Logo *Update*

The Logo Foundation Newsletter ▲ Volume 3, Number 3 – Spring 1995

In This Issue

Most of this issue of *Logo Update* is about the happy marriage between LEGO and Logo. The coincidental similarity between the two words conveys a deeper synergy. Both encourage and enable us to design and build complex systems out of small simple pieces. The combination of the two enriches both.

LEGO/Logo has come to be used in thousands of schools over the past ten years. Beginning on page 6, Marian Rosen gives us a glimpse of what is taking place in her school.

In "Building and Learning with Programmable Bricks" we get a look at the current LEGO/Logo research which will very likely develop into the classroom practice of the near future.

LEGO/Logo is a general term that refers to a range of specific products. There are also robotic systems that combine Logo with other construction kits. A brief summary of what is available may be found on page 8.

Beginning on page 10, Carol Sperry reviews Herbert Kohl's *I Won't Learn From You*, a book which has nothing and everything to do with Logo.

This issue of *Logo Update* is also full of information about workshops, Logo software, and other resources. Most of the products and services that are announced and advertised in these pages may be ordered from the Logo Foundation. An order form is on page 15.

Finally, if you feel that *Logo Update* is a valuable source of information and ideas please help keep it alive by making a contribution to the Logo Foundation. We want to continue to provide free

continues on the next page

Building and Learning With Programmable Bricks

by Randy Sargent, Mitchel Resnick, Fred Martin, and Brian Silverman

In many educational computer projects, children control and manipulate worlds in the computer: for example, turtle graphics and animation projects done with Logo.

But instead of controlling and manipulating worlds in the computer, what if children could control and manipulate computers in the world? That is, what if children could spread computation throughout their own personal worlds? For example, a child might attach a tiny computer to a door, then program the computer to make lights turn on automatically whenever anyone enters the room. Or the child might program the computer to greet people as they enter the room – or to sound an alarm if anyone enters the room at night.

In this article, we describe a new technology, called the Programmable Brick, that makes such activities possible, and we explore how this new technology might open new learning opportunities for children. The Programmable Brick is a tiny, portable computer embedded inside a LEGO brick, about the size of a deck of cards. The brick is capable of interacting with the physical world in a large variety of ways (including via sensors and infrared communication). Our hope is that the Programmable Brick will make possible a wide range of new activities for children, encouraging children to see themselves as designers and inventors. At the same time, we believe that these activities could fundamentally change how children think about computers and computational ideas.

LEGO/Logo

The Programmable Brick project extends our previous work with LEGO/Logo (Resnick, Ocko, & Papert, 1988; Resnick, 1993). LEGO/Logo links the popular LEGO construction kit with the Logo programming language. In using LEGO/Logo, children start by building machines out of LEGO pieces, using not only the traditional LEGO building bricks but newer pieces like gears, motors, and sensors. Then they connect their machines to a computer and write programs, using a version of Logo, to control the machines. For example, a child might build a LEGO house with lights, and program the lights to turn on and off at particular times. Then, the child might build a garage, and program the garage door to open whenever a car approached.

In the early years of Logo, its most popular use involved a "floor turtle," a simple mechanical robot connected to the computer by a long "umbilical cord." With the proliferation of personal computers in the late 1970's, the Logo community shifted its focus to "screen turtles." Screen turtles are much faster and more accurate than floor turtles, and thus allow children to create and investigate more complex geometric effects.

In some ways, LEGO/Logo might seem like a throwback to the past, since it brings the turtle off the screen and back into the world. But LEGO/Logo differs from the early Logo floor turtles in several important ways. First of all, LEGO/Logo users are not given ready-made mechanical objects; they build their own machines before programming them. Second, children are not restricted to turtles. Elementary-school students have used LEGO/Logo to build and program a wide assortment of creative machines, including a programmable pop-up toaster, a "chocolate-carob factory" (inspired by the Willy Wonka children's stories), and a machine that sorts LEGO bricks according to their lengths.

continues on the next page

Logo Foundation
250 West 57th Street
New York, NY 10107-2228
Telephone: 212 765-4918
FAX: 212 765-4789
email: michaelt@media.mit.edu

The Logo Foundation is a nonprofit educational organization incorporated in New York State.

Logo *Update* is published three times yearly by the Logo Foundation. Subscription is free.

In This Issue

continued from page 1

subscriptions. But printing and mailing costs are rising and our circulation has increased by more than 25% over the past year. Advertising revenue covers only part of the cost of publication. The Logo Foundation is a nonprofit organization which relies upon contributions as an important source of income. Please consider making a tax-deductible donation of $25 or more to help us continue to provide service to the Logo community. Thank you for your support!

Michael Tempel

Building and Learning with Programmable Bricks

continued from page 1

LEGO/Logo has some limitations. For one thing, LEGO/Logo machines must be connected to a desktop computer with wires. Wires are a practical nuisance, particularly when children use LEGO/Logo to create mobile "creatures." They get tangled with other objects in the environment, they get twisted in knots as the creature rotates, and they restrict the overall range of the creature. Wires are also a conceptual nuisance. It is difficult to think of a LEGO/Logo machine as an autonomous creature as long as it is attached by an umbilical cord to a computer.

Members of our research group have tried to solve these problems in several ways. We experimented with various technologies for wireless communication to get around the problem of wires. But none of these approaches satisfied us. So we decided to make a more serious modification: we began to build electronics *inside* the LEGO bricks. We have taken several approaches. The "Braitenberg Brick" system, developed primarily by Fred Martin with inspiration from the book *Vehicles* (Braitenberg, 1984), is based on a set of low-level "logic bricks" (such as and-gates, flip-flops, and timers). Students can create different behaviors by wiring these bricks together in different ways (Granott, 1990; Hogg, Martin, & Resnick, 1991).

The Braitenberg Bricks have dedicated functions. The flip-flop brick, for instance, has a very specialized function: It holds one bit of state, and it changes that state whenever it receives a sharp transition in its input. But why should we be restricted to dedicated bricks? Why not put a full computer into a LEGO brick? That is what we have done in the Programmable Brick project.

Designing the Programmable Brick

In designing the Programmable Brick, we had several overarching goals. Each goal involved some type of "multiplicity":

Multiple Activities

We wanted the Programmable Brick to support a wide variety of different activities – so that it could connect to the interests and experiences of a wide variety of people. While some people might use the Brick to create their own scientific instruments, others might use it to create their own musical instruments.

Multiple Input and Output Modalities

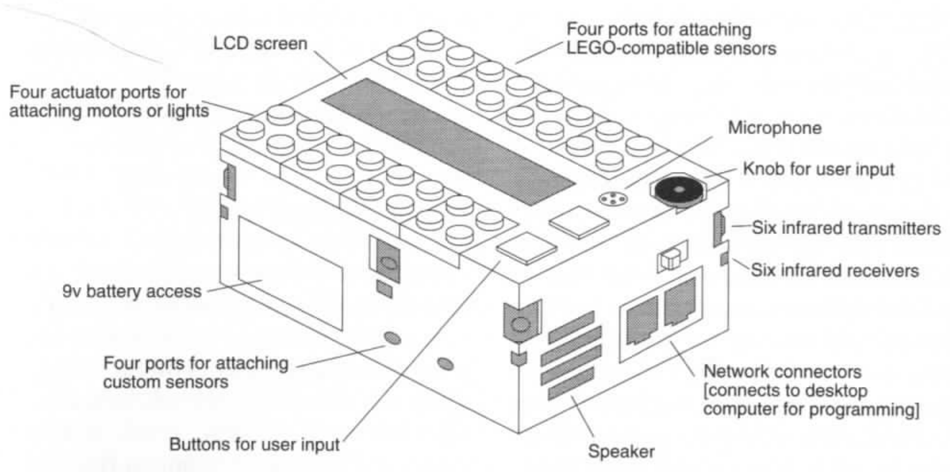
Because we wanted the Programmable Brick to connect to many things in the world, the Brick needed many different types of output devices (such as motors, lights, beepers, infrared transmitters) and input devices (such as touch sensors, sound sensors, light sensors, temperature sensors, infrared receivers.) Indeed, the number of possible applications of the Brick expands greatly with each new input or output device, since each new device can be used in combination with all of the others.

