Logo Update The Logo Foundation Newsletter ▲ Volume 2, Number 3–Spring 1994

In This Issue

The first version of Logo that I played with was TILOGO. It had fast-moving sprites that could change color and shape, and a rather anemic turtle which ran "out of ink" when it tried to draw anything complex.

Meanwhile, the Logo implementations for the Apple II had no sprites and few colors, but the turtle could fill the screen with elaborate graphics.

The kinds of projects that emerged in each environment – games and animations on the one hand; drawings and patterns on the other – reflected these differences.

Now we have MicroWorlds, the first radically different Logo to come along in many years. What will emerge in this environment? This issue of *Logo Update* includes two tentative answers with articles by Kathryn Verzoni and Gerald Crisci about their students' MicroWorlds projects.

The software environment influences, but does not determine the direction that people take with it. What you do with any version of Logo is a matter of choice. In "Advancing Logo," Seymour Papert reflects upon a discussion with a group of teachers about moving in new directions with Logo: introducing new kinds of

(Continues on next page, first column)

By Seymour Papert

Advancing Logo*

In the last issue we were grappling with the theoretical problem of defining what kind of Logo is advanced Logo. Soon after writing my contribution I found myself in a meeting of Logo teachers who were grappling with a related but very practical problem. They felt that many of their students were not advancing in programming skills. For example even after several years of doing Logo, students still composed long and intricate strings of Logo instructions and resisted all suggestions of organizing their work as subprocedures. The teachers saw the situation as a "bug" in their own work and had tried, so far with limited success, a number of strategies to remedy it. Some of them, working on the assumption that the students did not understand the idea of subprocedure, had developed teaching units to explain some principles of structured programming. Others tried to influence the students by showing concretely how their code could be written differently. The teaching materials they developed were excellent and I am sure that these teachers would have exercised great skill in engaging students in discussion about the relative merits of different styles of programming. Why then did their strategies not succeed?

In the course of the discussion I came to appreciate more sharply than I had before the force of another factor that enters into this kind of situation. One might call it "the Piaget insight" which I formulated in the following way back in the days I when worked in Geneva. One of the major contributions Piaget made to the investigation of children's thinking is understanding that it is not right to label as "wrong" children's declarations that there are more red beads than blue beads or more water in the tall glass. If you truly succeed in putting yourself in their intellectual framework (no easy task!) you will understand that "the children are correctly answering the questions they are really answering . . . only their questions are not the same as yours even if both use the same words." This does not mean that children, like the rest of us, aren't sometimes just plain wrong. But the whole point of Piaget's empathic and interactive way of conducting these conversations is to distinguish between errors or slips or "mere misunderstandings" and what the children really believe.

The Piaget insight applies to programming. For example, in the early days of our work at the Hennigan School a remarkable fifth grader, Nicky Brown (now an MIT undergraduate), put me right when I tried to advise him to break up a program into subprocedures. He systematically refuted all my arguments. No, it would not help him understand his program more clearly: he demonstrated by his ease of debugging and modifying his work that his own way of understanding it worked perfectly well. Well yes, subprocedures might help others take over parts of his work, but this wasn't his main goal and besides he didn't really think that they were interested anyway. Finally it was clear from his discussion that he just didn't like the kind of structuring of his

(Continues on next page, second column)

* The following is based on an interaction with a group of teachers who will surely recognize themselves in it. I decided not to identify them because the limited space available here forces me into an oversimplified presentation that casts me in the role of "the expert" coming with "answers" to problems that other people ("just teachers") couldn't solve. I hate this role. In reality the discussions were far richer and more creatively interactive. A fuller narrative would cast me as a catalyst rather than as an expert.

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)

projects, approaches, and programming techniques.

This issue of *Logo Update* also includes readers' responses to earlier articles, and news about courses, workshops, and new products. One of these new products is Turtle Math, a specialized version of Logo which is described in an article by its creators, Doug Clements and Julie Meredith.

And don't forget the National Educational Computing Conference, which will be held in Boston from June 11 to 15. In addition to the many Logo workshops and presentations planned for the conference, there is Logosium, a full-day Logo symposium, scheduled for Sunday, June 12. This event is sponsored by ISTE's SIG-Logo, the Logo Foundation, and the MIT Media Lab. Turn to page 4 for more information. I look forward to seeing you there.

Michael Tempel

(By Seymour Papert, continued from page 1)

project that would be produced by the subprocedurization I suggested.

Remembering Nicky Brown – and many other talented young programmers – was a sharp reminder that there is something wrong with describing the problem raised by these teachers as if their students lacked "programming skill." They may have lacked one particular programming skill, but what their style of programming required was another kind of skill – and when one looks at how fluently many students debug their "spaghetti code," one is impressed by the level of that skill.

This diatribe against imposing structured programming on these children is not intended to avoid the issue that it would be immensely valuable for them to learn new programming ideas. Quite the contrary, it led me to a crisper formulation of an alternative strategy to achieve this not by trying to force new ways to write the children's old programs but by introducing them to new programming domains in which other ideas would come up more naturally.

My new insight consisted of looking more closely at the development of the practice of Logo in the schools from which these teachers came – and when it was finally formulated I see that it applies on a much larger scale perhaps to the way that Logo programming in schools has developed across the board. One of the most positive educational benefits that came with the introduction of Logo into the schools I was looking at was a shift towards project-based learning. An excellent progressive step. But every progress risks bringing a certain kind of conservatism in its wake. In this case the conservatism took the form of establishing a certain model of "project" and this model of project carried with it a match to a certain style of programming, in fact precisely the style from which the teachers were now trying to wean their students.

