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

What Is Advanced Logo?

In This Issue

What is "advanced Logo"? I asked Dorothy Fitch, Brian Harvey, Brian Silverman, Alberto Cañas, Sharon Yoder, and Mitchel Resnick. Their answers comprise the lead article of this issue of *Logo Update*. As far as I know, none of them discussed the question with each other, at least not specifically in connection with this article. Yet there are certain viewpoints that several of the authors share. Seymour Papert reflects on those themes in his commentary beginning on page 4.

If you are about to put this issue of *Logo Update* aside because "advanced Logo" is not for you, I suggest that you give it a chance. The various perspectives of the authors collectively convey a view of advanced Logo that is inclusive and not at all elitist or obscure.

Carol Sperry's Book Review continues that theme with a look at several advanced Logo books. Accompanying the review is a bibliography of books and papers suggested by Carol and by the other authors.

I puzzled for a while about how to sequence the six pieces that form the main article. Since my name begins with "T" I never liked the idea of listing authors in alphabetical order. In the end, I arranged them randomly. (Yes, of course I used a Logo procedure. It's on page 7, but why don't you try it for yourself before looking there?)

Michael Tempel

by Dorothy Fitch

When you are 10 years old, 15 seems old. When you are 20 years old, 30 seems old. When you are 30 years old, 45 seems old. "Old" is always older than you are. Age is a moving target. So is "Advanced Logo". When you first begin with Logo, the idea of writing procedures seems advanced. Then variables seem advanced. Then words and lists. Then interactive programming. Then recursion, the non-tail-end kind.

Advanced Logo is what you don't yet understand. It's something you have to work at, a hurdle, something new to your experience that grinds you to a halt. Learning new things can be frustrating, but it's exhilarating when you figure them out, understand them, and can use them.

It is easy to say that advanced Logo is a set of technically obscure primitives. These are the less intuitive programming tools, the fancier "bells and whistles" of the language. You are probably on your own as you learn these primitives, not generally covered in a typical first Logo course or book. Even experienced Logo users can find "advanced" primitives that they haven't fully explored or with which they don't feel totally comfortable (property lists, arrays, stream I/O, machine language interfacing, bitwise operations).

However, in a deeper sense, advanced Logo is not the language, but what you do with it. People sometimes call Terrapin and ask for our "easy" version of the language. There is no "easy" version. There is no "advanced" version. There are just "easy" or "advanced" things you can do with Logo, just as there are "easy" and "advanced" things you can do with a hammer, nails, and wood, or with a needle, thread, and fabric.

What makes a project "advanced?" It might require more complex problem-solving strategies. It might be a program with scores of procedures, creating many paths through it, more opportunities for bugs to creep in, and more testing routines. It may be an elegant but simple program that shows a full comprehension of recursion.

An "easy" Logo project can become an "advanced" project simply by changing the rules: "Do this using only five instructions." "Find a way to solve this recursively." "Do this without using the following primitives: . . ." "Solve this problem in three different ways." Sometimes advanced projects push you into learning primitives you might not have otherwise investigated, bringing together both ways of looking at "Advanced Logo".

Logo is a set of tools. You learn to use them at your own pace; you make with them what you want. And if your Logo contains the tools for the projects you want to complete, then that's all you need. Unlike most computer environments, Logo continually offers new challenges to those who seek them.

by Brian Harvey

This feels like a funny question to me, since the party line these days is that Logo is a tool to be used for studying other things, and if that's so, the emphasis should be on, let's say, advanced math rather than advanced Logo.

Nevertheless, there's a paradox in Logo-as-tool, namely that unless you study Logo itself, it's not a very transparent tool. Logo beginners can get trapped in small details of syntax, which make it hard to get past the mechanics of programming into the math or whatever else you're trying to learn about. If you say "I'm not interested in Logo itself," you never overcome this vulnerability to the syntax.

There are two ways to solve the problem of programming details. One is for some expert, a teacher or a Logo implementor, to build a simplified interface

(Continues on next page)

Logo Foundation
250 West 57th Street
New York, NY 10107-2603
Telephone: 212 765-4918
FAX: 212 765-4789
email: michaelt@media.mit.edu

Board of Directors
Seymour Papert, Chair
Clotilde Fonseca
Tessa R. Harvey
Geraldine Kozberg
Michael Tempel
Takayuki Tsuru

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.

(Continued from page 1)

that you can use to get at some set of ideas without really using the Logo language. The other, and here is the paradox, is to invest some effort into Logo as an object of study, so that the language truly does become a transparent means of access to ideas about other things.

So I'd say that one meaning of "advanced Logo" is reaching a state of mind in which the syntax rules feel sensible rather than arbitrary. Why do you put quotes in front of the variable name with **make** but a colon in front with **print**? A beginner will say, "Generally you use the colon for variables, but **make** is an exception." An advanced Logoite will say, "It has nothing to do with **make** or **print**, really. When you use **make**, you generally (although not necessarily) want to use the *name* of a variable as its input, and that's what the quote means. But when you use **print**, you usually (but again not necessarily) are using the *value* of the variable as its input, and that's what the colon means."

