to survey the statistical regularity of any language is to count the different aspects that may recur: sounds, words, and so on. We propose a simple enough task to demonstrate a point.

How often do the letters of the alphabet (a fairly adequate indicator of sounds) occur in a selected text? You may have the students conduct a "field test." Cut out sample paragraphs from a daily newspaper and have the students tally the number of times each letter of the alphabet occurs in their selection. It shouldn't surprise you that the order of frequency closely adheres to the order in which sounds are presented to initial readers. I shall omit digraphs (th, wh, ng, sh, ch), diphthongs (oi, oy, ou, ow, au), and other little complications.

a n t e b s r i d l o m u c
v p f w h k g y j z x q

The listing above corresponds somewhat closely to the finding of Zettersten (1969, p. 21). He discovered the following average rank order of sounds (represented by the alphabet) in a comparative study of 15 different categories of over 1 million words, ranging from press reporting through scientific writing:

e t a o i n s r h l d c u m
f p g w y b v k x j q z

Over the years, in watching the television program *Wheel of Fortune*, I've deduced that contestants who tend to follow the frequency of the occurrences of letters of the alphabet have a better chance of winning.

Now carry on the task to the next higher level of phonology: sounds into words. How often do words appear within a selected text? It is now that word lists become important. Again compare selected texts from a daily newspaper. This time tally the frequency in which words occur. From the Kucera–Francis Word List, I provide the first 20 of the most common words as they appear in readers for younger children:

the of and to a in that is was he
for it with as his on be at by I

How does this compare with the tally of the frequency of words in your selected paragraph?

Another manner of statistically delineating frequency in the study of language is to study the frequency of words in relation to their length. How does the relationship of syllable length compare to the frequency of a word's occurrence in the language? Again select paragraphs from the newspaper, but this time do your "field study" on the occurrence of the length (number of syllables) of a word within a selected text. I think you might strongly suspect that as the number of syllables increases, the words occur with much less frequency. Indeed, George Kingsley Zipt (1902–1950), a noted American philologist, proved

> that there was an inverse relationship between the length of a word and its frequency. In English, for example, the majority of the commonly used words are monosyllables. The same relationship obtains even in a language like German, which has a marked "polysyllabic" vocabulary. This effect seems to account for our tendency to abbreviate words when their frequency of use rises, e.g. the routine reduction of *microphone* to *mike* by radio broadcasters. It would also seem to be an efficient communicative principle to have the popular words short and the rare words long. (Crystal, 1987, p. 87)

Syntax

We have progressed from sounds to words. We now use words to communicate. These words form a framework that gives cohesion to our thought. In brief, we are now ready to undertake a study of grammar. (I may be excused from even considering that branch of grammar, morphology, that studies the structure of words. I have little taste for digressing into morpheme problems, which I can't imagine will help in the following discussion.)

David Crystal (1987, p. 91) describes words as sitting "uneasily at the boundary between morphology and syntax." Words are certainly the easiest parts of a written language to identify. In most written languages they are the entities that have spaces on either side. In a spoken language, we do not hear these spaces between words. When embarking on the adventure of listening to a new language, this provides all of us with many of our initial problems.

Contemporary linguists are not fond of applying definitions embodied in traditional grammar. They prefer *word classes* to *parts of speech*. It embodies a change in emphasis. But such in-depth analysis is really beyond what we shall attempt below. We shall assume that all of our readers recognize the traditional classification of parts of speech (nouns, pronouns, adjectives, verbs, prepositions, adverbs, conjunctions, and interjections).

Some computer applications

In the above "field tests" we have examined the phonological and syntactical aspects of our language. Let's construct some Logo procedures that will help us collect "some actual structures that arise from that phonological stuff." Here the actual structures will be some of the familiar parts of speech. With these we will have the tools for undertaking a brief study of semantics (meaning).

The following procedures will be built over a period of weeks. It is, after all, a wonderful way to make technology a vital part of a normal classroom environment. When the students understand the premises on which the procedures are built, they should be responsible for constructing their own samples. I have done exercises such as these with fourth graders. Many students entering my class had more than enough third-grade experiences with parts of speech to undertake this fairly early in the year. I explain only as the need arises.

The first procedure we will need is a **pick** procedure. If your version of Logo includes a **pick** primitive, you do not need to define it as a procedure. We will use this procedure frequently in what follows. Because the students will be collecting parts of speech in many of the procedures that follow, we need a procedure that will pick, for example, one noun out of many that the students provide.

```
to pick :list
output item (random count :list) + 1 :list
end
```

In the following procedure the students will use the command **output**. The command takes one input. In the procedures below, that input is a list of words; these, of course, are enclosed in brackets.

```
to noun
output pick [dogs cats tigers teachers
  students principals mothers fathers
  brothers sisters]
end

to verb
output pick [run jump teach growl play
  scream hit talk bark study meow]
end
```

You might note that I have capitalized each adjective below. Do you see why? It will become evident as you move on.

```
to adjective
output pick [Big Little Yellow Red Tall Thin
  Fat Striped Hairy]
end
```

To help with the chore of typing, I have shortened the word **adjective** to **adj**. I will do the same thing for some of the other parts of speech, and you may wish to do the same.

```
to adj
output adjective
end

to adverb
output pick [hungrily lovingly greedily
  kindly quickly immediately slowly]
end

to adv
output adverb
end
```

Please note the brackets in this sample.

```
to prepositional.phrase
output pick [[in the park.] [in the office.]
  [in the zoo.] [in the class.] [in the
  kitty litter.] [around a tree.] [up on the
  housetop.] [under the desk.]]
end

to prep
output prepositional.phrase
end
```

You might wish to note the bracketing in this procedure.

```
to interjection
output pick [Alas! [Good Gracious!] Help!
  Hurry! Stop! [Holy Smokes!]]
end

to interj
output interjection
end
```

Note that I have not included a procedure with conjunctions. If you wish to expand my approach, please do.