The style of project in question developed a wide presence with the introduction of LogoWriter. Schematically described, it consists of using Logo to make a presentation of the results of research conducted by students. In the best cases Logo serves as more than just presenting ideas that are already formulated; the struggle to represent material is part of thinking about it. Some examples of this kind of work have become well known in the Logo community. Early examples included a project on the life cycle of the fly developed by Marian Rosen's computer summer camp group, and projects by students in Joanne Ronkin's class at Hennigan on such topics as the human skeleton. From Costa Rica came a bold students' project on human reproduction and many on historical and ecological themes. There was no doubt in the minds of the teachers at the meeting I am reporting here that such projects were educationally valuable and a good use of Logo. What emerged only slowly during the meeting was an awareness that the projects favor a certain style: they are essentially narrative projects and favor a "narrative" style of programming that looks in the end more like a page full of text than like a structured piece of mathematics. Indeed I would go further and say that there has been a co-evolution of this kind of project and this style of programming.

The best way to diversify the style of Logo programming is to diversify the kinds of projects for which Logo is used. A clear example is provided by Lego-Logo. Programming a Lego turtle to follow a light requires programming based on repetitive runs of conditionals. But there is room for infinite diversity without hardware beyond the computer. Writing a video game in the style of Pacman or Mario requires efficient testing for new conditions and thinking carefully about a user interface that will permit rapid response. Simulations involving probability lead into issues of representation. All of these put so much of a real premium on small self-contained tool-like procedures that there is no need to persuade students to use them. They are what comes naturally in these contexts. ▲

Modeling Real Life with MicroWorlds Project Builder

by Kathryn Verzoni

What factors affect the probability that Joe will get a date? How do diet, and timing and dosage of insulin affect blood sugar levels in juvenile diabetics? How accurately can I simulate a wheel on an inclined plane using Logo? These are just a few of the questions my seventh grade students pondered as they used MicroWorlds to model real-life cause and effect relationships.

MicroWorlds' slider is a wonderful vehicle for developing understanding of variables. The ability to both display and change a value directly on the screen enables kids to get a concrete feel for a concept that is too often abstract. Our experiences with this new feature led naturally into its use in projects.

The project design problem: Analyze some aspect of real life for cause and effect relationships. Define at least four factors that would have significant effects upon some dependent variable or outcome. Develop equations that will describe the relationship between the factors and the outcome. Finally, use MicroWorlds to build your model. Your model should be dynamic and interactive. Any user (even your five-year-old brother) should be able to manipulate values and witness the effects of such changes.

What followed was an onslaught of ideas ranging from the very silly to the serious. No matter what the selected domain, each learner needed to think about concepts, relationships, and interactions in ways they (and I) never had before.

During the early stages of simulation development, defining an effect and distinguishing it from the causes proved to be quite a stumbling block. On several occasions when I was called over for help, a student would say, "Here are my factors, but now I don't get what to do next." My question was, "Well, what are you trying to model here? What do these factors affect?" "Oh, . . I don't know. . . . I mean, wait. . . ." Often, they had included the "effect" as one of the causes. Their befuddlement and subsequent progress toward ultimate resolution was fascinating and raised issues relating to differences between correlational and causal relationships.

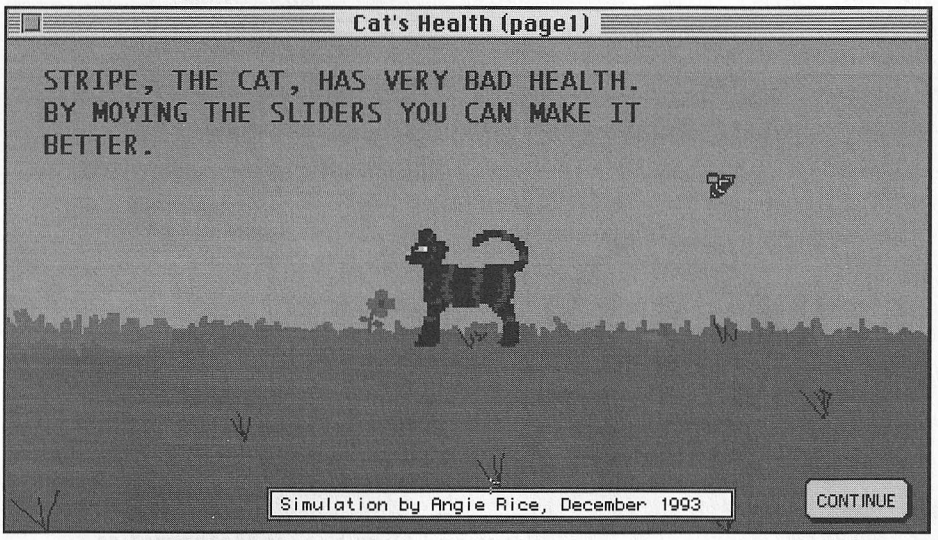
Learners continually found themselves faced with the problem of how to describe a relationship in terms of an equation that Logo would understand.