To me, "advanced Logo" also means being able to write a program that's too big to fit in your head all at once. Computer programming is pretty easy as long as you can remember every detail of your program. But as programs get larger, they get much harder to debug, because the result of this instruction depends on some leftover result from another instruction 300 lines away. Partly this kind of advancement comes from practice: from writing lots of programs. But there are ideas that programmers can learn to help control the complexity of their programs; the most important of these ideas is *abstraction*, which means extending the vocabulary of your programming language by inventing new features, new data types, and so on, and then using those extensions to make your program shorter and cleaner. (It was because BASIC didn't provide the language extension tools needed for abstraction that Logo people fought against it back in the early days of educational computing.)

Finally, of course, some of those extra features that help you write more complex programs are already in Logo, so "advanced Logo" also means learning about property lists, about procedure-as-data (**define** and **text**), about nonlocal exit (**catch** and **throw**), about parallelism in LCSI's MicroWorlds, about object-oriented programming in Paradigm Software's Object Logo, and so on.

by Brian Silverman

It's often hard to get a handle on what "advanced" means. I remember someone telling me a story about asking a bunch of children what they thought advanced mathematics were. The answer was usually something like doing sums. But it was sums of really big lists of really big numbers.

I suppose we could use reasoning like this to say that advanced Logo is writing really big programs filled with long, complicated, exotic commands. In fact, for some people, as the years go by, the programs that they write get bigger and sometimes they also get more complex.

I don't think that this has been the case for me. When I look at the programs that I've written for myself, they haven't grown much in either size or complexity. In fact they may have even become shorter and simpler as successive versions of Logo have let more and more things be increasingly concrete. (For instance, when LogoWriter came out I started **print**ing to the page to keep lists of results instead of using **sentence** to keep **list**s of results.) What's grown instead is the sophistication of the ideas I've let myself explore and the range of interests that I've mucked about with using the computer. Over the last couple of years the projects have usually had something to do with experimental mathematics or evolutionary biology or whatever else managed to catch my interest.

Using programming as part of an exploration requires a certain amount of fluency. Fluency in the sense of being able to create Logo programs while thinking about the exploration rather than thinking about Logo or the computer. This comes with practice, with looking at examples, with discussing "neat hacks" and tricks of the trade with friends and mentors, with essentially the same kind of learning strategies that you'd use to pick up any other non-trivial skill. The kind of fluency that lets you express yourself without thinking about the language is for me what advanced Logo really is.

by Alberto Cañas

Advanced Logo can refer to an advanced use of Logo or to an advanced version of the language. I propose that any truly advanced version of Logo must lead to an advanced use of the language.

The first idea that too often comes to mind when considering the phrase "advanced use of the language" is the complexity of the instructions used in programming: the more complex (or "advanced") the instructions, the more advanced the programming, and thus the more advanced the use of the language. But, is the "use of more advanced instructions" in a programming language equivalent to a more "advanced use" of the language?

We can consider two types of complexity: the complexity of a program and that of the problem to be solved. The complexity of a program is subjective; what appears

to be a complex program to one person may be simple to another. Experienced programmers, for example, analyze and recognize "chunks" of code instead of looking at individual instructions as novice programmers do, and therefore can more readily understand what a long program does. A deeper understanding of the problem being solved may also render the program less complex. It is thus inappropriate to talk about the advanced use of a language by discussing program complexity.

Complexity can also be attributed to problems. Although this complexity is also subjective, it is independent of the instructions used in solving the problem. A simple set of instructions may be an elegant solution to a very complex problem, while a complicated and obscure set of instructions may be an inefficient solution to a simple problem. Although there may be more than one "best" solution to a problem, any clean and elegant solution will employ the available features of the language in the most appropriate way. Therefore, the design of clean and elegant solutions to problems accurately reflects an advanced use of the language.

But clearly there will always be problems which could be solved more elegantly if the programming language offered a new construct which it does not currently contain. The features that Logo provides therefore have a direct effect on the quality of the solutions a user can design, and should thus be considered when talking about an Advanced Logo. However, just adding features to Logo does not make it a more Advanced Logo. In programming language design, the problem is not only to carefully select which constructs to add to the language, but more importantly, to carefully select what to leave out. Any new feature should allow the user to design an even cleaner and/or more elegant solution to the problem at hand. And such a feature should be usable over a wide of range of problems with varying degrees of complexity.

In summary, it is appropriate to speak of Advanced Logo from two perspectives. First, Advanced Logo may refer to using the existing features of Logo in such a manner as to provide clean and elegant solutions to an increasingly complex set of problems. Secondly, the term Advanced Logo may refer to the constructs which we add to the language in order to solve a wide range of problems in a cleaner and more elegant manner than previously considered possible.

by Sharon Yoder

As editor of the *Logo Exchange* I often encounter "advanced Logo." Potential authors write to tell me of the "advanced" work their students are doing. Others submit "advanced" articles for one column or another. And yet others describe some columnist's work as "advanced."

But what, indeed, is an advanced user of Logo? I often ask myself that question when I teach a Logo class. Some of my students seem destined to be "advanced" Logo users and others do not. Why? I believe the answer lies in an understanding of Logo that goes beyond the specific version being used. Let's examine some examples: An advanced user will try something like **cs** to clear the screen, and if that doesn't work, will try commands like **cg**, **cleartext**, **clean**, and **home**. An advanced user has enough understanding of different versions of Logo to try different possibilities. An advanced user is undaunted by differences in syntax. So if **if :a < :b then forward 10 else back 10** doesn't work, then variations such as **if :a < :b [forward 10] [back 10]** will be tried.