Now let's try out ways to use these procedures. In the Command Center enter the following commands:

```
print noun
```

The following appears in the work area:

```
brothers
```

Now enter:

```
repeat 5 [print adj]
```

The following appears in the work area:

```
Red
Fat
Big
Yellow
Big
```

```
repeat 5 [type adj]
```

In the Command Center, the following appears with no carriage return:

```
TallStripedHairyLittleHairy
```

To insert spaces, enter:

```
repeat 5 [type adj type char 32]
```

(Your version of Logo may require the use **print1** instead of **type**.) The following appears with no carriage return:

```
Big Fat Tall Yellow Big
```

Now enter:

```
show prep
```

In the Command Center, the following appears with a carriage return:

```
[in the office.]
```

Now enter:

```
repeat 5 [show prep]
```

In the Command Center appears:

```
[around a tree.]
[in the zoo.]
[up on the housetop.]
[in the zoo.]
[in the park.]
```

We shall delay combining parts of speech until we consider some other ideas.

Semantics

Native speakers take the structure of language for granted. We are so conditioned to speaking and writing our mother tongue that we lose sight of the linguistic complexities that form a kind of underpinning for almost every sentence. If we were to analyze only a part of these complexities, we might understand why a child in the initial stages of comprehending his language may utter an infinite number of sentences while applying only a finite number of grammatical patterns.

All of us render meaning to our thoughts by adhering to the paradigms of our language. We do this whether or not we are aware of the patterning. Even silly sentences (one in which words do not seem to make an intelligent interplay), if they follow structure, do not alarm us. Let me demonstrate with some silly sentences.

Silly sentences

English sentences are highly positional. We find comfort in the order of **subject**, **predicate**, **object**:

```
Cows chew grass.
```

To this basic frame we may add adjectives, which generally precede their nouns, adverbs, and prepositional phrases and are located near the words they enhance. Interjections generally appear at the beginning of a linguistic structure:

```
Hurrah! Big brown cows quietly chew grass in
  the lovely green meadow.
```

Because of the positional characteristic of English, we can create an exercise much like the one described below with little effort.

How many different silly sentences could you build from the following words collected in eight different columns? Without giving the task much thought, quickly read any word from column 1, then any word from column 2, and so on through column 8. Any sentence you read will sound a bit peculiar and might cause concern about semantics. Syntactically, the sentences do not disturb our grammatical sensitivity.

Column 1	**Column 2**	**Column 3**
fourteen	silly	rockets
many	circular	fish
some	checkered	camels
several	crazy	snowballs
seven	violet	hamburgers

Column 4	**Column 5**	**Column 6**
danced	lazily	through
fell	gracefully	down
hopped	merrily	over
climbed	happily	up
skipped	dolefully	under

Column 7	**Column 8**
any	staircase
a	cloud
the	parade
each	hilltop
every	tree stump

The language encounter above was one I used with first graders for years. They enjoyed it. It is also an easy task to program the above activity on a computer. In this example, rather than apply the primitive **output**, I create a series of variables with the command **make**. I place all of this in a startup procedure, clear the page, and order a single printout. As with the procedures above, this short program is written in LogoWriter for the Macintosh.

```
to startup
clearpage
make "columnl [Many Three Several Some
  Forty]
make "column2 [crazy violet striped
  checkered silly]
make "column3 [rockets hamburgers fish
  snowballs camels]
make "column4 [fell climbed hopped swam
  danced]
make "column5 [quickly lazily merrily
  awkwardly gracefully disdainfully]
make "column6 [down over through under up
  between]
make "column7 [a each every any the]
make "column8 [cloud. log. staircase.
  parade. mountain. skyscraper.]
print.out
end
```

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

```
to print.out
(print (pick :columnl) (pick :column2) (pick
  :column3) (pick :column4) (pick :column5)
  (pick :column6) (pick :column7) (pick
  :column8))
end
```

If you wish multiple printouts, after you have received the one provided by the program, you may write the following in the Command Center:

```
repeat 10 [print.out]
```

A reminder: If you are entering this program initially and not bringing it up from the Contents page, you will have to give the command **startup**, or you will not initialize your variables.

Below are some samples:

```
Forty crazy snowballs swam disdainfully up
  any mountain.
Some violet fish fell gracefully between
  each log.
Three checkered hamburgers hopped gracefully
  under every mountain.
Three crazy snowballs fell gracefully up
  each staircase.
Forty checkered hamburgers fell lazily over
  any staircase.
Three silly rockets fell disdainfully over
  the mountain.
Some crazy camels hopped lazily under the
  mountain.
Many crazy snowballs fell disdainfully
  between every skyscraper.
Several silly hamburgers hopped disdainfully
  down any mountain.
Some crazy snowballs swam lazily over every
  mountain.
Forty violet hamburgers fell merrily between
  each skyscraper.
```

Twenty years ago it was possible to find many children's books that made use of this positional characteristic. The pages of these books were cut horizontally (often 4 to 6 cuts per page), and the spiral binding permitted any or all the pieces of a page to be turned at will. In my last survey of book stores and libraries, I was unable to locate even one. Surprisingly, when I inquired, few of my former teaching colleagues had any knowledge of such books.