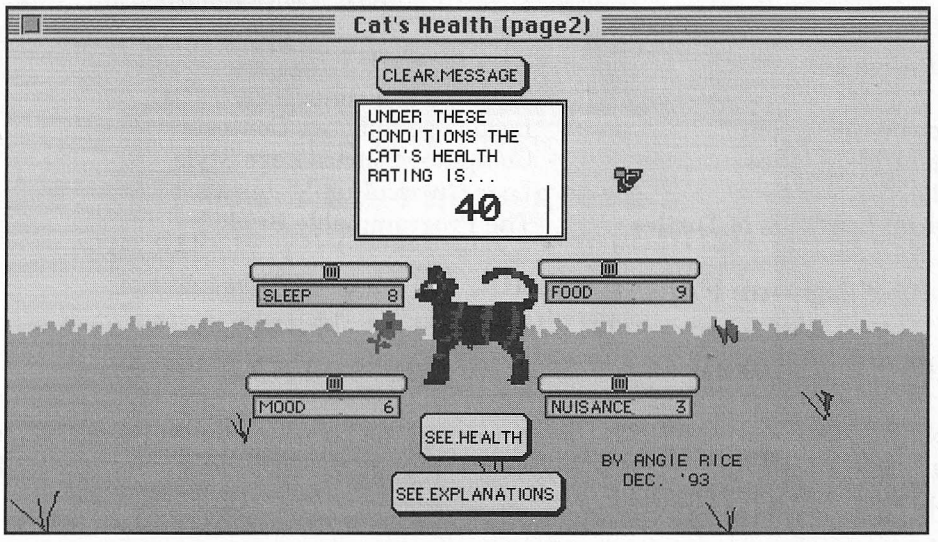
One student decided to develop a simulation of four factors that affect a cat's health. In Angie's mind, "mood rating" and "number of nuisance dogs as neighbors" had straightforward, linear effects. The better the cat's mood rating, the better his health. The more nuisance dogs he had to co-exist with, the worse his health.

However, thinking about "hours of sleep" and "amount of food eaten" caused her more trouble. Angie asked, "How would I make it so that the more a cat eats, the better his health, but only to a certain point? After that, the more he eats, the worse his health will be."

When posed with this question, I quickly recalled and tried to apply my high school algebra – NOT! Abandoning that approach, I initially found that I had no idea how she might go about such a task. Finally, I showed her how to set up ranges. The solution we came up with required a number of **if** statements. It served as a fair approximation of how she thought that relationship

(Continues on next page)





(Continued from page 3)

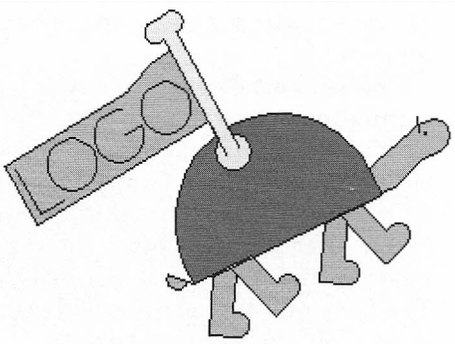
would work in real life.

It wasn't until a few weeks after the kids had completed their projects that I started to make connections between the business of causal modeling and school duties such as solving for x, making "x y" tables, and plotting functions. Too bad I didn't make these connections nineteen years ago.

With modeling experiences like these under their belts, maybe – just maybe – our future high-schoolers might begin to make some connections between school math and everyday life. ▲

Kathryn Verzoni teaches at the West Point Post Schools, West Point, NY.

```
to message
talkto "text1 ct
print [under these conditions the cat's health rating is...]
repeat 8 [insert char 32]
setfontsize 20 setstyle "bold
insert :healthrating
repeat 2 [insert char 32]
end

to calc.health
if food > 5 [make "food minus food]
if food < 6 [make "food food]
if sleep > 8 [make "sleep minus sleep]
if sleep < 9 [make "sleep sleep]
make "healthrating :sleep + :food + mood - nuisance.dogs + 38
end

to see.health
calc.health message
end

to continue
getpage "page2
text1, ct
end

to go.back
getpage "page2
end

to see.explanations
getpage "page3
end

to clear.message
text1, ct
end
```



Logosium - Sunday, June 12, 1994

- **Sponsored by ISTE's SIG-Logo and the Logo Foundation**
- **Hosted by the MIT Media Lab**
- **An ISTE Pre-Conference Activity at NECC '94, Boston**

For Registration Information Contact:
NECC '94 Office, Lesley College
29 Everett Street • Cambridge, MA 02138
Phone: (617) 349-8965 • Fax: (617) 349-8968
Internet: necc94@bbn.com

Discussions, Sharing Sessions, and Presentations (Partial Listing)

Long Lasting Logo Environments
Global Logo Links
Action Research
Inservice Teacher Education
College Logo Courses
Logo in the Lab and in the Classroom
Programming Tricks and Techniques
Linking Logo with Other Software
StarLogo: Learning with Thousands of Turtles
Logo and High School Mathematics
Logo and the NCTM Standards
Logo and Physics
Logo for Little Kids
Children as Game Designers
Lego Logo Robotics Contests
Computer Science Logo Style
Logo Curriculum Materials
The Programmable Brick

Lunchtime Keynote Speech by Seymour Papert

And Jitterbug Logo Style. It's Procedural!

Important Note: The NECC '94 Advanced Program, and *ISTE Update* printed the Logosium fee as $100. This is an error. If you have already registered you will receive a refund. If you have not yet registered, send the correct amount, which is $50.

Pinball Wizards

by Gerald Crisci

One of the challenges of using the project approach when teaching with Logo is to develop activities which have some structure, yet allow children to express their individual creativity.

