WINTER 1997

Volume 16 / Number 2

Logo Exchange

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

LOGO RENAISSANCE

INSIDE

Peter Reynolds 97

Logo Exchange

Volume 16 / Number 2

Logo Exchange is published quarterly by the International Society for Technology in Education Special Interest Group for Logo-Using Educators. *Logo Exchange* solicits articles on all aspects of Logo use in education.

Submission of Manuscripts

Manuscripts should be sent by surface mail on a 3.5-inch disk (where possible). Preferred format is Microsoft Word for the Macintosh. ASCII files in either Macintosh or DOS format are also welcome. Submissions may also be made by electronic mail. Where possible, graphics should be submitted electronically. Please include electronic copy, either on disk (preferred) or by electronic mail, with paper submissions. Paper submissions may be submitted for review if electronic copies are supplied on acceptance.

Send surface mail to:

Gary S. Stager
21825 Barbara St.
Torrance, CA 90503 USA

Send electronic mail to:
logoexchange@moon.pepperdine.edu

Deadlines

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

Vol. 16, No. 2	June 1, 1997
Vol. 16, No. 3	Oct. 1, 1997
Vol. 16, No. 4	Jan. 1, 1998
Vol. 17, No. 1	Mar. 1, 1998

Logo Exchange is published quarterly by the International Society for Technology in Education (ISTE), 1787 Agate St., Eugene, OR 97403-1923, USA; 800/336-5191.

ISTE members may join SIG/Logo for $24. Dues include a subscription *to Logo Exchange*. Non ISTE member subscription rate is $34. Add $10 for mailing outside the USA. Send membership dues to ISTE. Add $4.00 for processing if payment does not accompany your dues. VISA, MasterCard, and Discover accepted.

Advertising space in *Logo Exchange* is limited. Please contact ISTE's director of advertising services for space availability and details.

Logo Exchange solicits articles on all topics of interest to Logo-using educators. Submission guidelines can be obtained by contacting the editor. Opinions expressed in this publication are those of the authors and do not necessarily represent or reflect official ISTE policy.

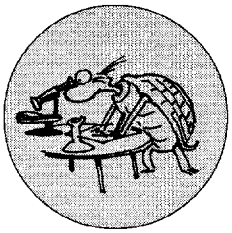
POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 480 Charnelton St., Eugene, OR 97401-2626 USA. Periodicals postage paid at Eugene, OR. USPS# 660-130. ISTE is a nonprofit organization with its main offices housed at the University of Oregon. ISSN# 0888-6970

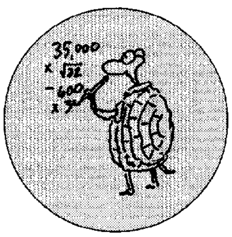
This publication was produced using Aldus PageMaker®.

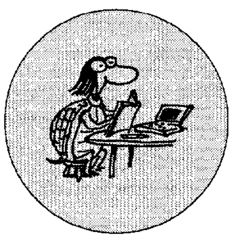
Logo Exchange

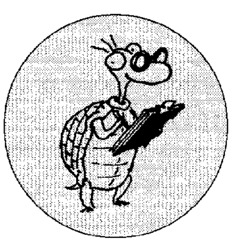
Winter 1997 Vol. 16/ No. 2

Contents









ARTICLES

COLUMNS

ISTE

EDITORIAL / GARY S. STAGER

Hard Fun

In *The Childrens' Machine*, Seymour Papert tells the story of one kindergarten telling student another that Logo is **hard** fun. Papert goes on to use **hard** fun as a metaphor for thinking about the deepest learning which occurs when passion, challenge, context, and purpose are all present. Any educator who has tried to make Logo or Logo-like learning a staple of the school day understands how **hard** and fun this process can be.

Logo Exchange is thrilled to publish a provocative speech by Dr. Seymour Papert. The "Father of Logo" has spent decades challenging our thinking and inventing learning tools used by children around the globe. His unbridled optimism and advocacy for children has been tireless and infectious. Although Dr. Papert is arguably one of the great minds of the 20th century, his ideas are too often marginalized. In the intellectual-free zone of educational computing his ideas are not debated or even disputed—merely ignored. *Logo Exchange* is one of the few educational publications to review his groundbreaking books, *The Childrens' Machine* and *The Connected Family*.