A combination of familiarity with more than one Logo dialect and understanding of the way in which Logo processes commands and operations allows advanced Logo users to make intelligent guesses as to the correct syntax. Further, an advanced Logo user deciphers error messages and uses the information contained in them. So for example, when typing **setpos 23 45** gives the message **setpos doesn't like 23 as input**, instead of repeated random trial and error, the advanced user

(Continues on next page)

About the Advanced Logo Authors

Dorothy Fitch is Director of Product Development at Terrapin Software. A former music teacher, she has also written several Logo tool packages including the *Logo Data Tool Kit* and *KinderLogo*.

Brian Harvey is Lecturer in Computer Science at the University of California at Berkeley. He also taught computer science for many years at the high school level and is author of the three volume *Computer Science Logo Style*.

Brian Silverman is President of Logo Computer Systems, Inc. where he has been responsible for the design and development of many versions of Logo including MicroWorlds, LogoWriter, and the *Phantom Fish Tank*.

Alberto Cañas is Assistant Professor of Computer Science at the University of West Florida. He has been involved with the introduction of Logo into schools in various Latin American countries while working with the IBM Latin American Education Research Center in Mexico. He currently leads Project Quorum, a partnership between IBM and the University of West Florida to establish a communications network among schools in Latin America.

Sharon Yoder is Editor of *Logo Exchange*. She has taught Logo at the high school and college levels for many years and is the author of several Logo books including *Introduction to Programming in Logo*. She currectly teaches computer education courses at the University of Oregon.

Mitchel Resnick is assistant professor at the Media Lab at MIT. He is one of the inventors of LEGO Logo, the creator of StarLogo, and has taught Logo to people of all ages for many years.

(Continued from page 3)

thinks about the possible meaning of the message and tries appropriate alternatives.

When beginners first learn Logo, they are very sensitive to the particular version and the syntax and semantics of that version. Unfortunately, too few users move beyond that level of understanding. Such users constantly ask for rules to translate from one version to another or become completely overwhelmed by minor differences in commands. As a teacher of Logo, my goal has always been to prepare students to be advanced users. I have always tried to get them to look beyond the particular version they were using to the structure of the language itself. I have always tried to expose them to several dialects of Logo. If you are a teacher of Logo, consider exposing your students to more than one version of Logo. Even if they only see a different version in class, such exposure will help prepare them for the inevitable day when they find a program they want to use in a dialect of Logo different from the one that they learned.

By Seymour Papert

When Michael Tempel asked me whether I would respond to responses to the question "What is advanced Logo?" I secretly hoped for a fight. But when the statements came, my initial disappointment at their non-belligerent tone soon gave way to an enjoyable sense of having acquired from them a more advanced sense of "advanced." And the opportunity to spend an hour with a fourth grade class in between writing my first draft and this draft allowed me to feel quite concretely how this acquisition could give rise to advancement in my skill as a Logo teacher. Thank you, Michael, for a very good idea.

Every teacher has (more or less consciously) developed strategies for responding to a set of intellectual statements. Among my own, the following were drawn out when I thought about writing this piece:

Classify: Make a taxonomy of the various positions.

Theorize the taxonomy: Make up explanations of why who says what.

Quibble: Even if you agree with most of what everyone says, exploring objections creates a tension that contributes to energizing intellectual work.

Appropriate: However many objections to an idea you have found, think about how you can use it.

Personalize: However much you like the language of the original author, recast it in another way.

(Continues on next page)

by Mitchel Resnick

What is meant by "advanced Logo"? Are we talking about "advanced" features in the language? Or "advanced" programs written in Logo? Or "advanced" project ideas? Or "advanced" ways of thinking encouraged by Logo?

And what do we mean by "advanced" anyway? If "advanced" means that only a small handful of people can do it, then I'm not very interested. I'd like "advanced Logo" to be more accessible, not less so. If we're going to advance, let's all advance together.

In my mind, advanced Logo should enable more people to make more things more easily, and it should engage more people with more powerful ideas in the process. In short, advanced Logo should encourage new ways of making and new ways of thinking.

The addition of multi-processing to Logo offers an example of how that's possible. In the computer science community, multi-processing is seen as an "advanced" feature. But children need it more than anyone (and many are shocked to find that most programming languages don't allow it!). When children make video games in Logo, they want to make many characters move at the same time. When they create an amusement park in LEGO/Logo, they want to make all of the rides move at the same time. Multi-processing makes that possible.

At the same time, multi-processing can encourage the development of new ways of thinking, new epistemological stances. When students see patterns in the world (like a flock of birds), they generally assume that someone or something is in charge (the "leader bird"). But as students play with multi-processing versions of Logo (such as LCSI's MicroWorlds, or my own StarLogo), they can begin to see the world in new ways. They can begin to understand how patterns can form through multiple interactions, without any centralized control.

Multi-processing, of course, is just one aspect of "advanced Logo." In what other ways can Logo become more "advanced"? My intuition is to look at other types of "multi." In traditional Logo activities, *one* student sits in front of *one* computer using *one* process to control *one* object (the turtle) in *one* medium (graphics.) Many new opportunities arise if we change each *one* to *many* – that is, if we develop Logo activities involving *many* users (communicating over networks), *many* objects (thousands of turtles, for example), *many* processes, and *many* media. And, of course, we should make sure to support and encourage *many* different thinking styles.