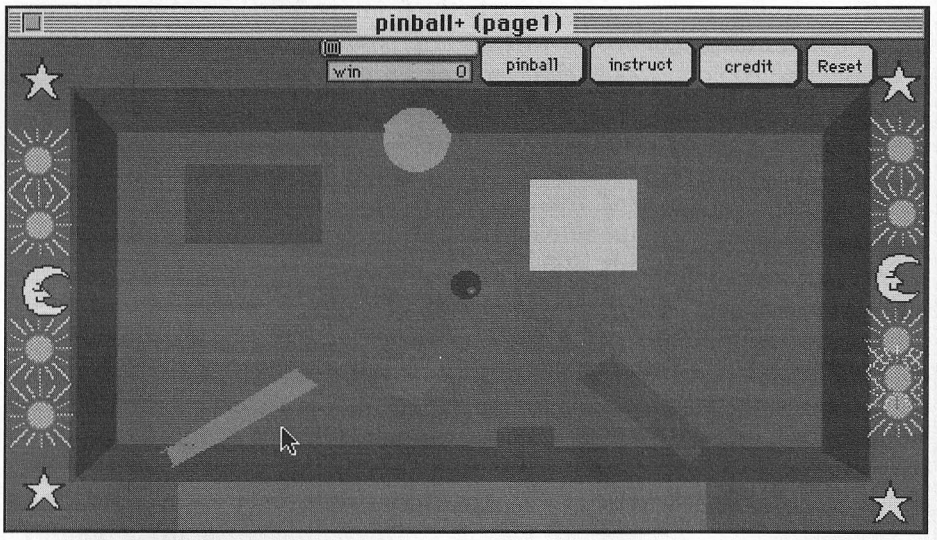
When my students reach the fifth grade, I introduce them to new Logo concepts that allow them to explore more sophisticated ideas like variables and conditionals. During this time, students use Logo to create a simulation of a physical object, such as a working analog watch. This model allows them to construct fairly complicated projects while demonstrating their knowledge of advanced Logo features.

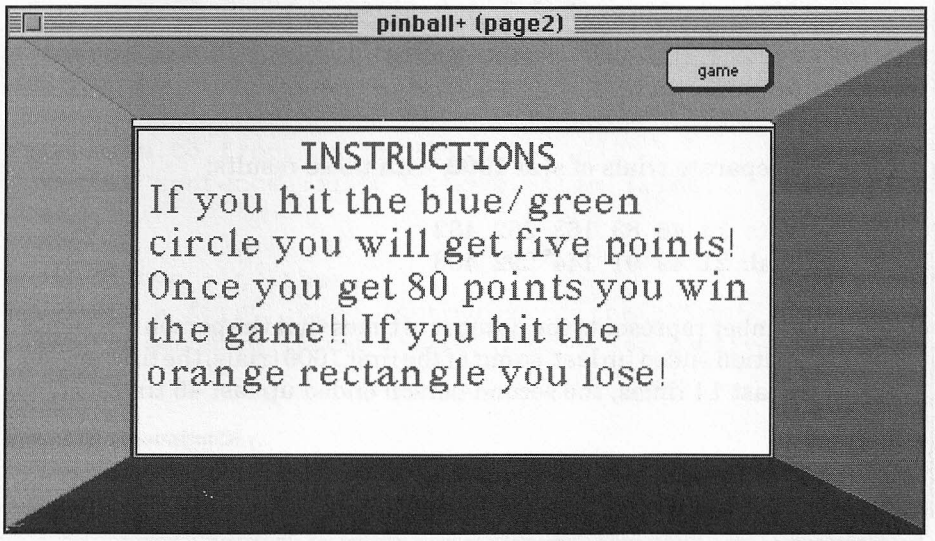
Last year, my students designed pinball games using the Macintosh version of LogoWriter. Since these projects were very successful and motivating, I decided to repeat them with this year's classes. This time students used MicroWorlds to create more sophisticated games. They were encouraged to use their imaginations in creating their designs within a structure I provided.

Each project had three basic requirements:

- The game should contain non-violent subject matter.
- The game should have clear ways to win and to lose.
- Instructions should clearly explain how to play the game.

Fourteen fifth-grade classes in five elementary schools were involved in the project. It lasted approximately four months. Students were provided with examples of new MicroWorlds commands and shown how to incorporate them into a program. I introduced concepts by meeting with classes approximately once every three weeks. The regular teachers brought the classes to the computer lab for additional sessions. Most students worked in pairs.





The pinball projects helped students to integrate many Logo concepts and techniques. They used

- sliders to keep score and to vary the speed of their games.
- conditional statements (**if** commands) to make the ball react to objects on the screen, detect collisions with other turtles, and set winning and losing thresholds.
- multiple pages to provide for different game levels and for instruction screens.
- the Paint Tools to create background graphics and the Shape Editor to create screen objects.
- buttons to link pages together and to control the action.
- coordinate commands (**setpos**, **setx**, **sety**, **mousepos**) to move a "paddle" and to position objects on the screen.

Some students created musical sequences which played at the end of the game, and some made use of the reporters **timer** and **random**.

When the project began, I showed students examples of pinball games I had created myself. These games featured a ball bouncing off objects on the screen and a paddle keeping the ball in play.

Most of the students' initial game designs mimicked what I had shown them, but many went on to create other types of games, including hockey, soccer, and basketball. Others built games with themes based on units studied in school, like acid rain. Many based their projects on fantasy

(Continues on next page)

(Continued from page 5)

themes involving space aliens, pirates, or "smiley faces."

As the year went on, I began to notice a subtle difference between the boys' and girls' projects. Boys generally created games which were functional, yet visually unattractive. Lines on the screen might be crooked and graphics haphazardly placed. Most girls, on the other hand, took great care in the visual design of their games. I conferenced with some boys and helped them to "clean up" the appearance of their screens.

Wrapping up the project was difficult for the young game designers. Many students wanted to add additional levels, game options, and more graphics. We could have spent the rest of the year improving the games and adding new features.