We have considered the positional element that exists in English, so let's now combine commands for the parts of speech, which we provided as procedures above. We will use the **print** command to combine the output. Please notice the use of parentheses, which helps you avoid using the reporter **sentence** (**se**). Type the following in the Command Center.

```
(print noun verb)
(print adj noun verb)
(print adj noun verb adv)
(print adj noun adv verb)
(print adj noun adv verb prep)
(print interj adj noun adv verb prep)
(print prep adj noun adv verb)
```

As you play around with the procedures, the output clearly demonstrates the positional features of English. I have made my own lists conform to standard practice. Consequently, I have capitalized the adjectives and placed a period at the end of my prepositional phrases. If you wish, you might modify these to enrich your own undertakings. How would you change the material to produce a question? Inverted sentences indicating a question are almost mandatory in English, unless you wish to resort to inflecting your tone of voice in a declarative sentence to impress on the listener that you imply an interrogation.

If we so desired, we might construct programs in which the structure of a sentence (or a clause) is given functional labels. For example, S (subject), V (verb), O (object), A (adverbial phrase), and so on. You might model the procedures already provided.

Extensions

This article was written in late June, so I thought an example using a summer theme and the parts of speech combined with figures of speech might be appropriate. Metaphor and simile are the more widely used figures of speech. We apply them frequently to enrich the rhetorical and literary aspects of our language. Simile compares two essentially unlike things and uses pointer words such as *like* or *as*. For example, cheeks like roses. A metaphor is a compressed simile. It uses no pointers. The program is written in LogoWriter for the Macintosh.

The command for this program is **poem**. To create a simile rather than a metaphor, write the fourth line in **poem** as:

```
(print [Summer is like a] pick noun)

to poem
(print [Summer is] pick adj)
(print [Summer is a] pick noun)
(print char 32 pick location)
print []
end

to adj
output [warm. humid. breezy. fragrant. lazy.
  easy.]
end

to noun
output [[swimming pool] picnic [thunder
  storm] hike trip vacation]
end

to location
output [[under a rainbow.] [in the woods.]
  [by a strawberry patch.] [beside a pool.]]
end
```

Here are some sample printouts:

```
Summer is humid.
Summer is a vacation
  under a rainbow.

Summer is warm.
Summer is a swimming pool
  by a strawberry patch.

Summer is breezy.
Summer is a swimming pool
  under a rainbow.

Summer is easy.
Summer is a vacation
  in the woods.
```

Some conclusions

In its minor way, this article is a celebration of language and the uses to which it is placed continually. Language has great range and complexity and is frequently able to reflect a beauty that is not only aesthetically pleasing but also emotionally stimulating.

Whether we are interested in a child's earliest babblings or captivated by a great actor's interpretative re-creation of a Shakespearean monologue, we all have a right to employ our mother tongue in these delights and to apply the skills we have gained in mastering that tongue to the investigation of others.

While we have touched only briefly on the surface of a vast discipline, perhaps the touching may prove to be the most valuable part of this brief essay. ▲

References

Bernstein, Leonard (1976). *The Unanswered Question. Six Talks at Harvard.* Cambridge, MA: Harvard University Press.

Chomsky, A. Noam (1957). *Syntactic Structures.* The Hague: Mouton.

Crystal, David (1987). *The Cambridge Encyclopedia of Language.* Cambridge, UK: Cambridge University Press.

Zettersten, A. (1968). A Statistical Study of the Graphic System of Present–Day American English. Lund: Studentlitteratur.

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

ISTE Alliances Presents

If You're a New Technology Leader, You *Need* to be Here.

The ISTE Academy for Technology Leadership is a 13-month committment including nine days of instruction during three sessions.

Learn to provide leadership related to:

- Technology to support student learning and the curriculum
- Effective professional development
- Long-range technology planning
- Networking trends

This innovative program includes:

- Instruction by nationally recognized practitioners
- Practical information and hands-on experiences
- Year-long collaboration with colleagues
- Support from ISTE staff

When & Where?

- August 13–16, 1997, Eugene, Oregon
- November 13, 1997, at the Tel•Ed '97 Conference, Austin, Texas
- Four-day session in Summer, 1998, in Eugene, Oregon

The Academy includes enrollment in all three sessions.

The ISTE ACADEMY *for* TECHNOLOGY LEADERSHIP

Sound interesting? *For more information and an application contact:*

Leslie Conery, ISTE Alliances
lconery@ccmail.uoregon.edu
541/346-2479
http://www.iste.org

Seeing Algebra: My Turtle Plots Graphs

by John Gough

I have recently spoken with several people who expressed indifference to Logo, LogoWriter, and MicroWorlds. These people were dealing with secondary mathematics. They *knew* that Logo was kids' stuff—only good for drawing houses and squares and boxes. They *knew* you couldn't use Logo to teach advanced mathematics or nongeometric mathematics.

What a challenge! How wrong could they be?

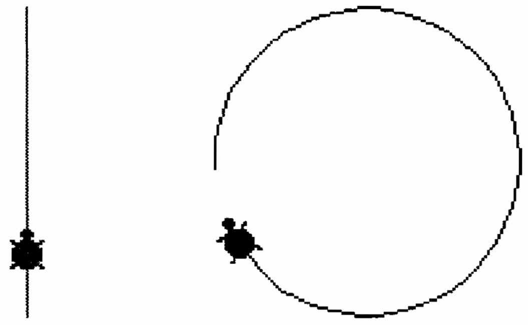
The collection of procedures below illustrates some fairly advanced ideas in applied mathematics, physics, and trigonometry. Of course, Logo is not inherently a graph plotter. Nor is it an oscilloscope (are such things still used, or am I betraying my Stone Age training?). However, a few simple lines of typing will give some fascinating pictures, and allow students to begin to look for patterns, and to respond to new challenges—exactly the kind of empowering for which Logo was invented.