This multitude of multi-ness might seem overwhelming. Indeed, more is not always better. But my hunch is that these new forms of multi-ness could (in the appropriate contexts) make Logo activities more personal, more connected with children's interests, passions, and experiences. And, at the same time, they could help children develop new ways of looking at the world. ▲

(Continued from page 4)

Dorothy Fitch articulates most explicitly a relativistic position that could be expressed as: One Logoist's advanced is another's elementary.* She is suggesting that one should think about *advancing* as a process, a direction of movement, rather than *advanced* as a destination or terminal state. But whether one opts for continuous movement or for discontinuous jump, there is a further subclassification based on what it is that moves (or jumps).

Mitchel Resnick most cleanly opts for Logo as the moving object; his discussion of "advanced Logo" refers to a changed Logo rather than the same Logo used in different ways. Of course, given that his own work has been so largely concerned with creating new forms of Logo, it is not surprising that change in Logo would be most salient in his response. For the same reason one is not surprised to recognize in Brian Silverman's piece a similar emphasis, though in his case it is secondary to what he shares with Brian Harvey, Alberto Cañas, and Sharon Yoder who talk of the movement of the Logoist rather than of the Logo – movement *in* Logo rather than movement *of* Logo.

The advancing movements of Logoists are classified by all the commentators (though in subtly different ways) into two subclasses exemplified by (a) someone who uses the same set of Logo primitives, methods and ways of thinking to tackle more complex projects and (b) someone whose progress is seen not in the projects tackled, but in the way Logo is used to tackle them. Finally an important distinction (a sub-subclassification) within (b) is brought out by thinking (b1) of a Logoist coming to use more primitives and (b2) a Logoist coming to think differently about the primitives being used.

The last distinction is very relevant to a view of advanced vs. elementary that is far more strongly represented in thousands of classrooms than in the discussion of the sophisticated people writing here. Many teachers have been taught to think in terms like: elementary Logo is about graphics primitives; advanced Logo is about lists. This definition can't be taken quite literally, since a list is present in instructions used by beginners such as **repeat 4 [fd 50 rt 90]**. But a more modulated form of the distinction can be cast in terms of steps that mark a direction of advancing such as the following three stages. In stage 1 the **[...]** used in a **repeat** instruction is an unanalyzable cliché without any list-related meaning. In stage 2, understanding instructions like **repeat 4 readlist** shows that **repeat** is actually seen as a primitive with two inputs, a number and a list. Stage 3 is marked by being able to discuss reasons for choosing Logo's syntax, rather than simply using **repeat 4 fd 50 rt 90**. A major fallout for me of writing this column is a decision to think a lot about how to incorporate such discussions more explicitly in teaching Logo at all levels: Let's have more talk about why Logo is as it is and how it came to be so.

This resolution appears to place me in the taxon defined by advancing the Logoist rather than the Logo, and advancing in thinking about Logo rather than in what you do with it. But it is quibble time for the classification. Brian Harvey (whose reference **to make** and **quote** exactly parallels my **repeat** and **[ ]**) refers to the distinction as a paradox for the "party line" view that favors thinking of Logo as a tool. My sense of the party line is that indeed the way to introduce Logo is as a tool with lots of uses. But I see nothing paradoxical in a firm focus on the fact that all tools to a large extent, and this tool to a very large extent, cannot be used well without being understood. People often cite the hammer and the automobile as tools that don't have to be studied as objects. But this is because relevant knowledge is so embedded in our cultures that we don't recognize it as knowledge.

I'd take this a step further: we can use hammers so well because we know a lot about nails. Knowing about the tool and knowing about the use are not easily separated. I am prepared to quibble about the distinction between using Logo in a more advanced way versus using it for more advanced purposes – say mathematics. Maybe a central component of what makes mathematical thinking more "advanced" is exactly the same reflective attitude that makes Logo thinking more advanced.

From this point of view the most important criterion for judging the evolution of Logo might be whether the change advances the ways in which Logoists think about Logo. There are many allusions in the papers to how this might happen. An unexpected one was Sharon Yoder's observation that students with an advanced attitude enjoy comparing different versions of Logo – I'd add that the cause and effect here is a two-way street: playing with the differences also fosters the growth of the "advanced" way of thinking. A more deeply controversial case is Brian Silverman's reference to directions of development that favor greater concreteness in thinking.

A distinction between epistemological and instrumental criteria pervades Mitch Resnick's discussion of what is more advanced about new versions of Logo. Multi-processing is an advance because it allows more people to do more things. It is also an advance because it facilitates ways of thinking. I use it as a springboard to end with a question that will allow lots of quibbling.

It is obvious to me that Logo requires and facilitates ways of thinking about itself and about other stuff. But does it do these good things to one fundamental way of thinking

(Continues on next page)

* Two lingusitic observations: Brian Harvey says "Logoite" where I say "Logoist." I couldn't decide whether this reflects a different view of Logo users, a difference in taste in linguistics, or a mere "neutral" accident. The commentators are collectively shy about using a word for the opposite of advanced. Is this because elementary has (unfortunately) acquired a pejorative connotation? Or another neutral accident?