As the year comes to an end, fifth grade students will invite third graders to their classes to play the games they have created. This will allow third graders to see how they will use their own knowledge of Logo in future years to create their own exciting projects. ▲

Gerald Crisci is District Elementary Computer Teacher with the Scarsdale Public Schools. He also wishes to thank Alan Alterman, Scarsdale's Math / Science Enrichment Teacher, for his help in developing this project.

For more information, contact the author at:

Curriculum Center
Scarsdale Public Schools
Brewster Road
Scarsdale, NY 10583
914 721-2430
jerryschol@aol.com

Logo Tool Box

by Michael Tempel

The Tool Box column in the last issue of *Logo Update* included a Logo program which I used to randomize the order of the six authors of the lead article, "What is Advanced Logo?" Brian Harvey, one of the six, has a complaint.

To the editor:

Your shuffling algorithm is unfair! Although everyone has an equal chance of being first, the probability of ending up last depends on the starting position. The person who starts first has only about a .015 chance of ending up at the end, whereas the person who starts last has a .402 chance of staying last. To be fair, the probability should be .167 for everyone.

A person moves toward the end of the line at each call to **shuffle** if someone who was after that person moves to the front. So, in order for the first person to end up last, here's what has to happen:

First shuffle: anyone else moves to the front, p=5/6. Second shuffle: one of the four last people moves to the front, p=4/6. Third shuffle: one of the three last people moves to the front, p=3/6. And so on for the remaining two shuffles, p=2/6 and p=1/6. These five probabilities are multiplied together to give the result .015 as above.

By contrast, in order for the person at the end to remain at the end, all that has to happen on all five shuffles is that someone else be chosen to move to the front, so it's p=5/6 each time. Multiplying these five probabilities together gives the result .402.

I double-checked my math by simulating several trials in Logo. Showing off the array facility of UCBLogo, here's what I did:

```
to sim :n
local [lasts this]
make "lasts (array 6 1)
for [i 1 6] [setitem :i :lasts 0]
repeat :n [make "this last shuffle shuffle shuffle ~
   shuffle shuffle [1 2 3 4 5 6] ~
   setitem :this :lasts (1+item :this :lasts)]
print :lasts
end
```

I did two separate trials of **sim 1000**, with these results:

First trial: 14 46 83 163 262 432
Second trial: 21 49 91 144 292 403

Each number represents the number of times that the person originally in that position ended up last, so out of the first 1000 trials, the first person ended up last 14 times, the second person ended up last 46 times, etc.

(Continues on next page)

(Continued from page 6)

To make your algorithm fair, whenever someone moves to the front, that person should be exempt from further shuffling (so the next shuffle is of a smaller set). Here are my modified versions of your procedures:

```
to shuffle :deck
if empty? butfirst :deck [output :deck]
output move.to.top (pick.from :deck) :deck
end

to move.to.top :item :stack
output sentence :item (shuffle remove :item :stack)
end

to remove :it :them
if :it = (first :them) [output butfirst :them]
output sentence (first :them) (remove :it butfirst :them)
end

to pick.from :group
output item (1 + random count :group) :group
end
```

A single call to **shuffle** now handles the entire thing, because of the recursion, so there is no need for five shuffles in a row. When I simulated 1000 trials of this new version twice, I got these results:

198 163 140 161 176 162
169 155 159 169 176 172

Brian Harvey, who started third and ended up second, is a Lecturer in Computer Science at the University of California at Berkeley. He is the author of the three-volume Computer Science Logo Style *and is the creator of* UCBLogo.