Simple harmonic motion

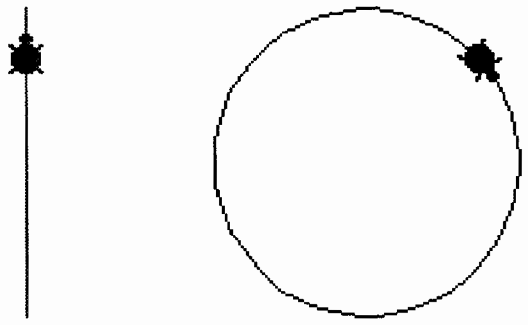
The first procedure makes use of LogoWriter's ability to have two turtles on the screen at one time.

```
to simple.harmonic.motion
set.up.turtles
repeat 100 [move.turtles]
end

to set.up.turtles
tell 0
pu setpos [20 0] pd
setc 3
tell 1
pu setpos [-50 0] pd
setc 5
st
end

to move.turtles
tell 0
forward 5
right 5
tell 1
sety ask 0 [ycor]
end
```

These procedures cause turtle 0 to move around in a circle.



Turtle 1 uses the current y-value of turtle 0 to slide up and down vertically while turtle 0 rotates around the circle. This is an elegant demonstration of the link between circular motion and simple harmonic motion.



Ssâng an angle variable instead of a turning turtle

This next example shows how an angle variable, rather than the physical turning of a turtle, can use sine and cosine to determine both circular and simple harmonic motion. Referring to a unit-circle diagram that defines the geometric concepts of sine and cosine can explain what the procedures are doing.

Simple harmonic motion is the explanatory heart of many common oscillating phenomena such as vibrating springs, clock pendula, ocean tides, the motion of a point on the edge of a rotating circle, alternating electric current, electromagnetic radiation, and so on.

```
to s.h.m :angle
set.up.turtles
repeat 100 [s.h.m.move :angle make "angle
  :angle + 10]
end

to s.h.m.move :angle
tell 0
setpos list 50*cos :angle 50*sin :angle
tell 1
sety 50*sin :angle
end
```

These procedures could be written recursively, but I personally think that the explicit "plain English" meaning of **repeat** is easier to follow than the procedurally implied meaning built into looping recursive commands. Readers who are familiar with recursion might like to develop recursive equivalents to the procedures that use **repeat**.

Compare the two sets of procedures above line by line, piece by piece, to see how they are the same and where they differ. To run **s.h.m :angle**, try any of the following:

```
s.h.m 0
s.h.m -720
s.h.m 90
```

If you don't have access to a version of Logo with multiple turtles, then one turtle can do the same work with the following procedures.

```
to s.h.m.one :angle
pu
repeat 100 [s.h.m.move.one :angle make
  "angle :angle + 10]
end

to s.h.m.move.one :angle
setpos list 50*cos :angle 50*sin :angle
dot
setpos list -20 50*sin :angle
dot
end

to dot
pd fd 0 pu
end
```

This makes the one Logo turtle hop in turn around the circle and across to the vertical sliding line, return to the circle and so on, each time drawing a dot of "zero" length. Incidentally, notice how **repeat** is used to limit the amount of work done. Can you change these procedures to use recursion instead?

[*Editor's Note:* The procedures in this article use the following procedure to make larger dots for clearer printed graphs.]

```
to dot
pu
repeat 6 [fd 1 rt 60]
pd
end
```

Interestingly, the dots appear triangular in shape.]

Trigonometric functions and more

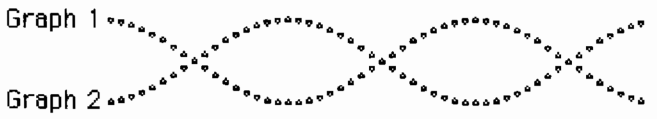
The next procedure lets the turtle plot the trigonometric function y = a sin (bx +c). You can choose values for the constant coefficients **a, b** and **c** when you run the procedure. The value of **x** runs from –230 to 230. This may not show the full periodicity of the function with the coefficients selected. But it is not hard to adjust this so that **x** goes from –360 to 360. You may wish to adjust the value of **x** so that it uses the horizontal width of your Logo graphics window.

```
to a.sin.bx.plus.c :a :b :c
make "x -230
pu
repeat 90 [sine.plot :a :b :c]
end

to sine.plot :a :b :c
setpos list :x 20*:a*sin ((:b*:x)+:c)
dot
make "x :x + 5
end

to a.cos.bx.plus.c :a :b :c
make "x -230
pu
repeat 90 [cos.plot :a :b :c]
end

to cos.plot :a :b :c
setpos list :x 20*:a*cos ((:b*:x)+:c)
dot
make "x :x + 5
end
```

The corresponding cosine procedures included here are simple substitutions of the sine procedures and use the LogoWriter primitive **cos** instead of **sin** in the obvious places.

These are nice to try:

```
rg
a.sin.bx.plus.c 1 2 0
a.cos.bx.plus.c 1 2 90
```



Also try:

```
rg
a.sin.bx.plus.c 1 2 0
a.cos.bx.plus.c 1 2 -90
```

Surprised? These last two instruction lines draw the same graph, even though one uses sine and the other cosine, and their third variable is different. In fact they demonstrate what is called the *phase relationship* between sine and cosine. Two wave patterns are said to be the same, but "out of phase" if we can move one along the horizontal axis and find that after a suitable shift along the axis the two waves exactly overlap. That's what is happening here. Why?