(Continued from page 5)

or to a multitude of them? Talking about movement *of* Logo, Mitch Resnick and Brian Silverman pick what looks like different features. But I see the abstract-concrete tension in the galaxy of possible mindsets as intimately related to the centralized-decentralized mindsets and both as related to hard-soft, hierarchical-heterarchical and a host of other slants on multiple ways of knowing that have become prominent (if not faddish) in the recent discussions of alternative espistemologies. Is it possible that on the deepest level there is *one* direction in which Logo facilitates thinking? I understand how this very question might appear to be a proof that I have not been able to break away from the "canonicalist" mindset – even in rejecting one canon I can't resist looking for another. But this may not be so bad if the canon is based on extending the principle with which Alberto Cañas ends his paper: the only really advanced Logo would be one that would allow the Logoist to reject anything and everything about it by building his own. ▲

Logo Tool Box

by Michael Tempel

In her article in this issue of *Logo Update*, Dorothy Fitch defines advanced Logo as "...what you don't yet understand." But not understanding enough Logo to write a certain procedure doesn't mean that you wouldn't be able to make good use of it if you had it.

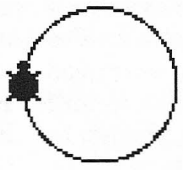
Young children may not understand how to write procedures to draw squares, triangles, or circles, but many Logo teachers give them such procedures as "tools" to enrich their drawings and give them greater satisfaction in what they can do with Logo. (And, they may also learn to recognize and name various geometric shapes.)

Tools can fill the gap between what you can make for yourself and what you need in order to accomplish your goals. If we look closely at what students are attempting to do as they work with Logo we can provide them with appropriate tools they need for their projects.

Recently, in Costa Rica, I watched a number of students work on projects that included drawings of "carretas." These brightly painted wooden oxcarts have large wheels that are decorated with geometric designs radiating from the hubs. A useful tool came to mind.

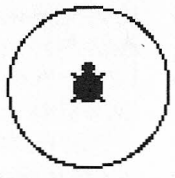
Most often, people use a Logo instruction such as this to draw a circle:

```
repeat 360 [fd 1 rt 1]
```



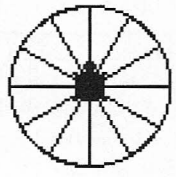
The turtle begins and ends at a point on the circumference of the circle. It's not so easy to find the center. Here's a procedure that draws a circle of a given radius, centered at the turtle's position:

```
to circle :radius
pu
fd :radius rt 90
pd
repeat 360 [fd (3.1416 * 2 * :radius) / 360  rt 1]
pu
lt 90 bk :radius
pd
end
```



After the circle is drawn the turtle is back at the center.
Now you can draw spokes or other designs that emanate from the hub.

```
circle 50
repeat 12 [fd 50 bk 50 rt 30]
```



You could use **circle** as a starting point for a program to draw pie charts. It's also good for drawing concentric circles.

But I thought of a problem. What if I draw a circle and then decide it's too big or too small? I'd like to be able to undraw it by drawing it again in penerase mode:

```
circle 40          oops!
pe
circle 40
```

But the second **circle 40** just draws a circle again because the **pu** and **pd** commands within the procedure override the **pe** command. Actually, these pen commands do not work the same way in all versions of Logo. (I was working in *LogoWriter*.) In *Terrapin Logo for the Macintosh* the pen's drawing mode is independent of whether it is up or down. The procedure I wrote actually does work either to draw or erase, depending on whether the last pen mode command was **penerase** or **penpaint**. In *Object Logo* I modified **circle** like this:

```
to circle :radius
make "pen penmode
pu
fd :radius rt 90
setpenmode :pen
repeat 360 [fd (3.1416 * 2 * :radius) / 360  rt 1]
pu
lt 90 bk :radius
setpenmode :pen
end
```

The procedure remembers the pen mode before doing anything else, restores it before circling, and again just before ending. In *LogoWriter* a second procedure is needed to erase circles:

```
to erase.circle :radius
pu
fd :radius rt 90
pe
repeat 360 [fd (3.1416 * 2 * :radius) / 360  rt 1]
pu
lt 90 bk :radius
pd
end
```

Shuffle

Here's the Logo program I wrote to randomize the order of the six authors of the advanced Logo articles in this issue of *Logo Update*. Before I began working in Logo, I thought about several models of shuffling a deck of cards. Split the deck into two piles and then mix them back together. Throw them up in the air and then pick them up after they land. I settled on a simple model that I felt would work well in Logo: Pull a single card from some random place in the deck and put it on top. This move, in itself, doesn't mix things up very much, but repeating it a few times does the trick.

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

to move.to.top :item :stack
output sentence :item (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