Here is the original **shuffle** program as it appeared in *Logo Update*, Volume 2, No. 2:

```
to shuffle :deck
output move.to.front (pick :deck) :deck
end

to move.to.front :item :list
output sentence :item (remove :item :list)
end

to remove :it :them
if :it = first :them [output butfirst :them]
output sentence (first :them) (remove :it butfirst :them)
end

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

OOPS!

In the previous issue of *Logo Update*, part of Carol Sperry's Book Review was omitted.

The affected sentences, which begin near the bottom of the first column on page 9, should have read:

> For years, I bought many of these books but was too daunted by what I thought I'd find in them or, to be perfectly honest, somewhat fearful of finding myself in over my head. I now regret the time of lost opportunity because these books, though some can be rather difficult, are filled with thought-provoking and imagination-expanding ideas and can be digested, if necessary, a little at a time.

How to Order UCB Logo

UCBLogo is available in versions for MSDOS, Macintosh, and UNIX computers. It is in the public domain, and may be obtained via anonymous FTP from

anarres.cs.berkeley.edu.

Get the file pub/README for details.

If you do not have FTP access, but can read a 3.5-inch disk, send $3.00 to cover costs and specify whether you want the Mac or the MSDOS version, or send $6.00 for both, to:

Brian Harvey
2634 Virginia St.
Berkeley, CA 94709

Outside North America, please send US $4.00 for one disk or US $7.00 for both.

Where Does Programming Fit In?

by Mark Steinberger

The questions, "Is Programming a Good Activity for Children?" (Papert, *Logo Update*, Fall 1993) and "Is Programming Obsolete?"(Clements and Meredith, *Logo Exchange*, Fall 1993) are important questions for us because they go to the heart of the issue, "Why do we want children to work with computers?" The answer to that question is rooted in our vision of schools and their role in a young person's life. The most important task of formal education is to facilitate children's development into lifelong independent learners who take control of their own destinies. They must be imbued with a strong sense of themselves and their ability to act in this world. For this kind of education to be meaningful, children's self-confidence must be backed up with competence in a variety of life skills, or literacies. Computer literacy is one of those skills.

A computer literate student understands that a computer is a tool that people use to accomplish self-defined, real-life tasks. They have a realistic understanding of what a computer can do, even if they don't know the specifics of how to get it done. Implied is an understanding that computers are not mysterious or magical, but useful and controllable. Our goal as educators is to nourish students who see themselves as actors in the computer arena, not passive bystanders, or worse, victims. We want students to grow into adults who will not be intimidated by computers but will go out and use computers to achieve their personal and professional goals.

In the forward to *Mindstorms*, Seymour Papert describes his love of gears. They were an important learning model for him because he was in love with them. He specifically did not propose producing sets of gears for all children to study because ". . . this would miss the essence of the story. I fell in love with gears. This is something that cannot be reduced to purely 'cognitive' terms. Something very personal happened, and one could not assume that it would be repeated for other children in exactly the same form."

Papert, as an adult, has clearly fallen in love with Logo the way he was in love with gears as a child. That is why he uses it to solve a myriad of daily problems. In his column Papert argues that programming is a good activity for children because it can be used to create anything from whimsical doodles to microworlds that assist in the exploration of a variety of interesting situations. Papert relates a number of stories that describe how he used programming to explore interesting relationships and phenomena.

Papert tells the story of a Logo alarm clock. He used Logo because he loves it and didn't have access to a clock radio. Someone else might have used the Mac's microphone to record a loud, repetitive sound. His Macintosh alarm would have then played that sound, rather than the simple beep, at the appropriate time. Not as elegant as Logo, but it gets the job done. If he had a telephone (How many classrooms do?) he might have called a friend and asked him to call back and wake him. Perhaps someone would have built something out of tinker toys.

When Logo was first introduced, it provided the only avenue for children to control a computer. It promoted the idea of a low threshold and high ceiling. Today there are open-ended powerful tools that have lower thresholds for children. While not yet obsolete, programming does have a narrower role than it did when Logo was first introduced.

I no longer see programming as an entry level computer activity for children. Word processing, computerized paint programs and hypermedia production are more accessible. As children begin to feel limited by these menu-driven applications, programming can be introduced as a meaningful solution to the student's real life problem.

I love working with Logo, as do many of the hundreds of children with whom I have worked. I have discovered, however, that there are many teachers and students who do not share that affection. It seems to me that our job as educators is to help these people find other paths to computer literacy.