Papert's speech reminds us that Logo was always about more than just turtles. Since the 1960s, Dr. Papert has viewed school's ability to embrace increasingly powerful personal computers as a measure of the institution's willingness to respect and support student learning.

While the availability and power of personal computers begin to make possible the ideas Cynthia Solomon and Seymour Papert described in *20 Things to do with a Computer* (1971), many schools react with increasingly authoritarian policies and trivial computer usage. In the hands of many schools the democratizing power of the Internet is used to cover the curriculum. New laptop computers are being advertised for their ability to deliver disembodied content and take student attendance—not as powerful vehicles for learning and self expression.

Those of us still healing from the psychic scars left by our own schooling are thrilled to assist Logo's creators in realizing their visions of constructionism and personal computing. Perhaps Papert's greatest achievement was creating a transitional object (Logo) with which adults charged with the care of children could feel creative and intellectually powerful. Logo is great **fun**. I can't imagine what my life would be like without the joy of learning and teaching with Logo.

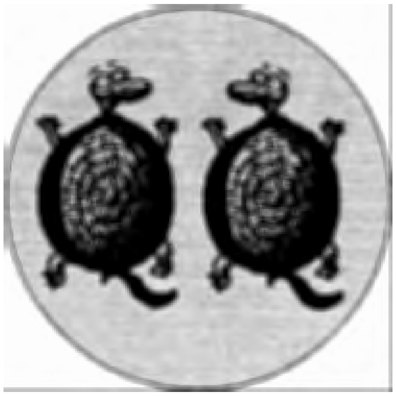
I am reminded of how much **fun** learning and teaching can be when I visit Cathie Galas' class in which 8- to 11-year-old kids create sensational Logo simulations of the brain or when I help Wesley College children program their own virtual pets. Watching graduate students and veteran teachers giggle with joy while struggling to solve a turtle graphics challenge reminds me that Logo is a powerful object to think with.

Bob Tinker makes the case that the availability of low-cost ubiquitous computing will support a reemergence of Logo. This will occur only if schools begin to trust teachers and support their desire to create rich personal learning environments for kids.

About a decade ago the citizens of Costa Rica made such a **hard** decision. They decided to leap-frog the instructionist technologies of 19th/20th century schooling and embrace the **hard fun** of Logo. They invested a great deal in professional development and in supporting the dreams of their teachers. Along the way, these educational pioneers had a great deal of **fun** and their work has inspired educators around the world. Well, it seems that their hard work and commitment to excellence is beginning to pay off. The Intel Corporation has decided to build its new $500 million chip plant in Costa Rica. Intel chose Costa Rica instead of 10 other countries because of its "well-educated, highly-skilled, highly-motivated labor pool" and "the skills of the Costa Ricans are only going to grow as a result of a MIT media program that has already placed computers in 30 percent of Costa Rica's primary schools."[1] Congratulations are certainly well deserved for our Costa Rican Logo-using colleagues.

Many of you know how **hard** it is to "shoehorn" Logo into an overflowing curriculum or justify open-ended projects in an age of national testing. In spite of the obstacles, *Logo Exchange* readers work **hard** to make schools wonderful places for children. You know how **hard** it is to sustain Logo and constructionism in increasingly

See HARD FUN (Page 21)



Home Work

I have always enjoyed the thinking that different Logo commands sometimes stimulate. Recently, I was working on a Logo project and things were not going well. I had moved the turtle around quite a bit and had lost track of things. What to do?

For the first time in a long while, I used the **home** command. Bam! In a twinkling, the turtle blipped to the middle of the screen, ready to start over. Cool!

As I began anew on the project, I also found myself thinking about the home command. It had rescued me from a confusing and somewhat complicated situation. It had returned the turtle to its starting point and its initial heading. In a way, it was revolutionary, because it gave me a fresh start on the problem I was trying to solve.

Suddenly, my mind was flooded with aspects of home.

Remember ever being in the home stretch, and then arriving after a long trip? The exclamation "I'm home!" has been music to many an anxious parental ear. "Home sweet home" reminds us of our hedge against the bitterness of life, a place where they have to take you in if you show up.