Print shuffle shuffle shuffle shuffle shuffle [Cañas Fitch Harvey Resnick Silverman
Yoder]
Fitch Harvey Silverman Cañas Yoder Resnick
```

There are many ways to shuffle. How would you do it? ▲

Navajo textile design created using **MicroWorlds Math Links**

Learn More About MicroWorlds Project Builder

On March 12, 1994 LCSI is offering a free MicroWorlds Project Builder™ workshop in St. Louis, MO. Call for details.

LCSI 800 321-5646

Logo Users Groups

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

Los Angeles Logo Users Group
January 25 • March 15 • May 24
Time: 4:00pm–7:00pm
Campbell Hall
4533 Laurel Canyon Blvd.
North Hollywood CA 91607
Contact: Carolina Goodman
213 980-7280 ext. 234

New York Logo Users Group
February 8 • April 12 • June 7
Contact: The Logo Foundation
212 765-4918

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

NEW PRODUCTS

The Logo Foundation is offering several new publications. You may use the response form on page 11 to order them.

A Full-Screen LogoWriter Printshape Procedure by Tom Trocco allows large display and printing of turtle shapes.

A LogoWriter Ecology Simulation by Tom Trocco explores interactions between plants and animals in a pond.

Headlight Stories is a collection of writings by teachers at the Hennigan School in Boston about their experiences during eight years of Project Headlight.

MicroWorlds Math Links™, the third product in the MircoWorlds series, has been released by **Logo Computer Systems, Inc.** It includes the same new version of Logo as **MicroWorlds Project Builder ™** and **MicroWorlds Language Art ™** along with more than two dozen activities with polygons, patterns, permutations and combinations, and transformations. Call LCSI at **800 321-5646** for more information and a free demo disk.

Object Logo 2.70 ™ has been released by **Paradigm Software**. Free to current **Object Logo** users, this upgrade includes 32-bit addressing, a new compiler, and a new applications generator. For more information contact Paradigm Software at **617 576-7675**.

SoftEast Corporation has announced the release of **WIN-LOGO for Windows™**. This new version incorporates the features of the earlier MSDOS-based **WIN-LOGO**, but takes full advantage of the MicroSoft Windows environment. Also available from SoftEast is the **WIN-LOGO/ LASY Robotics System**, a construction kit consisting of plates, blocks, gears, wheels, sensors, switches, lights, motors, and other elements. Electro-mechanical models may be controlled using **WIN-LOGO**. For more information call SoftEast at **508 897-3172**.

LEGO Dacta has announced the spring 1994 availability of **Control System** which combines **LEGO®** elements, sensors, lights, and motors with the same computer interface and Logo-based software as the recently released **Control Lab™**. Science and math curriculum materials for the upper elementary grades are provided with **Control System** . A Macintosh or MSDOS computer is required. Contact LEGO Dacta at **800 527-8339**.

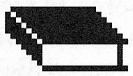
Crystal Rain Forest™ has been introduced by **Terrapin Software**. This adventure game teaches Logo throughout, and provides for learning about environmental topics. It incorporates **Crystal Logo™**, a simplified Logo, which may be used separately as well as within the game. **Crystal Rain Forest** won the British Educational Technology Gold Award in 1993 for Best Primary Software. Contact Terrapin at **800 972-8200**.

The International Society for Technology in Education has announced the publication of a series of three books by Gary Flewelling:

Math Activities Using LogoWriter
-Patterns and Designs
-Investigations
-Numbers and Operations

Each volume includes lessons, worksheets, and disk-based activities. For more information contact ISTE at **800 336-5191**. ▲

Book Review



by Carol Sperry

Computer Science Logo Style:
Volume 1: Intermediate Programming
Volume 2: Projects, Styles, & Techniques
Volume 3: Advanced Topics
by Brian Harvey, MIT Press, Cambridge MA, 1985, 1986, 1987

Exploring Language with Logo
by E. Paul Goldenberg and Wallace Feurzeig, MIT Press, Cambridge MA, 1987

Visual Modeling with Logo
by James Clayson, MIT Press, Cambridge MA, 1988

I'm going out a bit on a limb with this review for *Logo Update* and admit that I've had to quash some intellectual insecurities to broach the topic of advanced Logo. However, it was some of these same intellectual insecurities that kept me, for some time, from some of the pleasures that can be found in this part of the Logo landscape.

There are many different answers to the question, "What is advanced Logo?", some of which appear in this issue's pages of *Logo Update*. Brian Silverman, for example, supports doing advanced Logo projects using simple Logo commands and procedures. Brian Harvey approaches the subject from a computer science perspective. But what does the average teacher think about advanced Logo and does she think she can ever get to that phase?

I was a teacher for many years in the New York City public school system, and then a teacher of teachers in various places. I have found that the average teacher often gets too bogged down in the day-to-day of her practice and is sometimes uneasy about getting involved in the so-called advanced Logo books. 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 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. I have wished from time to time that these specialized books could go through some kind of interpreter so that generalist teachers could take advantage of the powerful ideas they contain. But then I wonder what would be lost in translation and how important a part of the endeavor is the challenge to understand.