I am interested in environments accessible to children that include a programming language for purposes of extending the environment, but do not require use of that language to cross the threshold. HyperCard, Hyperstudio, MicroWorlds and Control Lab all fit into that category. They all have reasonably powerful menu-driven capabilities while having an integrated programming language that expands those capabilities.

The HyperCard model provides a good example. At its less sophisticated levels, HyperCard is a very powerful, menu-driven, creative environment. When and if a student wants to create something that can't be accomplished with the commands available in the menu she can opt to learn something about scripting. This is needs-based teaching at its best. The student is not learning to script because it will improve her mind. She is learning it because she has an authentic need for it. From this need an emotional connection may form and that student may share a love of the elegance, challenge, and flexibility of programming that I share with Seymour Papert. ▲

Mark Steinberger, who has taught with Logo for many years, is the Computer Coordinator in Community School District 4 in New York City.

Turtle Math

by Douglas H. Clements and Julie Sarama Meredith

Can Logo help teach mathematics effectively? Research and classroom practice over the years provides an answer: "Yes, but, it takes some doing!" Fortunately, research has also provided some answers about what to do and how to do it.[1] We've used what we've learned to design a Logo environment fine-tuned for the learning of geometry and other mathematical topics. The result is *Turtle Math*, a version of Logo accompanied by activities geared to the upper elementary grades. We based *Turtle Math* on five principles we gleaned from research.

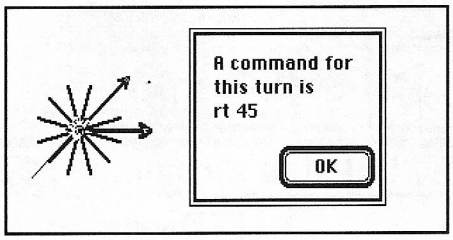
1. Encourage building mathematics from visual and intuitive knowledge

Research supports Papert's idea that Logo's turtle can encourage students to build mathematical knowledge out of their personal, visual, intuitive knowledge, such as knowledge of walking and moving. However, it also warns that children have a lot of difficulty with measurements. For instance, turns and angles can become stumbling stones for children trying to use turtle geometry meaningfully. To help, *Turtle Math* features a turtle that turns slowly and, optionally, a ray turns with it, showing the change in heading throughout the turn. *Turtle Math* also provides measurement tools; for example, an on-screen protractor, placed at the turtle's position and heading, measures turns. One arrowhead shows the turtle's heading. The other follows the cursor, which students move with the mouse. When they click the mouse, this arrowhead "freezes" and a turn command is displayed.

Another way *Turtle Math* supports the growth of mathematics from the visual lies in its overall structure, discussed next.

2. Maintain close ties between representations

In Logo programming, children need to make relationships between

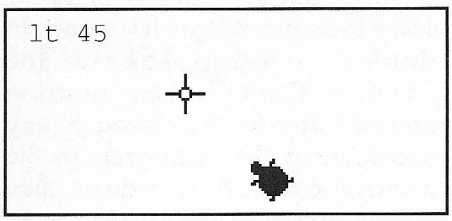


symbols (code) and drawings explicit. That's good, but they often lose these connections. *Turtle Math* encourages immediate mode programming – students type commands into the command center and see them immediately enacted. Students (especially novices) often prefer this exploratory mode and find it easier to make sense of tasks. Hoyles and Noss found that having students record their commands while working in immediate mode encouraged them to take a more global perspective on the task and to look for structure within their program design.[2] In contrast, use of procedures can lead to a separation between symbols and drawings. Further, students working in immediate mode are also more likely to abandon inappropriate solution strategies before they make incorrect generalizations. This keeps them on track. *Turtle Math*'s command center displays all the commands, in sequence, even as they work in immediate mode.

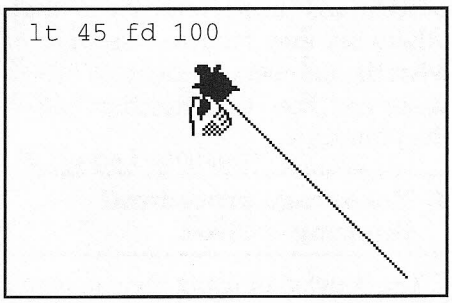
In *Turtle Math*, students enter commands in "immediate mode" in a command center. Any changes to commands are reflected automatically in the drawing. For example, if you change the **fd 40** to **fd 50**, the drawing automatically shows that change when you press <RETURN>. A tool (the first icon on the left) copies these commands into the teach window, applies a student-supplied name, and so defines the procedure.

The structure of the Command Center – long and narrow to the side of the graphics screen, instead of the traditional short but wide placement below this screen – permits the immediate inspection of more commands. This facilitates connecting symbols and drawings, as well as pattern searching. Further, students can easily modify the code. This encourages experimentation and supports later work with procedures. (See the illustration on the next page.)

The second basic issue is the direction of the symbol-drawing connection. One of Logo's main strengths has been its support of linkages between drawings and symbols. One of its limitations has been in the lack of two-way connection between these modes. That is, one creates or modifies symbolic code to produce visual drawings, but not the reverse. *Turtle Math* provides a "draw commands" tool that allows the student to use the mouse to turn and move the turtle, with corresponding Logo commands created automatically. The cursor changes to a crosshairs, the turtle continually turns to face it, and the corresponding turn command is dynamically updated in the Command Center.



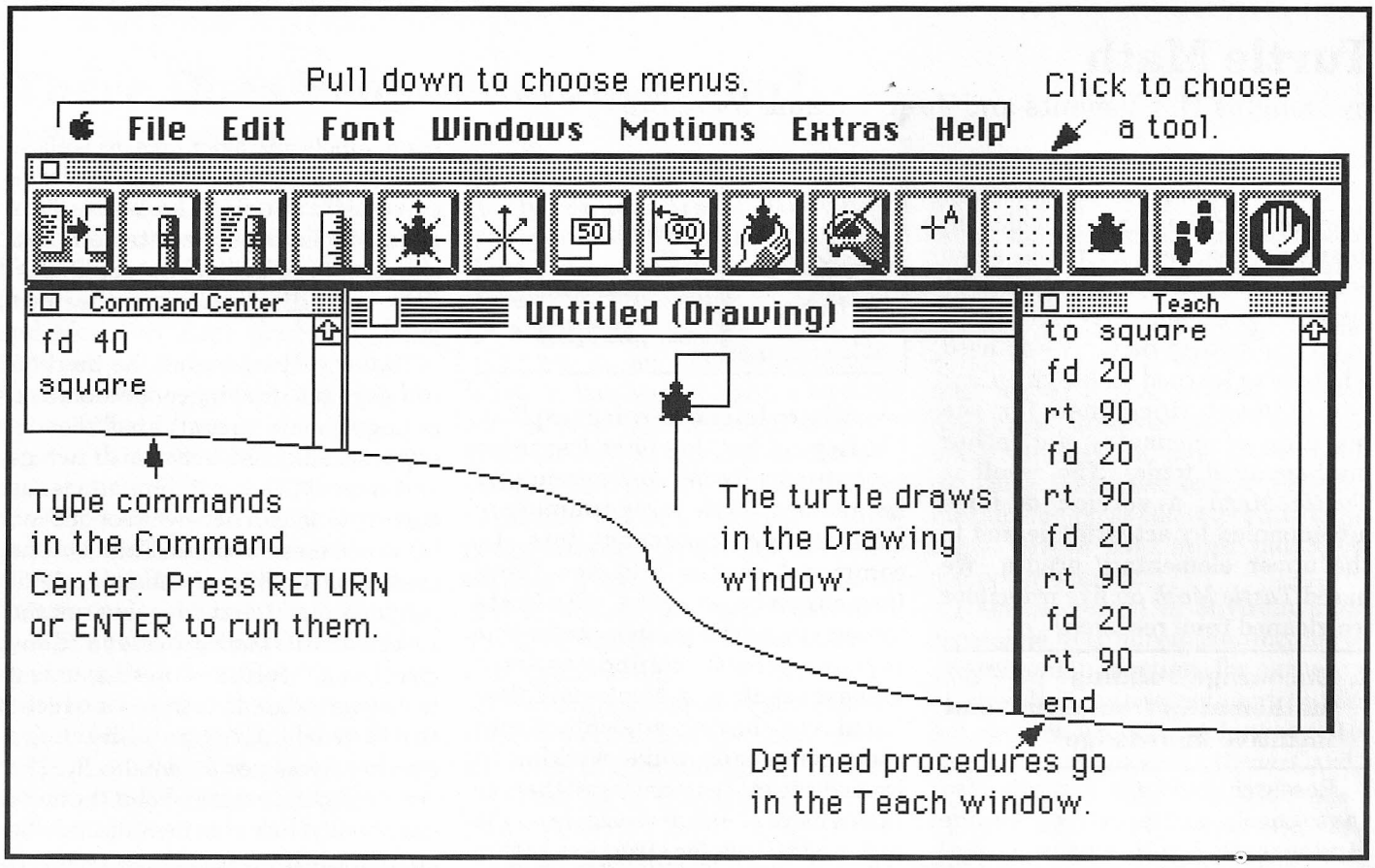