Multiple Processes

Children working on LEGO/Logo projects often want to control two or more things at the same time. For example, they might want to make a Ferris wheel and merry-go-round turn in synchrony, while a song plays in the background and an electric eye automatically counts the rotations of the rides. With standard programming languages, it is very difficult to achieve this effect: The user must explicitly interleave the multiple threads of control. In the Programmable Brick, we wanted to support parallel processing, so that users could easily write programs to control multiple outputs and check multiple sensors all at the same time.

Multiple Bricks

We wanted Programmable Bricks not only to act on their own but to interact with one another. In that way, children could program Bricks to share sensor data with each other, or they could create "colonies" of interacting creatures. These types of activities would enable children to explore the scientific ideas of emergence and self-organization (Resnick, 1994).


LCD screen
Four ports for attaching LEGO-compatible sensors
Four actuator ports for attaching motors or lights
Microphone
Knob for user input
Six infrared transmitters
Six infrared receivers
9v battery access
Four ports for attaching custom sensors
Network connectors [connects to desktop computer for programming]
Buttons for user input
Speaker

Based on these goals, we developed the Programmable Brick shown above. About the size of a deck of cards, the Programmable Brick is based on the Motorola 6811 processor with 256K of non-volatile RAM memory, and it has a wide variety of input-output possibilities. The brick can control four motors or lights at a time, and it can receive inputs from eight sensors. A speaker and microphone are built into the Brick for sampling and emitting sounds. Around the sides of the Brick are six infrared transmitters and receivers, so that the it can communicate with other Programmable Bricks (and other electronic devices). On top is a two-line liquid-crystal display, plus a knob and two buttons for interacting directly with the Brick.

To program the Programmable Brick, you first write programs on a standard personal computer, then download the programs via a cable to the Brick. You can now disconnect the cable and take the Brick with you. The programs remain stored on the Brick. When you want to execute a program on the Brick, you can scroll through a menu of programs on the two-line liquid crystal display screen (using the knob to scroll), then press a button to run the selected program.

Brick Logo, the version of Logo developed by Brian Silverman for the Programmable Brick, has some special features not found in previous versions of LEGO/Logo. Users may write multiple "condition-action" rules that connect sensor stimuli to behavioral reactions. For example, short pieces of Logo code can tell a robot to back up and turn left when the right-hand touch sensor is pressed, and back up and turn right when the left-hand touch sensor is pressed. While these responses are active, one can add rules telling the robot how to respond to a light sensor. With these condition-action rules, it's easy for learners to develop complex behaviors for their robots, while broadening their thinking about the meaning of sensors and programming.

Experiences with the Brick

An Active Environment

One of the earliest projects with the Programmable Brick involved two kids named Andrew and Dennis, aged 11 and 12. They were intrigued with the idea of making an "active environment" – making the environment "come alive" and react to people. After some consideration, they decided to make a device to flip on a room's light switch when people entered the room, and flip it off when people left.

Andrew and Dennis decided to try a "bend" sensor to sense the opening of the door. (The bend sensor is a plastic whisker, several inches long, that gives a measure of how much it is bent.) They first tried to mount the sensor to the wall approximately where the doorstop was, but then decided that people would need to open the door very wide before the sensor detected anything. Then, they tried mounting the sensor at the door hinge in such a way that the sensor was bent in proportion to how wide the door was opened.

Before programming, Andrew and Dennis tested the value of the bend sensor at different positions of the door, to find out if they had mounted the sensor well and if the sensor would really give them the information they wanted. Then, they built a LEGO mechanism to flip the light switch on the wall of the room. The mechanism connected a motor, through a gear train, to a lever that pushed against the light switch. They designed their mechanism in such a way that spinning the motor one way would turn the light on, while the reverse direction would turn the light off.

At this point, Andrew and Dennis started focusing on the algorithm for flipping the light switch when the door opened. They realized there was a problem: The door sensor indicated when the door was opened, but it did not tell whether people were entering or exiting the room. They wanted some sort of sensor to tell whether someone was entering the room (in which case their machine should turn on the light) or leaving the room (in which case the machine should turn off the light).

After a little thinking, Andrew and Dennis came up with a clever solution: They attached a LEGO bar to the door handle on one side of the door, and connected a LEGO touch sensor to this bar so that the sensor was activated when someone grasped the door handle. In this way, the Programmable Brick could tell if people were leaving (in which case the door would be opened and the touch sensor in the handle pressed), or if people were entering (the door would open without a signal from the touch switch).

Once the second sensor was in place and tested, they wrote their program:

```
to light
if (sensor-a < 105)
   and touch-b
   [turn-off-light]
if (sensor-a < 105)
   and not touch-b
   [turn-on-light]
end
```

Run motor forward for 3 seconds:

```
to turn-off-light
motor-a, this-way
onfor 30
end
```

```
Run motor backward for 3
seconds:
to turn-on-light
motor-a, that-way
onfor 30
end
```

Once Andrew and Dennis got the project working , they ran in and out of the room repeatedly, breaking into big smiles each time the lights switched on and off.

Artificial Creatures

We conducted a four-day workshop at the Boston Museum of Science in which students used Programmable Bricks to create "artificial creatures." The five participants, ages 12 to 16, had three hours of workshop time per day, for a total of 12 hours. Although students had varying amounts of previous experience with LEGO and programming, all were able to make a working programmable "creature" by the end of the workshop.

One focus of this workshop was the use of multiple processes for multiple behaviors. With the Programmable Brick, different simple programs (such as "follow light" or "follow wall") can be run as separate processes. Users can turn these individual programs on and off using the Brick's screen, knob, and buttons. The Brick's software also includes primitives that allow students to turn on and off the different processes under program control. Thus, processes have the ability to start or stop other processes.

Three participants who had not programmed before wrote fairly simple programs for their creatures. One made a creature that followed a line (a piece of tape laid down on the floor). Another made a creature that simply backed up when it hit an obstacle. The third made a creature that backed away from bright light, and had lots of fun playing with his creature using a flashlight.

Mark, who had programmed before, started his project with a single, simple behavior. The creature tried to navigate a path between several rooms using timing only (no sensor feedback): go forward for 20 seconds, turn left, go forward for 15 seconds, turn left, go forward for 20 seconds. This path was intended to make the creature leave the classroom, go down the hall, make a left into a different classroom, and turn left again to try to get out the other classroom's back door. But this simple behavior didn't have much of a chance of working. The second classroom was full of tables and chairs, and the creature invariably hit one or two and got stuck. Sometimes the timing of the path was a little off, or the creature drifted off its planned path, and ran into a wall unexpectedly. For this creature, running into a wall, chair, or table typically meant getting stuck and progressing no further.