From two points of view, in trigonometry, we can see that sine and cosine are essentially the same idea, or at least very closely related. First, remember that the sine of angle A in a right-angled triangle is sometimes defined as the ratio of the opposite side divided by the hypotenuse. But the total angle sum of internal angles in any triangle is 180 degrees. This means that if B is the other angle in the same right-angled triangle, then B is 90 degrees – A. (Notice the simple arithmetic relationship between B, A, and 90 degrees.) The cosine of B is the same ratio as the sine of A.

(Try it: Draw a right-angled triangle, with angles A and B, and label the appropriate sides "adjacent," "opposite," and "hypotenuse" for angle A. Then use these labels to specify the correct ratio of sides for cosine of angle B. Then what is the sine of angle B? And cosine of A?)

Moreover, cosine also can be defined as the x-coordinate when we express sine and cosine in terms of a unit radius rotating around a unit circle. Also the value of any angle, say M, is measured as positive when the radius rotates in a counterclockwise direction around the unit circle. But surprisingly the x-coordinate or the cosine is the same in size and in positive or negative sign when we rotate the radius through a negative clockwise angle –M. This means that cosine of 90 degrees minus A is the same as the negative of (90 degrees minus A), which is the same as cosine of A minus 90 degrees. As we have already seen, cos (A – 90 degrees) = cos B = sin A. (If you're uncertain, draw a few simple unit circle diagrams to make sure of this.)

These LogoWriter functions show that as the input value for the angle **:x** is changed by subtracting 90 degrees, the cosine is the same as the sine for **:x** without any subtraction or phase shift. The third variable **:c** is the amount of the phase shift for each plotting function. Can you find a value of **:c** for **a.sin.bx.plus.c** so it is the same as **a.cos.bx.plus.c**? In other words, how much needs to be added to (or subtracted from) the sine graph so it exactly matches **a.cos.bx.plus.c** where **:c** is zero?

You can run several versions of these sine and cosine graphs and display them simultaneously. Use different colors to identify each one. Can you make procedures that will draw a set of scaled, marked axes?

Incidentally, if you try using values of **a** or **b** that are too large vertically, the graph will "wrap" around the screen vertically and look quite strange. The scale factor of 20 has been chosen to allow for small values of **a** and **b**, but you might like to try some larger ones to see what happens.

In the sine drawing above, we can slightly alter the command

```
setpos list :x 20*:a*sin ((:b*:x)+:c)
```

so that we don't have to worry about the vertical value getting too high or too low on the screen. Try these commands as a substitute:

```
make "y 20*:a*sin ((:b*:x)+:c)
if :y > 80 [make "y 80]
if :y < -80 [make "y -80]
setpos list :x :y
```

If the vertical y-value of the coordinate plotted by **setpos** is too high or too low, we will simply plot an upper (or lower) limiting value of 80 or (–80), instead of allowing the turtle to leave the screen and wrap around and confuse the resulting plotting of successive points.

Adding sine and cosine

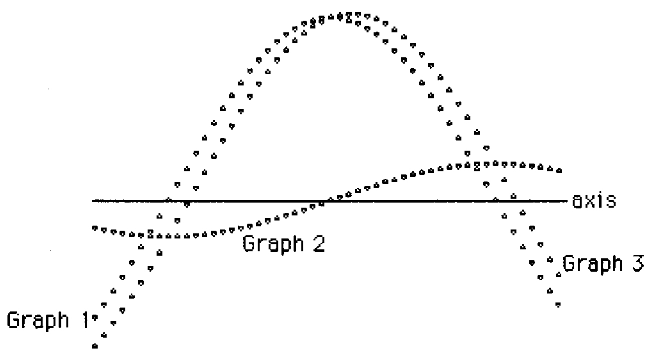
The next procedure uses the sine plotter and the cosine plotter to make far more complex graphs. In fact, this is the beginning of Fourier analysis, in which complex curves are analysed into constituent sine and cosine components. An alternative view is that such addition of separate sine and cosine pieces is equivalent to the acoustic theory of harmonics and overtones in music—the kind of thing that makes a piano sound different from a violin, even when both play a note of the same pitch.

```
to a.cos.mx.plus.b.sin.nx :a :b :m :n
setc 1
a.cos.bx.plus.c :a :m 0
setc 3
a.sin.bx.plus.c :b :n 0
make "x -230
setc 5
repeat 90 [cos.plus.sine :a :b :m :n]
end

to cos.plus.sine :a :b :m :n
setpos list :x (20*:a*cos :m*:x) +
  (20*:b*sin :n*:x)
dot
make "x :x+5
end
```

This procedure plots the cosine in one color and the sine in another color, and the sum of cosine and sine in a third color.

Try

```
a.cos.mx.plus.b.sin.nx 5 1 1 1
```



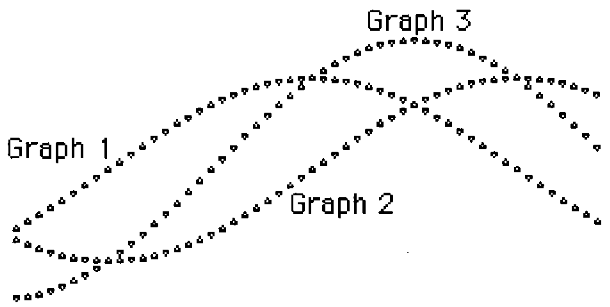
Look at the way the third graph is made out of combining the first graph, which swoops up and down, and the second graph, which moves slightly on either side of the "zero" line through the centre of the screen. You might like to incorporate a horizontal axis that shows where the vertical values are zero.

```
to zero.axis
pu setpos [-230 0]
pd setpos [230 0]
end
```

Sometimes during all of the point plotting of **a.cos.mx.plus.b.sin.nx 5 1 1 1**, the second graph, the sine curve, is small and positive; this either diminishes the larger negative value of the first cosine graph or increases the value of the cosine graph when it is above the zero line. You may need to think about the simple adding of two quantities, which are sometimes both positive, sometimes both negative, and sometimes of opposite sign (and large or small in absolute magnitude). Remember that the absolute magnitude of +3 is 3, and for –4 the absolute magnitude is 4.