The turtle is then dragged forward or back and the corresponding **fd** or **bk** command is placed in the Command Center.



3. Facilitate examination and modification of code

Research indicates that there is little reason for students to move from visual-only to analytic approaches unless they are provided with support

(Continues on next page)



(Continued from page 9)

for such analysis. Several of *Turtle Math*'s tools provide such support. In addition to the structure of the Command Center, a step function allows children to "walk through" any procedure, or the commands in the command center. As they do so, they may edit any command in the sequence. The change is immediately reflected on the screen.

Students can also step through procedures with inputs. This is particularly important given that, otherwise, they tend to lose sight of what the numbers specified as inputs mean and how they function within the procedure.

4. Encourage procedural thinking

The benefits of using the command center and immediate mode programming have already been described. *Turtle Math*'s structure encourages use of procedures from the beginning. First, the teach tool walks students through the steps of defining procedures. Second, changes made to procedures within the teach window are immediately reflected on the graphics screen upon exiting that window.

5. Provide freedom within constraints

Our goal is to encourage problem solving within structured activities as well as student directed exploration, both within the constraints of the *Turtle Math* environment. We provide a set of activities, but they are not intended to be static. Teachers and students can invent and save their own.

Turtle geometry has long been a rich mathematical microworld. With *Turtle Math* we aim to extend and enhance it still further. ▲

1 Clements, D.H. and Meredith, J.S. *Research on Logo: Effects & Efficacy*, 1992, Logo Foundation (Use the Response Form on page 11 to order.)

2 Hoyles, C. and Noss, R., "Synthesizing mathematical conceptions and their formalization through the construction of a Logo-based school mathematics curriculum," 1987, *International Journal of Mathematics Education, Science, and Technology*, 18, 581-595.

Douglas H. Clements is Professor of Education and Julie Sarama Meredith is a Ph.D. candidate at State University of New York at Buffalo.

New Products

Crystal Rain Forest™, from **Terrapin Software** will be available for Macintosh computers in June. This adventure game teaches Logo throughout, and provides for learning about environmental topics. It incorporates **Crystal Logo**™, a simplified Logo, which may be used separately as well as within the game. Contact Terrapin at **800 972-8200**.

Turtle Math™, the program described on pages nine and ten of this issue, is available for Macintosh computers. For more information and a free demo disk contact **LCSI** at **800 321-5646**.

Also from LCSI, **MicroWorlds Project Builder**™ and **MicroWorlds Math Links**™ will be available for the MSDOS environment in June. An MSDOS version of **MicroWorlds Language Art**™ will follow later this year.

While They Last

During the summers of 1984, 1985, and 1986, Logo conferences were held at MIT. We have a few copies of the Logo '85 and Logo '86 Conference Proceedings, and Logo '86 Bibliographies. They're yours for the cost of shipping and handling. Use the Response Form below to order.

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

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 Response Form

❑ Enter my free subscription to *Logo Update*.
❑ Send me the full list of Logo Foundation publications.

Enter my order for	price	quantity	amount
❑ *Research on Logo: Effects & Efficacy*	$ 4.75	_____	$_____
❑ *Logo '85 Conference Proceedings*	$ 5.00	_____	$_____
❑ *Logo '86 Conference Proceedings*	$ 5.00	_____	$_____
❑ *Logo '86 Bibliography*	$ 5.00	_____	$_____
❑ *Mindstorms* (Second Edition) by Seymour Papert	$ 13.00	_____	$_____
❑ *The Children's Machine* by Seymour Papert	$ 22.50	_____	$_____
Tax-deductible contribution to the Logo Foundation			$_____

Total $ _____

Name _____

Organization _____

Address _____

City_____ **State**_____ **Zip**_____ -_____

Day Phone (**)**_____ **Evening Phone (** **)**_____

Please enclose payment or a school purchase order.

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

Summer Logo Courses

FLORIDA
Programming with Applications in Education: Logo 7/5 – 8/13
Contact: Dr. Joel Levine
Barry University
Miami Shores, Florida
305 899-3608

GEORGIA
Summerscape
July and August sessions
(For Students, grades 7 to 9)
Contact: Scott Shepherd
Georgia Institute of Technology
Atlanta, Georgia 30332
404 894-9544

ILLINOIS
Logo Course 7/11 – 8/5
Contact: Doris Dale
Southern Illinois University
at Carbondale
618 453-4245

MASSACHUSETTS
Boston Computer Museum
Computer Clubhouse
(For Children ages 10 to16)
Contact: Education Department
617 426-2800

MINNESOTA
Logo St. Paul 6/20 – 6/24
Contact: Logo Foundation
212 765-4918

NEW YORK
Logo Course 7/11 – 7/15
LEGO Logo 7/18 – 7/22
Contact: Joseph Sede
Long Island University
516 299-2199

LEGO Logo 7/11 – 7/15
Contact: SCOPE
Staff Development
516 360-0800 ext. 13, 21, or 25

OHIO
LogoWriter Course 6/13 – 6/24
Contact: George Shirk
SciMaTech
University of Toledo
Toledo, Ohio
419 537-2079

TEXAS
Logo in Your Curriculum
6/20 – 6/24, 7/25 – 7/29 or
8/8 – 8/12
Contact: Theresa Overall
Lamplighter School
Dallas, Texas
214 369-9201

If you are offering a Logo course or workshop, please let us know so that we may include the information in *Logo Update*.

Logo Foundation

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

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