To try to deal with this problem, Mark added a second behavior: When the creature ran into an obstacle, it attempted to pilot around the obstacle and end up with roughly the same heading as it had in the first place. The behavior looked like this:

```
to avoid-obstacles
if touch-a
    [spin-right
    wait 10
    go-forward
    wait 10
    spin-left
    wait 10
    go-forward]
end
```

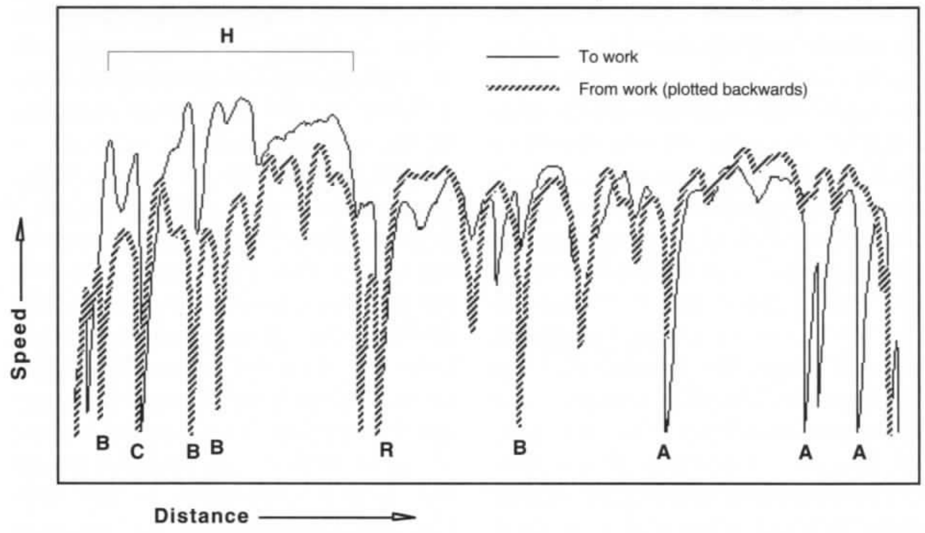
The added behavior did not completely solve the navigation problem – the creature did not navigate its course reliably. However, it typically got much further with this new behavior than without it.

Personal Programming

Brian Silverman and his son Eric mounted a Programmable Brick on Brian's bicycle along with a magnetic sensor that recorded each revolution of the front wheel. This information, along with the Brick's built-in timer, allowed them to keep track of the bike's speed at each point in Brian's daily commute to and from work. The graph below shows one day's commute with the evening trip plotted in reverse on the same graph as the morning trip. Brian slowed down for railroad tracks both coming and going. But an area of high speed in one direction and low speed the other way indicates a hill. Occasionally the speed drops to 0 at a red light.

Out into the World

During the summer of 1994 a one-week Programmable Brick workshop was held for teachers in the "Rhode Island School of the Future." a project led by Janice Kowalczyk. This workshop, led by Fred Martin with Wanda Gleason acting as assistant, was the first occasion on which a group of teachers had used the Programmable Brick technology outside of the MIT laboratory. The twenty participants



Points of interest
A: Traffic light (red going to work, green coming home)
B: Traffic light (green going to work, red coming home)
C: Traffic light (red both directions)
H: Hill (downhill towards work, uphill back)
R: Railroad tracks

developed small LEGO robots, controlled by Programmable Bricks, to solve "follow the line" and "get out of the box" challenges.

In the fall we brought the Brick into classrooms. Some of us were uncomfortable with the idea of requiring all students to participate in a competition, as had been done in the summer workshop, because it would impose a kind of performance pressure that might not be desirable. We developed the concept of a "Robotic Park" exhibit in which kids would create robotic LEGO "animals" that could either be noncompetitive show pieces or participate in a simple, fairly standardized performance event.

Work in Mariann Hayward's fourth grade class and John Bilotta's fifth and sixth grade class began in November of 1994. Students worked in teams of three to four and chose animals that they studied and adapted them to be implemented as programmed LEGO constructions. In Mariann's class, students are building a robotic crab, turtle, and alligator; and in John's class, students are building LEGO fish and a dinosaur along with some other LEGO projects that aren't "animals." In all cases, students are developing behaviors in their LEGO constructions that mimic the activity of the actual animals or artifacts. The LEGO crab has a pair of pincers that start snapping when the crab runs into something; the LEGO turtle's head retracts when its nose is bumped; and the LEGO dinosaur is attracted to flashes of light (like the dinosaur in Jurassic Park).

Work in Lee Rabbitt's high school class began in February of 1995. Since they knew there would be only a short time to prepare for the Robotic Park event, students elected to focus specifically on the capabilities needed to solve the performance challenge: obstacle avoidance and light-seeking. After early attempts in which the students wanted to build machines to climb over the obstacles rather than getting around them, students settled on car-like vehicles (adapted from LEGO's wheelchair design in the Control Lab kits) and the use of touch and light sensors. Being teenagers, most of them were quite happy redefining the activity to be building cars rather than animals, and are engaging in the programming aspect of the activity as readily as they did the building.

All of these Programmable Brick activities reap the educational value of "traditional" LEGO/Logo and then some. When working on the LEGO building, students learn about structures, mechanisms, and the process of designing – how to be inventors, share ideas, and adapt ideas from existing plans. When working on programming, students learn how to conceptualize and formally express an algorithm to make a mechanical object accomplish some task. Throughout the project, students work cooperatively to solve problems, collaborating and sharing ideas.

As this article goes to press, we are in the final stages of preparation for the Robotic Park exhibition which will be held on April 29, 1995 at the Peace Dale Elementary School. There will be a series of talks and workshops for teachers in the morning, with the exhibition of the children's work in the afternoon. A follow-up workshop for teachers will be held during the summer of 1995.

What's Next?

At the present time there exist 50 Programmable Bricks which are being used in the projects and activities we have described in this article. The pattern of diffusion has been similar to that of the original LEGO/Logo technology a decade ago and of Logo itself before that. It begins with a period of development in the lab at MIT and then in a few closely supported projects in schools and other settings. We expect that the Programmable Brick will eventually become available to the educational community at large.▲

Acknowledgments

Seymour Papert, Steve Ocko, and Allan Toft have provided encouragement, inspiration, and ideas for the Programmable Brick project. Andrew Blumberg, Fei Hai Chua, Chris Gatta, Hanna Jang, Owen Johnson, Victor Tsou, and Elaine Yang contributed to the development effort. The LEGO Group and the National Science Foundation (Grants 9153719-MDR, 8751190-MDR, and RED-9358519) have provided financial support.

References

Braitenberg, V. *Vehicles,* MIT Press, Cambridge, MA, 1984

Epistemology and Learning Group *Programmable Brick Handbook,* MIT Media Lab, Cambridge, MA, 1995

Granott, N. "Puzzled Minds and Weird Creatures: Spontaneous Inquiry and Phases in Knowledge Construction" in I. Harel and S. Papert (Eds.), *Constructionism,* Ablex, Norwood, NJ, 1991

Hogg, D., Martin, F., & Resnick, M. *Braitenberg Creatures*, Epistemology and Leaning Memo #13, MIT Media Lab, Cambridge, MA, 1991

Martin, F., *Circuits to Control: Learning Engineering by Designing Lego Robots*, PhD Thesis, MIT, 1994

Resnick, M., Ocko, S., and Papert, S. LEGO, "Logo, and Design" in *Children's Environments Quarterly*, 5 (4) pp 14-18, 1988

Resnick, M. "Behavior Construction Kits" in *Communications of the ACM*, 36 (7) pp 64-71, 1993

Resnick, M. *Turtles, Termites, and Traffic Jams,* MIT Press, Cambridge, MA, 1994

Sargent, R., *The Lego Programmable Brick: Ubiquitous Computing for Kids,* Masters Thesis, MIT, 1995

Randy Sargent recently completed his Masters Thesis at MIT
Mitchel Resnick is Professor of Media Arts & Sciences
Fred Martin is a Post Doctoral Researcher
Brian Silverman is a Visiting Scientist and is also Director of Research at Logo Computer Systems, Inc.