Brian Harvey's trilogy *Computer Science Logo Style* offers a rich environment of ideas. I stuck for the longest time with *Volume 2: Projects, Styles, and Techniques*. Though described as suitable for the intermediate learner, I found I could follow the procedures, use them as models, and go on from there. Harvey wrote the book in a case study format and created each project for the enjoyment of it, "not because it fit some subtle pedagogic purpose."

Project topics include cryptography, games, mathematics, programming utilities, and pattern matching. Harvey's discussions of the projects are broad and once I was involved, his style helped me feel as though I were in conversation with the author. He asks questions and then takes us through the process of the answers, giving us detailed narratives of his own explorations of the projects. The two other books in this series are *Intermediate Programming*, which introduces Logo from a computer science point of view, and *Advanced Topics*, which introduces some of the elements of university-level computer science, still in the context of Logo programming.

In 1987, E. Paul Goldenberg and Wallace Feurzeig published *Exploring Language with Logo*. I was excited about this since I consider myself more of a language person than a math type. Once again, I felt somewhat hesitant as I looked into the book since it seemed more like a romp through linguistics and called itself a "scientific approach into English." The effort, however, was well worth it since it's a playful romp through linguistic notation and fulfills its authors' humblest wishes that we at least find the book " 'merely' enjoyable, interesting, and stimulating." Using Logo programs as the notational system for theories about language allows us to make simple hypotheses, try them out, and make changes easily as our findings grow more and more complex. As stated in the introduction of the book, "The chance to experiment — to try out ideas concretely and see how they work . . . fundamentally changes the nature of linguistic exploration. The enterprise . . . becomes more dynamic, and this makes it more accessible to more people." A great feature of *Exploring Language* is experiencing, through their writing, the authors' palpable love of language in all its complexities.

Finally, I'd like to say a few words about *Visual Modeling with Logo* by James Clayson. This is a delightful exploration into problem solving through visualization. Clayson found that "visualizers seemed . . . less intimidated by vagueness because their picture-making abilities gave them concrete starting points, and they seemed to enjoy playing around with the painted pieces of complex problems." He wondered if "their visual play encouraged them to see where more analytic approaches might usefully be applied." The book is an outgrowth of the courses he devised to teach visual thinking to students who had problems doing it naturally.

Clayson's style also invites you in, to discover with him, and then to explore on your own. Like the books mentioned above, this is not a book about Logo, but one which used Logo to do something else. *Visual Modeling* shows us a fascinating array of designs including fractals, Islamic patterns, considerations of objects in

(Continues on next page)

(Continued from page 9)

space, and using variously shaped grids. Clayson welcomes us to "copy the ideas of any procedure" and then to "play rough with them," giving "the copied procedure funny and outlandish arguments." Ideas are used from the worlds of design, aesthetics, math, art, science, and even psychology. The wide range of exercises allows us a rather thorough personal investigation of the many aspects there are to the act of seeing.

The books mentioned here are just a few of the broad range there is to choose from. (You'll find more titles listed in the box below.)

Of course, there is the question of Logo fluency; to do anything else while using Logo, you have to know Logo. However, I have found that, bit by bit, my repertoire of Logo know-how was greatly increased each time I ventured into other realms of expertise. I was often surprised to find that seemingly unrelated Logo information cleared up other areas of programming I was having trouble with. If you are suffering from the same lack of confidence that I had, I hope these few paragraphs will encourage you to plunge in and not deprive yourself of the pleasures of so-called advanced Logo adventures. ▲

Advanced Logo Bibliography

LOGO BOOKS

Abelson, Harold, and diSessa, Andrea, *Turtle Geometry,* MIT Press, 1981. A well-known classic, with a very advanced look at the mathematics behind the graphics.

Birch, Alison, *The Logo Project Book,* Terrapin Software, Inc., 1986. Logo projects with words and lists.

Boecker, Heinz-Dieter, Eden, Hal, and Fischer, Gerhard, *Interactive Problem Solving Using Logo,* Lawrence Erlbaum, 1991. Case studies of interactive problem solving in fields such as mathematics, artificial intelligence, and linguistics.

Clayson, James, *Visual Modeling with Logo* MIT Press, 1988. Geometric activities such as tiling, with detailed attention to actual patterns used in different cultures.

Cuoco, Albert, *Investigations in Algebra,* MIT Press, 1990. Examples mainly from combinatorics and number theory used to develop ideas in abstract algebra.