Also try:

```
rg a.cos.mx.plus.b.sin.nx 2 2 1 1
```



Here the cosine and sine curves are actually the same shape and size but out of "phase" with each other, so that sometimes they are both positive, sometimes both negative, and sometimes of opposite sign. This sometimes results in the third graph being much higher than, much lower than, or in between the other two.

Another interesting possibility is

```
rg a.cos.mx.plus.b.sin.nx 2 2 5 .5
```

where the rapidly oscillating first graph (the cosine) is "carried" by the much slower second graph (the sine). This is the basic idea of "frequency modulation" used in FM radio.

You can make this even clearer if you use letters to indicate which graph is which. After each dot command in the **sine.plot** and **cos.plot** procedures you could add:

```
pd label "s pu
```

or

```
pd label "c pu
```

so you can see at a glance which curve is which. And similarly for the **cos.plus.sine** plotting procedure, add

```
pd label "+ pu
```

after the dot command.

A diversion: the physics of music

Why do this? Adding sine and cosine curves occurs all the time when we play musical notes and produce overtones or harmonics. For example, playing a C together with an E produces a mixed "note" or "chord" that the untrained ear hears as a whole sound and a musically trained ear distinguishes as two separate parts or "voices."

Moreover, the middle C (or any other note) played by a piano or a violin, for example, contains extra higher frequencies or vibrations that actually belong to other notes, such as E and G, and the C one octave higher. A nice way to show this is to use a piano. Silently press the sustaining pedal (so that all strings are free to vibrate) and strike the C above middle C hard, then release it. You should hear the C above middle C ringing softly, even though you didn't strike it.

That's because the overtones of middle C made the air vibrate, which made the string for C above middle C also vibrate. You can do this and experiment with other notes above middle C to find which vibrate sympathetically when middle C is struck.

In other words, the musical notes we are used to hearing do not consist of a single "pure" frequency (which we can think of as a simple sine curve); they have overtones produced by the addition of other higher notes known as harmonics or "partials."

These "impure" overtones (and related features about the way a note starts and ends) explain why we can hear the difference between a flute, violin, trumpet, or clarinet, for example, as they play the same note or frequency. In advanced mathematics this technique of matching a particular function by a sum of simpler functions is the fundamental idea of Fourier analysis, and is used in noise reduction and other modern techniques of acoustics such as wavelet theory.

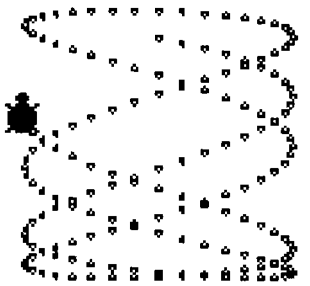
More trigonometric curves

Here is another complicated trigonometric graph. Instead of adding sines and cosines, we multiply them together and plot the product as the x-coordinate and the sine as the y-coordinate. Normally, such an interaction of two trigonometric functions on two axes can only be seen on a cathode ray oscilloscope, but in this case the willing turtle helps out.

```
to lissajoux :angle :factor
pu lissajoux.plot :angle :factor
lissajoux :angle + 5 :factor
end

to lissajoux.plot :angle :factor
setpos list (10*cos :angle)*(10*sin :angle)
  50*sin :angle/:factor
dot
end
```

Try the following:

```
lissajoux -720 3
lissajoux 0 1.5
lissajoux 180 1.414
```



This is a recursive procedure and can only be interrupted by using the Stop keys. What is the significance of the value 1.414? Try 3.141 also. Look for patterns with different values of **:factor**.

What happens if the two items listed as the values for **setpos** are put in reverse order?

What happens if we replace the product of cosine and sine by the product of sine and sine?

If this doesn't challenge secondary mathematics teachers to reconsider the potential of Logo for modeling advanced mathematics, and stimulating students with immediate feedback for algebraic manipulation and calculation... well, I'll just have to write another article all about using Logo and LogoWriter to simulate probability, statistics, chance events, and stochastic processes. And then an article about using Logo and LogoWriter to model set theory, logic, and propositional calculus. And then an article about number theory, primes, and residue classes. And then.... ▲

John Gough lectures in mathematics and computer education (LogoWriter and HyperCard especially) at Deakin University and is particularly interested in connections between mathematics and language. He can be reached at Deakin University SDS, 221 Burwood Hwy., Burwood, Victoria 3125, Australia. E-mail: jugh@deakin.edu.au

Logo: Search and Research
A Tale of Three Teachers, Part II

by Julie Sarama and Douglas H. Clements

In our last article, we started to tell the story of teachers using Turtle Math™ (Clements & Meredith, 1994) in their classrooms. That first article showed that we must do more than consider "factors" that affect implementation of an innovation, such as computer access, support, and time. We must also understand the different views and meanings that administrators, designers, and teachers hold for these factors. In this article, we look more closely at the views, meanings, and beliefs of the teachers. Their beliefs had a large impact on students' experiences.

Beliefs about teaching, technology, and mathematics

Ms. Jack was a confident, dedicated teacher. She was supportive of current reform efforts: "What I want to really accomplish in math is that they [students] are all mathematically literate. Beyond what does it mean to add, subtract, multiply or do long division, that they understand mathematics is involved in everything in every way possible."