The authors may be contacted at:
MIT Media Lab
20 Ames Street, E15-315
Cambridge, MA 02139
rsargent@media.mit.edu
mres@media.mit.edu
fredm@media.mit.edu
bss@media.mit.edu

This article is adapted from a chapter which will appear in the forthcoming book *Constructionism in Practice*, edited by Yasmin Kafai and Mitchel Resnick, to be published later this year by Lawrence Erlbaum Associates.

LEGO/Logo

by Marian Rosen

At Conway School, each fifth grader gets a four week LEGO/Logo unit taught by the classroom teacher and myself. The site is the classroom with five computers and "The Cart" stuffed with sorted LEGO pieces, idea cards from many different Technic series kits, video tapes of past projects, string, rulers, tape, scales, etc. "Simple Machines" is part of the fifth grade science curriculum. However, a lot more than simple machines gets taught. The crucial idea that underlies LEGO, Logo, and simple machines is that very complex objects can be made by combining simple things.

The number of students who have worked with gears, pulleys, and motors is much smaller than the number who have built things with bricks, plates, and studs. Yet, most of the kids understand how LEGO works. They know that the pieces can be combined and recombined into hundreds of different patterns. They know that some patterns are stronger, some prettier, some more creative, some more functional. They build complex machines made up of the simple machines they have studied.

Pairs of kids pick their level of comfort when using LEGO. Some choose to build exactly what is on an idea card, some start with a card and then modify it, some build totally from their imaginations. For example, during our last session, we had a drill copied from a card, an exercise machine based on a conveyor belt, and an original car wash complete with rotating wheel washers, overhead to and fro rag rack, and drying fans. Others accept a challenge such as building a machine that will lift or drag more than 30 pounds, or one that can balance on a single wire stretched across the room.

The programming part of LEGO/Logo makes the machines more interesting than they would be if they were run just by battery packs. We control speeds making blenders that can mix, mash, and puree (Cheerios and water smell awful!), or vehicles that accelerate and decelerate. We control the direction of the motors, making washing machines that both agitate and spin; cars that move in four directions; and cranes and toasters that go up and down. We use the sensors to make smart machines.

A car equipped with front and rear sensors bounces back and forth in a game played by two students armed with ping pong paddles.

A conveyor belt measures the length of pieces and then dumps them on a second conveyor belt that sorts longer pieces to the right and shorter ones to the left.

A conveyor belt is balanced on a 360 degree rotating machine which is balanced on top of a car. The car goes back and forth, the rotator spins to different headings, and the conveyor belt sorts pieces into six different piles.

A gondola moves along a wire stopping to lower and raise a magnet that picks up paper clips (representing stranded skiers) from the carpet (snow) below.

We use touch sensors to make a joy stick that has four settings and can be used to run many different machines.

Our fifth graders know Logo. They

A LEGO/Logo Project

This project is a 360 degree rotator with a pincher on top. Below is the program to control the rotator. Notice that **rotate.right** and **rotate.left** do not stop! The kids understood how to write a procedure with **onfor** that would move in small steps, but they didn't like it because it was jerky. It was more fun to let the rotator sweep to the left and right because the game was to **close.jaws** on a little LEGO person at exactly the right time, so that they nabbed it on the run and then dropped it off using **open.jaws** at exactly the right time to drop it in a bucket.

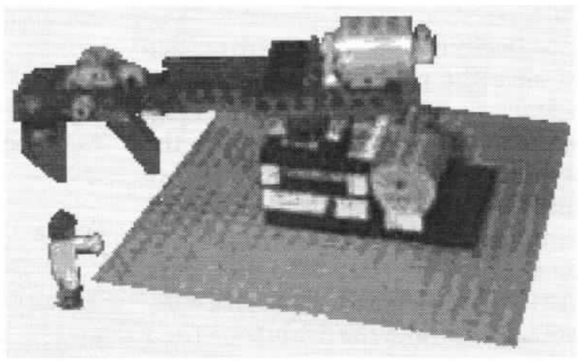
```
to control
lto [6 7]
if sensor? = [true false] [close.jaws]
if sensor? = [false true] [open.jaws]
if sensor? = [true true] [rotate.right]
if sensor? = [false fasle] [rotate.left]
if key? [ao stop]
control
end

to rotate.left
tto [b]
setpower 4 setodd on
end

to rotate.right
tto [b]
setpower 4 seteven on
end

to close.jaws
tto [a]
setpower 7 seteven
onfor 20
end

to open.jaws
tto [a]
setpower 7 setodd
onfor 20
end
```



have used it in multiple ways since kindergarten. They know that a program is a complex object made up of sub-procedures which are made up of primitives. They know it, but they don't necessarily think it's very useful. Sometimes a kid asks the equivalent of "Do you want me to have procedures called **right.leg** and **left.leg** or can I just call them both **legs**?" This annoys me because it shows (1) they aren't clear on where to "chunk" their work, (2) don't think it's important, and (3) are doing it just to please me. Worst of all I suspect the kids are correct and that it doesn't make a whole lot of difference how they chunk their programs. But LEGO/Logo projects are different. They give kids good reasons for writing small procedures that are meaningful and are a wonderful model for learning this important skill.

When students first program their machines, they make spaghetti programs that turn the machine on, wait a while, reverse the direction, wait a while, accelerate, wait a while, decelerate, wait a while, etc. There's excitement, but if anything goes wrong, debugging is very difficult. Small programs such as **car.on, car.off, rotate.left, rotate.right, open.jaws, close.jaws** are much more easily debugged. Many of these short programs are identical except for which motor is being addressed or which direction the electricity flows. Students begin to understand something important about machines – that the motor itself can be programmed in only a few ways, but there can be dozens of different effects depending on how that motor is attached to an engine and moving parts.

LEGO/Logo is a good way to teach the importance of having a subprocedure that can be changed without altering the whole program. Before students get sensors they will write a program, often called **waity**, that is just a simple wait command. A convertible drives up to a conveyor belt (MacDonalds), does **waity** as some bricks (Big Macs) roll off the belt into the back seat, and then drives off. As they learn about sensors, students make **waity** more sophisticated. First it may contain just code that waits until a sensor is tripped. But later students place an order for a certain number of Big Macs, the smart conveyor belt counts them, and the car drives off only after the order has been filled correctly. **Waity** has evolved from a simple time holder to the most interesting part of the program, and it has done so without disturbing the procedures that cause the convertible to drive up and away again.

Joysticks or keyboard control programs using **readchar** run best if they rely on small procedures. In the first program for the rescue gondola, the machine had to start at exactly the right place, move carefully to one chosen spot, lower exactly the right length of string on the magnet, etc. It was hard. A little friction on the line, a tangle in the string, a carelessly placed paper clip were each enough to throw off the whole plan. Gradually the programming evolved so that there were four small procedures: **gondola.forward, gondola.back, magnet.down**, and **magnet.up**. What's more, each of these commands worked for only a short period of time. The gondola lurched along the string in increments, but that meant that paper clips could be placed anywhere. The magnet inched its way down and back up, but that meant that when we tilted the wire, the magnet could pick up paper clips that were different distances from the wire. Best of all, the students learned a little about the idea of feedback. Depending on how far they needed to move and how far they had gone they could decide on their next command. They learned the idea of controlling something by breaking a complex task into a series of short more easily controlled tasks.