Friendly, Michael, *Advanced Logo, A Language for Learning,* Lawrence Erlbaum, 1988. Logo as a "real" programming language and as an educational methodology.

Goldenberg, E. Paul, and Feurzeig, Wallace, *Exploring Language with Logo,* MIT Press, 1987. Linguistics topics, exploring the structure of large units (poems) down to small units (sound and spelling of words.)

Harvey, Brian, *Computer Science Logo Style* A three-volume series: 1. Intermediate Programming 2. Projects, Styles, and Techniques 3. Advanced Topics MIT Press, 1985, 1986, 1987. The first volume is a Logo programming text with the emphasis on list processing. The second is a collection of ten programming projects with commentary. The third is the first week of six college courses, from automata theory, through compilers, to artificial intelligence.

Hoyles, Celia, and Noss, Richard, *Learning Mathematics and Logo,* MIT Press, 1992. A collection of research papers about various aspects of the use of Logo in math classrooms.

Lewis, Philip G., *Approaching Precalculus Mathematics Discretely* , MIT Press, 1990. Vectors, transformations on the plane, graphs of functions, limits, and other such topics.

Resnick, Mitchel, *Beyond the Centralized Mindset: Explorations in Massively-Parallel Microworlds,* MIT Press, 1994 (expected availability by fall 1994.) The ideas underlying StarLogo, in which thousands of turtles can move and interact in parallel, and how it encourages thinking about decentralized phenomena.

Silverman, Brian, *Phantom Fish Tank,* LCSI, 1987. An exploration of cellular automata, this is John Conway's Game of Life carried to its ultimate conclusion. Includes a specialized version of Logo.

Solomon, Cynthia, *Computer Environments for Children* MIT Press, 1985. A research document describing and comparing four different approaches to the use of computers in education, including Logo.

COMPUTER SCIENCE LOGO STYLE WITHOUT LOGO

Abelson, Hal and Sussman, Gerald, *Structure and Interpretation of Computer Programs,* MIT Press, 1984

Harvey, Brian and Wright, Matthew, *Simply Scheme: Introducing Computer Science,* MIT Press, 1994

PAPERS

Silverman, Brian and Tempel, Michael *Fuzzy Logo,* Logo Foundation 1985. Introducing randomness into the world of turtle geometry opens the door to explorations of feedback, probability, and statistics.

Silverman, Brian and Tempel, Michael *Creating a Logo Tool Box,* Logo Foundation 1988. Learning more about Logo while writing Logo procedures for the primitives you don't have.

Tempel, Michael, *Conversations with Logo,* Logo Foundation, 1989. Logo answers some questions about how she works.

Tempel, Michael, *Easy as 11223,* Logo Foundation, 1988. An exploration in number theory using Logo.

Logo Foundation Response Form

❑ Enter my free subscription to *Logo Update*.

❑ Send me the complete list of Logo Foundation Publications.

Enter my order for:	quantity	amount
❑ Turtle Geometry $37.50(HB) $19.95(PB)	____	$_____
❑ The Logo Project Book ❑ PC Logo ❑ Logo Plus for the Apple II $29.95	____	$_____
❑ Interactive Problem Solving Using Logo $79.95(HB) $39.95(PB)	____	$_____
❑ Visual Modeling with Logo $19.95	____	$_____
❑ Investigations in Algebra $47.50(HB) $29.95(PB)	____	$_____
❑ Advanced Logo $89.95(HB) $39.95(PB)	____	$_____
❑ Advanced Logo Disk $29.95 ❑ LCSI Logo II ❑ IBM Logo	____	$_____
❑ Exploring Language with Logo $21.95	____	$_____
❑ Exploring Language with Logo Disk $15.95 ❑ Apple Logo ❑ Mac Logo ❑ IBM Logo ❑ Terrapin Logo	____	$_____
❑ Computer Science Logo Style, Volume 1 $22.95	____	$_____
❑ Computer Science Logo Style, Volume 2 $21.95	____	$_____
❑ Computer Science Logo Style, Volume 3		(temporarily out of print)
❑ Computer Science Logo Style Disk $.9.95 ❑ LCSI Logo II ❑ Mac Logo ❑ IBM Logo	____	$_____
❑ Approaching Precalculus Mathematics Discretely $47.50(HB) $29.95(PB)	____	$_____
❑ Phantom Fish Tank $19.95 ❑ Apple 5.25 ❑ Apple 3.5 ❑ MSDOS 5.25 ❑ MSDOS 3.5	____	$_____
❑ Computer Environments for Children $13.95	____	$_____
❑ Structure and Interpretation of Computer Programs $55.00	____	$_____
❑ Simply Scheme: Introducing Computer Science $49.95	____	$_____
❑ Simply Scheme: Introducing Computer Science Disk $10.00 ❑ MSDOS ❑ Macintosh	____	$_____
❑ The Children's Machine by Seymour Papert $22.50	____	$_____
❑ Mindstorms (2nd Edition) by Seymour Papert $13.00	____	$_____
❑ Headlight Stories $15.00	____	$_____
❑ Fuzzy Logo $3.50	____	$_____
❑ Creating a Logo Tool Box $4.00	____	$_____
❑ Conversations with Logo $4.00	____	$_____
❑ Easy as 11223 $2.50	____	$_____
❑ A LogoWriter Ecology Simulation $10.00	____	$_____
❑ A Full Screen LogoWriter Printshape Procedure $3.50	____	$_____
Tax deductible contribution to the Logo Foundation		$_____
Total		$_____

Name _____

Organization _____

Address _____

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

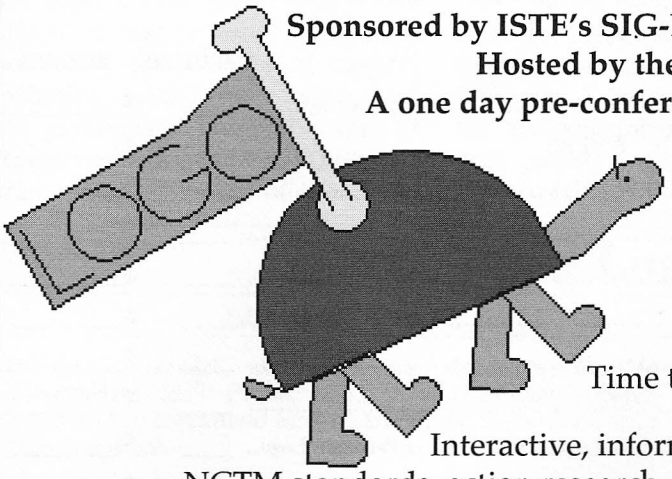
Day Phone ( )_____ **Evening Phone ( )**_____

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.

Logosium (Logo Symposium)
Sunday, June 12, 1994

Sponsored by ISTE's SIG-Logo and the Logo Foundation
Hosted by the MIT Media Lab
A one day pre-conference at NECC '94, Boston



Mini hands-on sessions

Jitterbug Logo Style — it's procedural!

Students, student projects, and video tapes

Time to schmooz with Logo users from around the world

Interactive, informal group discussions on Logo topics – assessment, NCTM standards, action research, new environments, beginning and advanced topics

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

Logo Foundation

250 West 57th Street • New York, NY 10107-2603

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