Whenever we observed Ms. Jack teaching a noncomputer mathematics lesson, however, she was leading students through a page of the textbook, having them read the text and work through the problems. She was eager to supplement this work with different activities, but the activities did not fulfill her expressed desire that the mathematics come from the situation.

Ms. Moore also showed an interest in learning about educational reforms. She consistently sought out teacher education opportunities and consequently modified her teaching style. At the same time, she explicitly and unashamedly rejected certain innovations as impractical for her situation.

In certain aspects, we saw Ms. Moore as a mirror image of Ms. Jack. Ms. Jack professed beliefs and methods in line with a constructivist philosophy but taught using a more transmissionist approach. Ms. Moore espoused a traditional philosophy, although her teaching reflected a constructivist philosophy. For example, in her classroom (and while using Turtle Math) students consistently helped one another and did not rely on either her or the textbook as the source of knowledge. She allowed students time to experiment with the software. She also required them to write about and share their experiences.

The third teacher, Ms. Gaughan, viewed herself as limited by her math knowledge. When describing her lessons, she went through a page in the teacher's manual, showing a step-by-step adherence to the manual. "You show them the basic good steps or good rules to follow, and it's all paper and pencil at that point." Ms. Gaughan believed that discovery learning has a place in quality math teaching but she struggled with how to implement it.

All three teachers viewed mathematics as a list of topics to be covered. When Ms. Gaughan was asked to describe her goals in mathematics, she immediately referred to the textbook teacher's manual and started to list the topics from the index. In addition, Ms. Gaughan also communicated to her students that math is something to be suffered through and gave challenging math sheets as punishment.

A related set of beliefs involved the role of the computer in their teaching. Ms. Jack thought the software was powerful because students viewed it as a game. Ms. Moore and Ms. Gaughan also believed that the students liked Turtle Math because they saw it as "a type of game… like Sega or Nintendo." When Ms. Gaughan was asked about the best aspects of the program, she responded, "It feels to the kids almost like a game format, as very appealing. That's its strength and its weakness probably."

The teachers did not view the programming aspect of Turtle Math as a creative process. They wanted to focus on the end product and provide rewards for getting there. This contrast with our beliefs about Logo was highlighted in an incident described next.

Teachers' and designers' beliefs

The teachers felt that their students needed affirmation when they did anything on the computers. Because most Turtle Math activities do not provide such feedback, the teachers felt called upon to give it individually to each student. Teachers agreed that it is "nice" when students could make judgments for themselves, but they decided that fifth graders are too immature or accustomed to video games to be content without a "checker" of some sort. Ms. Moore stated, "They need that immediate feedback, and if they don't get it, they start to ask, 'Is this right?'" Ms. Jack added, "The children feel that that [they] 'got it' and now [they] can move on to something else instead of standing there jumping up and down saying, '[Ms. Jack], come look, come look!'"

The teachers suggested that the designers reprogram the software and include a built-in "checker" that would tell the students whether they had completed the required drawing task, such as making an equilateral triangle. The designer running the session explained why this was not plausible or desirable. She said, "You want them to tell you.... If the computer always did that, students would not need to justify or examine it."

Later, the teachers were asked if they agreed with this point of view. Ms. Gaughan stated, "There are times when the teacher needs to say, this is the truth in the sense that these are the facts. When you have an equilateral triangle, you either don't or you do." Ms. Jack added, "My preference would obviously be that they take down the measurement tool and check it themselves.... But, that's not going to happen for everybody."

The designers initially believed that they would provide software that could help teachers attain their educational goals and work collaboratively to design activities. However, teachers' goals were not the same as designers' (or administrators') goals for the teachers. This accounted, in part, for the decided lack of influence the designers had on teachers' use of the software. The teachers viewed the philosophy that underlies the software and the recommendations for mathematics reform as an ideal but impractical one.

The plot thickens: students' beliefs

Themes emerged from the interviews indicating that the three teachers' beliefs influenced those of their students. However, the students were also able to see benefits in the software not reported by the teachers. First, they described the benefit of solving problems in context: "You weren't just multiplying numbers. You were multiplying numbers to find the area or degrees." Second, although the majority of the students mentioned angles and shapes as examples of what they learned using Turtle Math, they also mentioned arithmetic quite frequently: "[It] would help us a lot, like in adding subtracting, multiplying and divide and learning shapes and 90-degree angles and other angles."

Third, when the students were asked about their general impressions of the software, they described it as fun, but they did not mention any gamelike features. When they were asked what they would do if they could do whatever they wanted in Turtle Math, several students mentioned the games but also talked about exploring the tools.

Researcher: "What if you had to do only Turtle Math during free time?"

Student: "I would do turtle things like make shapes and stuff because that's fun... and measure them and stuff."

We interpret these statements to mean that, contrary to their teachers' concerns, the students did not view the software as a game in and of itself but saw the games as one part of the software.

"The End"

A deficit of mathematics knowledge hindered teachers' implementation of the innovation. The teacher who lacked both confidence and ability in mathematics was unable to bring the innovation and her everyday ordeal of mathematics together, even with considerable assistance. However, even extensive knowledge of pedagogy did not ensure that teachers exhibited decisions and practices consistent with the innovation's philosophy.

For two teachers, Ms. Jack and Ms. Moore, we observed specific contradictions between their stated beliefs and observed practices. Such contradictions are not surprising, but they do have additional relevance to the adoption of an innovation. For one, the district administrators thought that these teachers were ready to reform their practice of mathematics teaching.

Indeed, the teachers also thought themselves ready. They did not see their own ideas and behaviors as conflicting. But in our roles in curriculum and software development and staff development, we saw contradictions that they did not. This situation hampered communication, support, and the implementation in general.