Plastic studs and gears and axles make many machines. A score of primitives make many programs. A half dozen little programs empower students to move machines accurately. Using keyboards or joysticks made of touch sensors, the kids are part of a feedback loop they have created between the machine and the computer. To be successful at this kind of programming, students have to program their ideas in meaningful chunks. I used to think that Logo gave to LEGO, that the machines were made alive and smart by programming. Now I know that LEGO gives back to Logo, that it is a wonderful arena for modeling very important ideas about modular thinking and bite sized ideas.▲

Marian Rosen is Instructional Technology Coordinator at Conway Elementary School of the Ladue Public Schools. She is also the President of ISTE's SIG Logo. She may be reached at:
Conway Elementary School
9900 Conway Road
St. Louis, MO 63124
314-993-2878
mbrosen@oui.com

Stock Up On LEGO® TC logo Books

Do you need more reference guides and student project booklets? A set of these books was included in your LEGO® TC logo or LogoWriter® Robotics kit. Now you can obtain them in quantity at low cost.

LEGO® TC logo Student Pack
- 3 copies of *Getting Started*
- 3 copies of *Making Machines*
- 3 copies of *Teaching the Turtle*
- *Reference Guide*
- *Quick Reference Guide*

LEGO® TC logo Teacher Pack
- *Teacher's Guide*
- *Reference Guide*
- *Quick Reference Guide*

Each Pack is Only $9.95. Turn to page 15 to place your order.☞

This offer is made possible by a generous contribution of these materials from LEGO Dacta. The Logo Foundation is pleased to join LEGO Dacta in providing continuing support for educators who are using LEGO® TC logo and LogoWriter® Robotics.

StarLogo is Here

In the last issue of Logo Update, Carol Sperry reviewed Mitchel Resnick's *Turtles, Termites, and Traffic Jams*. The book describes explorations using a massively parallel version of Logo known as StarLogo.

A Macintosh version of StarLogo has been developed by Mitchel Resnick, Brian Silverman, Andy Begel, and Randy Sargent at MIT. It is being made available by MIT for educational and research purposes under the following conditions:

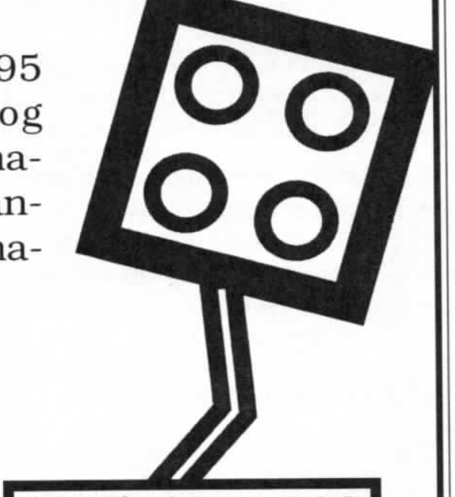
As a service to the Logo community, The Logo Foundation is distributing StarLogo for the cost of shipping and handling. If you want to obtain a copy, and you agree to abide by the conditions set out by MIT, use the order form on page 15 to place your request.

LEGO dacta®

The *educational* division of the LEGO Group.

For a limited time only we are offering **20% off our LEGO® TC logo and LogoWriter® Robotics product line.** Call us for details.

Ask for your free 1995 LEGO DACTA catalog which includes information on our new Logo language programming materials – LEGO DACTA® Control Lab™ and Control System.



LEGO Dacta
555 Taylor Road
P.O. Box 1600
Enfield, CT 06083-1600

(800) 527-8339

Varieties of LEGO/Logo

The term LEGO/Logo refers to various combinations of LEGO building kits and Logo software:

LEGO® TC logo is the first LEGO/Logo product from LEGO Dacta® (800 527-8339). The kit includes building elements, lights, motors, and sensors. An interface box attaches to an Apple II or MSDOS computer. The software is a modified version of LogoWriter. A reference guide, a teacher's guide, and student project books are included.

LogoWriter® Robotics is the same as LEGO TC logo except that the software combines all of LogoWriter with all of LEGO TC logo. This allows you to do your LEGO TC logo and LogoWriter projects with the same software.

Control Lab™, for Macintosh and MSDOS computers, is the newest LEGO/Logo product from LEGO Dacta. It includes software based on MicroWorlds Logo, and improved sensors, motors and connectors. The written materials are aimed at a junior high and high school audience.

Control System includes the same software as Control Lab with a different collection of building materials and written materials geared to upper elementary age students.

Logo Robotics from **Terrapin™ Software** (800 972-8200) allows you to control a LEGO buggy and other devices using Terrapin Logo, Logo PLUS, or PC Logo.

PC Logo™ Robotics, from **Harvard Associates** (800 774-5646) connects PC Logo to Fischertechnik and other construction kits.

The **LASY Robotics System** combines the LASY construction kit with **WinLogo™** from Softeast Corporation (617 893-4858).

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

Turtle Math and MicroWorlds Math Links from LCSI

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

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

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

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

Both packages support the NCTM Standards. **Turtle Math** is available for Macintosh computers; **MicroWorlds Math Links** is available for Macintosh and IBM computers.

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

1-800-321-5646.

LCSI®

And bring a little more sunshine into your classroom

Book Review

by Carol Sperry

I Won't Learn From You: The Role of Assent in Learning
by Herbert Kohl, Milkweed Editions, Minneapolis, 1991

This book is a must have! In just thirty-nine pages, Herbert Kohl provides myriad examples of why children and adults make decisions to "not-learn," to stay outside any system that threatens their integrity. Culling examples from his own experiences, as well as drawing on his long history as an educator, Kohl shows how learning to "not-learn" can actually be a positive and healthy response to a controlling and often hostile society. This small but insightful tract should be on every teacher's bookshelf. It discusses, simply and reasonably, why learning is often resisted. It intersects mightily with an educational philosophy that many of us in the Logo community hold to be essential in dealing with the mysteries of learning.

Spurred on by a Spanish-speaking grandfather's refusal to learn English so his grandchildren would not forget who they were, Kohl remembers his own experiences as a young boy who chooses to "not-learn" Yiddish, though it would have given him entree to lively discussions in the intimate company of his father and paternal grandparents. It was years later that he realized his decision stemmed from a desire to ally himself with his mother, who could not speak Yiddish; he would not allow himself to be privy to conversations she could not understand. This non-obvious perspective helps us open our minds to a variety of situations that might influence refusal to learn.

Kohl then takes us through a panorama of resisters, including Barry, a first-grader who had been held back by his previous teacher "for being uncooperative, defiant, and 'not ready for the demands of second grade.'" His calculated tantrums had gained him status in the class, frightened his previous teacher and got her to leave him alone. Kohl found Barry to be sensitive and intelligent, the best fighter and athlete in the class and a funny storyteller – "confident and cocky but not rude." He felt he was certainly bright enough to read but had never gotten close enough to a book to do it. But now his reputation was at stake and he would hardly go willingly into the learning process. Kohl's analysis and solution to this problem reads like a mystery story – one that has a happy ending and keeps the child's integrity completely intact. I think this kind of story also gives permission to teachers to follow their own instincts, to be creative – not the stuff you'll find in text books.

We meet Rick, "an articulate, conscious not-learner," who rejected the "conventional values of middle-class life." Rick rebelled against testing,

Logo PLUS™

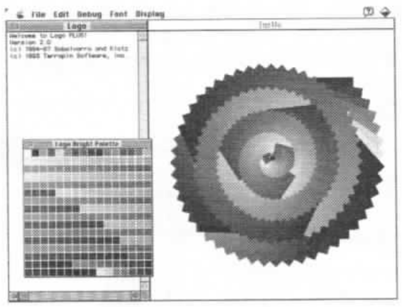
for the Macintosh

Logo PLUS for the Macintosh v2.0 is packed with exciting new features and 97 new commands.

Just imagine! Now you can…

- create pictures using 16 or 256 different colors
- create custom colors of your own
- preview your color selections in a Display window
- save your window settings with one command
- pop to the Turtle window and type text
- turn your turtle shapes to point in four directions
- flip and rotate shapes in the enhanced shape editor
- lock and unlock your shape's orientation
- create animations using 92 new ready-made shapes
- add music to your programs—Bach or the Beatles!
- select text and control the cursor with commands
- play 5 new Logo games *and much more!*

Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103
1-800-972-8200



Welcome to Logo PLUS!
Version 2.0
(c) 1984-87 Sobalvarro and Klotz
(c) 1995 Terrapin Software, Inc

Now Shipping!

Requirements: Macintosh® computer with 4 MB of RAM running System 6.0.5 or higher or System 7. Color optional. 32-bit compatible. Site License version is network aware.

Single Package: **$99.95** Site License: **$450.00**
(site license fee covers 25 machines; $5 for each additional machine)
5-Pak, **$199.95**; 10-Pak, **$299.95**; 20-Pak, **$399.95**
Upgrade from Terrapin Logo/Mac Site License: **$200.00**
from Single Package: **$30.00**, **$10** for each additional disk
from other Terrapin Logo languages: **$60.00**
Complete set of all-new documentation: **$15.00**
(when ordered with upgrade)
Add 5% (minimum $5.00) for U.S. shipping and handling. Foreign charges as incurred.

F395

and though he was forced to attend school, he could not be forced to perform tasks that had to do with assessing his abilities. Talented in mathematics, Rick had to "not-learn" very seriously in order to fail algebra three times in high school.

As a graduate student in the sixties, Kohl taught a course at Teachers College in New York City and became friends with a student, Akmir, a member of the Nation of Islam. By relentlessly drawing attention to racist language and statements made in class, Akmir helped Kohl to consider and understand the many ways we think and speak that are racist. This resulted in a growing empathic response to difference and helped Kohl to understand and appreciate, at a later time, the efforts of a young woman who consistently questioned sexist language in class.

These and other stories are told with an ease that helps us feel we are there, meeting these complicated learners, engaging in their experiences, congratulating them on their persistence, and suffering through the difficulties and the occasional unhappy ending. We see here the obvious and not-so-obvious reasons for not-learning, and we can examine our own experiences for other instances. It might cause you to smile ruefully at what you chose to resist; I not-learned how to type, which in those days, I thought, was a one-way ticket to the secretary's pool, and which handicaps me to this day. It might also make you wince at memories of past students whose not-learning you misread. I won't tell you my own memories of those. The message of I Won*'t Learn From You* can provide teachers with a powerful tool for reflecting on their practice and a reason for hesitating to label as failures some students who are actually choosing to not-learn.

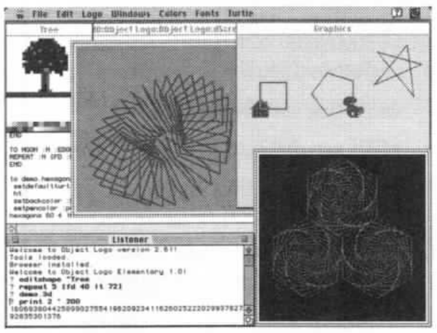
I'll stop here so the review doesn't become longer than the reviewed. This piece is published as part of the Thistle Series of Essays for Milkweed Editions Press. Its length and cost make it accessible, and illustrate a publishing model worth emulating. Kohl says what he has to say and that's that. There is no "padding" to make it into a BOOK that would cost twice as much. I wish there were more such tracts.▲

Carol Sperry is at the Harvard Graduate School of Education. She may be contacted at:
41 Linnaean Street
Cambridge, MA 02138
carols@media.mit.edu

I Won't Learn From You may be ordered directly from the publisher for $4.95 plus $1.50 for shipping and handling.

Milkweed Editions
430 First Avenue North
Suite 400
Minneapolis MN 55401
(612) 332-3192

*Object***Logo**™ for the Macintosh® - **The Logo Language for All Ages!**



```
TO MOON :N :SIDE
REPEAT :N [FD :
END

to demo.hexagons
 setdefaultturt
 ht
 setbackcolor :
 setpencolor :p
hexagons 60 4 1
```

```
Welcome to Object Logo version 2.6!!
Tools loaded.
Browser installed.
Welcome to Object Logo Elementary 1.0!
? editshape "Tree
? repeat 5 [fd 40 lt 72]
? demo.3d
? print 2 ^ 200
1606938044258990275541962092341162602522202993782792835301376
```

Regarded by educators as the most powerful Logo on the market, *Object*Logo is now also one of the easiest Logo languages to learn and use. Thanks to the 180-page highly acclaimed tutorial, *Logo for the Macintosh,* by Harold & Amanda Abelson. Whether your interest is for home or school, give *Object*Logo a try. The Student Edition (includes the tutorial) is well-suited for the beginner. The Full version (includes the tutorial and the Reference Manual) is for a more serious exploration of programming on the Macintosh. Lab Packs include both the tutorial and Reference Manual.

Features Include:

- Support for 32-bit Addressing
- Extensive Math Features
- Advanced List Processing
- Full Macintosh color
- Complete Object System
- Fast Incremental Compiler
- Supports Multiple Windows
- LEGO® Logo Support
- Comprehensive Tutorial

Student Edition	Full Version	Five Lab Pack	Ten Lab Pack	Twenty Lab Pack
$75.00	$195.00	$341.00	$487.00	$585.00

Paradigm SOFTWARE

P.O. Box 2995 • Cambridge, MA 02238 • tel: (617)576 7675 • fax: (617) 576 7680

Public Domain Logo

UCBLogo is a version of Logo written by Brian Harvey. It is in the Public Domain, which means that it may be freely copied and distributed.

UCBLogo lacks the fancy graphics of modern commercial Logos and it doesn't make much use of the Macintosh / Windows type of user interface. But it is a complete implementation of the Logo language and includes advanced features not found in any other version. It is ideal for use with Harvey's *Computer Science Logo Style*.

MSWLogo, written by George Mills, is based on UCBLogo and also takes advantage of features of the Windows environment in which it lives.

UCBLogo comes in Macintosh and MSDOS versions. MSWLogo requires Microsoft Windows 3.1. UCBLogo and MSWLogo are available from the Logo Foundation without charge. You pay only for shipping and handling. Turn to page 15 to place your order.☞

The End Is (not quite so) Near!

In the last issue of *Logo Update* we reported that Volume 1 of *Computer Science Logo Style* by Brian Harvey was out of print, and that there were only a few copies left. Those copies are gone, but recently an additional 27 books were discovered at the bottom of a closet in Berkeley, California. Of these, 23 remain and may be ordered from the Logo Foundation.