Teachers' beliefs about what their students wanted and needed educationally influenced their curricular choices. In some cases, these diverged from what the students said they wanted and from what, in our opinion, they needed.

In summary, the designers and teachers did not share a vision or image of mathematics education. The designers possessed a constructivist-oriented vision in which goals are mathematical power. The teachers' image of mathematics included components of this vision, although teachers shared an implicit belief that to achieve their declared mathematical goals, they had to "finish" the textbook—even given that these goals did not come from the textbook and were inconsistent with it. We believe that such implicit mixed beliefs denied the teachers a clear vision for reform. The teachers did not possess an internal guidance system for innovative approaches to mathematics education.

The effects of individual beliefs and differences were so profound that we began to question current approaches to "systemic reform." We need to consider not only the system, but also the many individual perspectives. Teachers must construct their own understandings of content, such as mathematics, computing, and innovations, and what all these mean for teaching and learning in their classrooms. They need to connect new ideas and approaches to existing ideas about teaching and learning. ▲

Time to prepare this material was partially provided by a National Science Foundation (NSF) research grant, "An Investigation of the Development of Elementary Children's Geometric Thinking in Computer and Noncomputer Environments" (NSF MDR-8954664). 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 SUNY Buffalo, has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. Through a NSF grant, he developed a K–6 curriculum, *Logo Geometry* (published by Silver Burdett, & Ginn, 1991). With colleagues, he is working on the previously mentioned NSF research grant and is finishing a second NSF-funded project, "Investigations in Number, Data, and Space," to develop a full K–5 mathematics curriculum featuring Logo. With Sarama, he is co-authoring new versions of Logo for learning elementary mathematics.

Julie Sarama, Ph.D., is an assistant professor at Wayne State University. She previously taught secondary mathematics and computer science, gifted math at the middle school level, and mathematics methods courses. She is coauthor of several Investigations units and of Turtle Math and is currently designing and programming new versions of Logo and other computer microworlds.

Douglas H. Clements
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall
Buffalo, NY 14260
E-mail: Clements@acsu.buffalo.edu

Julie Sarama
Wayne State University, Teacher Education Division
Detroit, MI 48202
E-mail: JSarama@coe.wayne.edu

References

Clements, D. H., & Meredith, J. S. (1994). Turtle math [Computer program]. Montreal, Quebec: Logo Computer Systems, Inc. (LCSI).

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 Resource Guide 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*
• *The Best Math & Science Web Sites for Teachers*	17.05	18.95
• *Logo Musings—Ten Mathematical Encoutners Using LogoWriter*	19.75	21.95
• *Introduction to MicroWorlds 2.0—A Logo-Based Hypermedia Environment*	25.15	27.95

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 $4.50
$16-$45.99 (subtotal)	$6.00
$46-$75.99 (subtotal)	$7.00
$76-$100.99 (subtotal)	$8.00
$101 or more	8% of subtotal

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 of the same title	-
SUBTOTAL	
*Shipping and Handling (see box above)	+
*Add Additional 7% 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 $4.75	+
TOTAL	=

* If actual shipping cost exceeds this amount, we will bill you for the difference.

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 $4.75 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 97401-2626 USA • Order Desk 800/336-5191 • Fax 541/302-3778

h t t p : / / W W W . I S T E . O R G

Prepare for Your Tel•Ed Migration this fall.

ISTE's Sixth International Conference on Telecommunications and Multimedia in Education is migrating to Austin, Texas, and Mexico City, Mexico, in November — and as an education stakeholder of the '90s who understands the dramatic, positive changes interactive technologies have made in the classroom, you should be too.

Exponential growth on the Internet. Transformations in the use of the World Wide Web. These developments have made a revolutionary impact in education over the past decade — literally opening up the classroom walls to new collaborations and innovative ways of communication.

ISTE's Tel•Ed has been at the forefront of this educational metamorphosis since 1989, providing practical solutions to better integrate multimedia and telecommunications into our diverse learning environments. ISTE's multisite Tel•Ed conference in 1997 will offer hands-on workshops, classroom-tested sessions, professional networking, a cutting-edge trade show and much more, inspiring you to take your students of today into the learning opportunities of tomorrow.

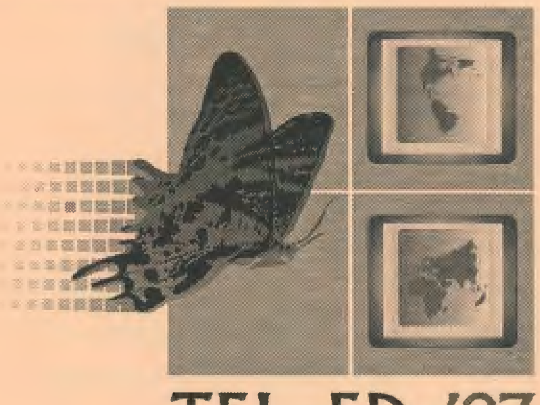
So don't let the Tel•Ed migration pass overhead. Spread your wings and Take Flight in the Digital Age.

Austin, Texas, USA *Mexico City, Mexico*
November 13–16, 1997

To receive your FREE Advance Program
or for more Tel•Ed information, please contact ISTE:

Phone: 541/346-2472 • **Fax:** 541/346-5890
WWW: http://www.iste.org
Internet: laurie_thornley@ccmail.uoregon.edu

Presented By:
International Society for Technology in Education (ISTE)
Texas Computer Educators Association (TCEA)
Instituto Latinoamericano de la Comunicación Educativa (ILCE)



TEL•ED '97
Taking Flight in the Digital Age