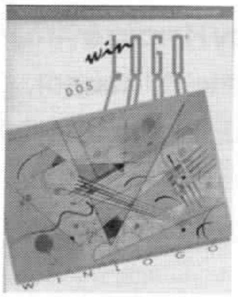
Turn to page 15 to place your order. ☞

While you're there, order UCBLogo or MSWLogo to use with *Computer Science Logo Style*. There's no additional charge for the software.

Open the Power of Logo with WinLogo™

A New Logo Language for Advanced Learning

WinLogo for Windows



Features:

- **12 turtles, color graphics, import of PCX, BMP, EPS, PS, TIFF, TGA files, etc.**
- **300 built-in primitives, math capability, list processing, debuging tool, extension using C or assembly language**
- **font and printer support, icons, on-line help**

	DOS	Windows
Student Edition*	**$ 49**	**$ 59**
Single user	**$ 89**	**$119**
5 User Pak	**$249**	**$349**
10 User Pak	**$299**	**$449**
20 User Pak	**$399**	**$649**

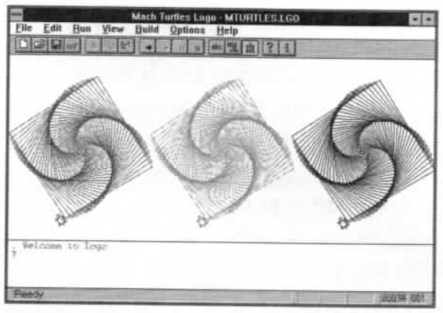
(* no documentation)

Also available: WinLogo/LASY Robotics System

SoftEast Corp.
P.O. Box 764
Concord, MA 01742
Call: (617) 893-4858 or Fax: (617) 893-4948

Mach Turtles Logo

Introducing a version of Logo specifically designed to run in Microsoft Windows. Mach Turtles Logo provides a full-featured version of the Logo language with a modern Windows-compatible user interface.


Mach Turtles Logo - MTURTLES.LGO
File Edit Run View Build Options Help
; Welcome to Logo
Ready

Features include:

- A **screen saver builder** that lets you turn Logo programs into Windows screen savers in just a few seconds.
- An **application generator** that builds "standalone" programs.
- **Extended turtle graphics** with **multiple turtles**.
- **MIDI sound** output.
- Extensive **on-line help**.

Requires Microsoft Windows 3.1 or later.

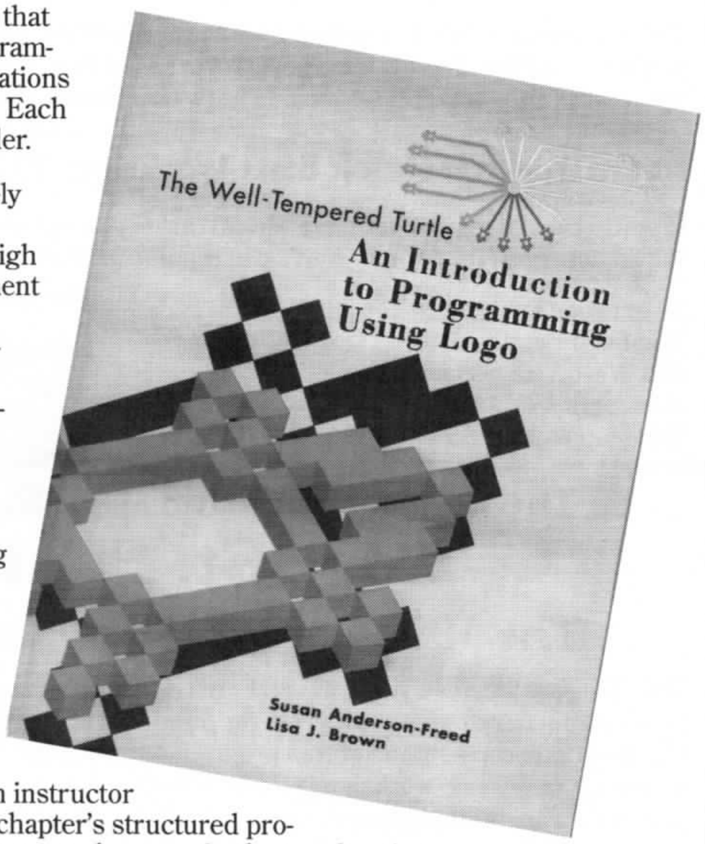
Single User	5 Lab Pack
$49	$99

Softsparks
1-800-390-7971
P.O. Box 68039
Hamilton, ON, L8M 3M7
905-527-3878

The Well-Tempered Turtle

An Introduction to Programming Using Logo

by Susan Anderson-Freed and Lisa J. Brown



The Well-Tempered Turtle is a new curriculum that uses Logo as a means of testing and exploring programming concepts. It emphasizes learning Logo applications and highlights Logo's unique programming power. Each chapter is independent and may be used in any order.

The Well-Tempered Turtle has been extensively field-tested in introductory college level computer science courses and is appropriate for students of high school and college age. By utilizing Logo to implement examples, *The Well-Tempered Turtle* has students quickly writing their own programs to explore computer science concepts. Students build on simple introductory programs to explore increasingly complex subjects, progressing for example from line drawings to fractals and bit-mapped graphics.

The Well-Tempered Turtle also provides a complete introduction to computer science covering such topics as data types, control structures, graphics, natural language processing, and music. Appendices provide supplementary information on the history of computers, mathematics and grammar.

Since *The Well-Tempered Turtle* contains more material than can be covered in a semester, an instructor can pick and choose the topics to emphasize. Each chapter's structured progression encourages students to learn at their own pace and pursue further exploration.

The Well-Tempered Turtle is written by Dr. Susan Anderson-Freed and Dr. Lisa J. Brown, Professors of Computer Science at Illinois Wesleyan University. Together they have more than 27 years' experience teaching mathematics, programming and computer science. Their Logo courses are both highly demanding and in high demand among students at Illinois Wesleyan, and always fill immediately.

250 pages. **$49.95**

To order *The Well-Tempered Turtle,* please call 1-800-774-LOGO
or fax 1-800-776-4610

HARVARD
ASSOCIATES, INC.

10 Holworthy Street • Cambridge, MA 02138 U.S.A. • Phone (617) 492-0660 • Fax (617) 492-4610
Compuserve 70312,243 • Internet pclogo@harvassoc.com

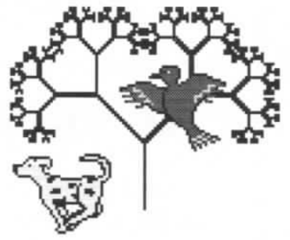
MicroWorlds™ Quick Start Workshop

•You've been using Logo and you've seen or heard about MicroWorlds. Should you switch?

•You've just made the change to MicroWorlds and you have many questions. What's the same as in your old version of Logo? What's new and different? What can you and your students do with it?

This workshop focuses in on the new features of MicroWorlds Logo:

✔ Enhance your Logo projects with buttons, sliders, text boxes, drawing tools, and a melody maker.



✔ Create animations and turtle graphics with many turtles programmed to do different things simultaneously. Now the bird can fly while the dog wags its tail while...

✔ Move text, graphics, and sound between MicroWorlds and other applications.

This three-hour workshop includes both presentations and hands-on time. You will develop a complete MicroWorlds project. The MicroWorlds Quick Start Workshop is conducted at your school on your computers for up to 20 people. The cost is only $390 plus travel. (That's less than $20 per person.) All materials are included.

MicroWorlds™ Quick Start Workshop PLUS

If your school does not yet have MicroWorlds you can obtain a site license and have a MicroWorlds Quick Start Workshop, both for $1095. That's the regular price of the site license alone.

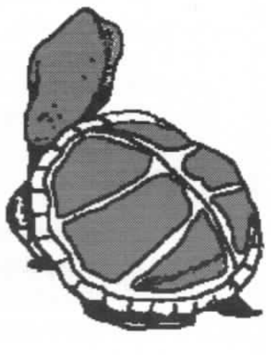
<u>To arrange for a workshop contact:</u>
The Logo Foundation
250 West 57th Street, Suite 2228
New York, NY 10107-2228
Telephone: 212 765 4918 Fax: 212 765 4789

If you would like more information, use the response form on page 15 to request a detailed outline of the **MicroWorlds™ Quick Start Workshop.** ☞

The Logo Foundation and the **St. Paul Public Schools** announce . . .

Logo St. Paul

The 1995 Logo Summer Institutes



Over the past fourteen years the St. Paul Logo Project has provided a comprehensive professional development program for hundreds of elementary and secondary school teachers. The cornerstone of this program has been the **Logo Summer Institute**, an intensive one-week workshop which provides for an immersion in Logo theory and practice. The individualized approach of the Logo Summer Institute accommodates experienced Logo users as well as novices.

A limited number of places are being set aside for people from outside the St. Paul Public Schools.

✔ The registration fee includes
- all workshop materials.
- use of a Macintosh or Apple II computer.
- use of whichever version or versions of Logo you choose: MicroWorlds, LogoWriter, Object Logo, Logo Plus, UCBLogo, and StarLogo.
- Optional attendance at two follow-up workshops in November 1995 and March 1996, each a day and a half long. The topics and dates will be determined during the Summer Institutes.

✔ As an option, you may also receive three graduate quarter credits from Hamline University.

✔ Major discounts on purchases of Logo software are available to Summer Institute registrants.

<u>When</u>:	June 26 - 30 or
	August 21 - 25
<u>Where</u>:	St. Paul, Minnesota
<u>Cost</u>:	$490 per person
	$120 for graduate credit

Use the response form on page 15 to request registration materials for the 1995 Logo Summer Institutes and to obtain more information about the St. Paul Logo Project. If you are a teacher in the St. Paul Public Schools these registration procedures and fees do <u>not</u> apply to you. Instead, contact Ms. Geraldine Kozberg at 360 Colborne Street, 228-3631.

Logo Foundation Response Form

❑ Enter my free subscription to *Logo Update*.

❑ Send me the complete Logo Foundation Catalog of software, publications, and services.

❑ Send me more information and registration materials for the 1995 St. Paul Logo Summer Institutes.

❑ Send me a detailed outline of the MicroWorlds Quick Start Workshop.

Enter my order for:	quantity	amount
❑ *Turtles, Termites, and Traffic Jams* by Mitchel Resnick $24.95 (see page 8)	____	$________
❑ *Computer Science Logo Style, Volume 1* by Brian Harvey $22.95 (see page 12)	____	$________
❑ *LEGO TC Logo Student Pack* $9.95 (see page 7)	____	$________
❑ *LEGO TC Logo Teacher Pack* $9.95 (see page 7)	____	$________
❑ MicroWorlds Project Builder $99.00* ❑ Macintosh ❑ MSDOS (see page 9)	____	$________
❑ MicroWorlds Math Links $79.00* ❑ Macintosh ❑ MSDOS (see page 9)	____	$________
❑ Turtle Math (Macintosh only) $69.00* (see page 9)	____	$________
❑ Object Logo Student Edition (Macintosh only) $75.00* (see page 11)	____	$________
❑ Object Logo Full Version (Macintosh only) $195.00* (see page 11)	____	$________
❑ Logo PLUS for the Macintosh $99.95* (see page 10)	____	$________
❑ WinLogo ❑ for MSDOS $89.00* ❑ forWindows $119.00* (see page 12)	____	$________
❑ Mach Turtles Logo $49.00* (see page 12)	____	$________
❑ UCBLogo (no charge) ❑ Macintosh ❑ MSDOS (see page 12)	____	$________
❑ MSWLogo (no charge) (see page 12)	____	$________
❑ StarLogo (no charge) (see page 8)	____	$________

Subtotal .... $________

Shipping and Handling .... $________
<u>United States:</u> $3.50 on orders up to $70.00, 5% of subtotal on orders over $70.00
<u>Canada and Mexico:</u> $7.00 on orders up to $70.00, 10% on orders over $70.00

Tax deductible contribution to the Logo Foundation .... $________

*These prices are for single-user sets. Call or write for prices of lab packs and site licenses. These products are available from the Logo Foundation only within the United States. Contact the developer for information about international distribution.

Total ....$________

Name ____

Organization ____

Address ____

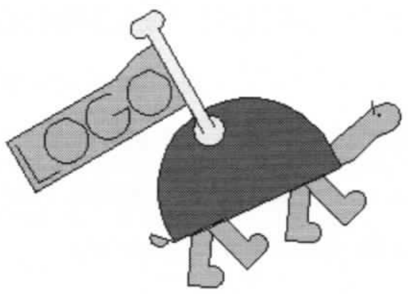
City____________________ **State**________________ **Zip**__________ - __________

Day Phone (   )________________ **Evening Phone** (   )________________

Please enclose payment or a school purchase order.

Overseas orders require additional shipping charges. Please inquire before ordering as the amount depends upon destination and carrier.

Logosium '95



Baltimore • June 16, 1995

A full day of Logo discussions, sharing sessions, and presentations including:

- **Music, Math, and Logo**
- **Exploring Genetics with Logo**
- **Meet Dorothy Fitch, the new Editor of *Logo Exchange***
- **StarLogo**
- **MicroWorlds**
- **Logo PLUS**
- **LEGO/Logo**

Contact:
Marian Rosen & Michael Tempel
c/o Logo Foundation
250 West 57th Street, Suite 2228
New York, NY 10107-2228
Telephone: 212 765 4918 Fax: 212 765 4789
e-mail: mbrosen@oui.com michaelt@media.mit.edu

For registration and hotel information contact:
NECC '95
1787 Agate Street
Eugene, OR 97403-1923
Telephone: 503 346 2834 Fax: 503 346 5890
e-mail: necc95@ccmail.uoregon.edu

Logosium is an NECC '95 pre-conference activity sponsored by the Logo Foundation and ISTE's SIG Logo.

LOGO USERS GROUPS

Long Island Logo Users Group
Contact: Marilyn Tahl
516 333-4018 (evenings)
516 627-8110 (days)

Los Angeles Logo Users Group
Contact: Carolina Goodman
Campbell Hall
4533 Laurel Canyon Blvd.
North Hollywood, CA 91607
818 980-7280 ext.234

Logo Anonymous
Contact: Marian Rosen
Conway School
9900 Conway Road
St. Louis MO 63124
314 993-2878

New York Logo Users Group
Contact: The Logo Foundation
212 765-4918

Philadelphia Logo Users Group
Contact: Mel Levin
Prince Hall School
Godfrey and Gratz Avenues
Philadelphia PA 19141
215 276-5369

Logo Foundation
250 West 57th Street • New York, NY 10107-2228

Nonprofit Org.
U.S. Postage
PAID
New York, NY
Permit #6378