Got any of that good ol' home brew? Well, one mix developed into a pretty good concoction not too long ago, when two Steves had members of their Homebrew Computer Club help them develop the first Apple home computer. Let's follow that line a bit further.

Ever hear of a home truth? This is usually defined as a fundamental or key truth, especially one that is somewhat unwelcome or that is uncomfortable to acknowledge. Stay with me now.

The July 1997 issue of *The Atlantic Monthly* magazine carried a cover article titled "The Computer Delusion," in which Todd Oppenheimer described how many school districts are cutting entire programs from their budgets to free up funds to purchase computers. The problem was the home truth: Oppenheimer had difficulty finding evidence that computer use improved teaching and learning significantly in the schools.

He mentioned a few shining examples, such as Dennis Frezzo at Thurgood Marshall Academic High School in San Francisco using computers and an "advanced Lego engineering kit" (presumably LEGO® TC logo) to challenge his students to solve realistic problems, and an elementary school using computers to assist children with disabilities. Quoting from books such as *Endangered Minds: Why Children Don't Think and What We Can Do About It, Silicon Snake Oil: Second Thoughts on the Information Highway*, and *The End of Education*, however, Oppenheimer sketched out a broadly pessimistic picture and suggested that perhaps priorities have been misplaced when it came to procuring more computers for schools. This is certainly some home-cooking food for thought.

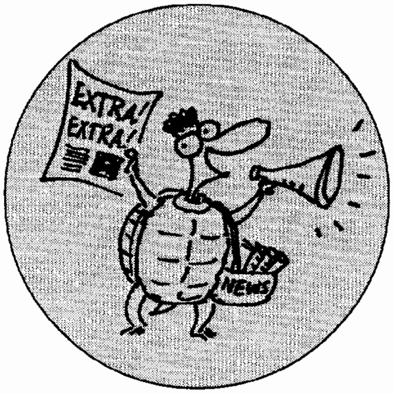
Anything else for home? How about a home remedy? In a recent lament about the sorry state of learning with technology and computers in the schools, Seymour Papert suggested that "the locus of innovation in thinking about learning is moving rapidly out of the school and into the home." Home education and home learning with a home computer have the potential to create a force for change that schools might not be able to ignore. If children (and parents) begin doing things at home with a computer that are more interesting, more stimulating, and more challenging than at school, won't the schools have to respond? Hmmm. This could get interesting, folks.

Like the home command, have we now come full circle in education, back to where education started in the first place — in the home? Perhaps we'll soon see if that is so. In the meantime, let's keep the home fires burning — and don't forget to do our homework!

FD 100!

PS: Speaking of home, my family and I have blipped to a new one! 105 Hickory Drive, Murray, KY 42071, to be exact. My new professional home is at Murray State University, teaching science education and working with nearby schools.

Tom Lough, Founding Editor
70020.223@compuserve.com



Logo News

EuroLogo 1997, Budapest, Hungary

August 20-23, 1997

by Dr. Brian Harvey,
U.C. Berkeley
bh@CS.Berkeley.EDU

Here are my impressions from EuroLogo. I wasn't in every session, so this isn't an authoritative report! There was some tension between the original idea of Logo as a tool for learning mathematics (that is, for learning how to be a mathematician, which may or may not be connected with school math) and the newly influential school subject of Informatics, into which a lot of recent Logo-related effort is going in Europe. A plenary session with Celia Hoyles and Richard Noss (UK) on the first day included a strong reaffirmation of mathematics as the goal. Mike Doyle (UK), who ran the last EuroLogo in Birmingham, strongly argued the intellectual case for the computer as a new thing that shouldn't be understood in terms of old topics. Many more pragmatic presentations were about using Logo in various countries as part of the national Informatics curriculum. The long-standing tension between reform and revolution is still in evidence, although reform seems to be winning. That is, most people are talking about using Logo in the context of public schooling (in the American sense, not the British sense), not about creating alternative institutions.

In contrast, what struck me most strongly was something outside of the conference itself, as I walked around Budapest. The kids there are completely unafraid! Maybe I'm romanticizing because I don't speak the language and could only judge by body gestures, but I had a strong sense that kids don't look over their shoulders expecting to be told to stop doing something. At the conference, I did a workshop about logic puzzles in which most of the participants were around junior high age. Because of language problems, I wasn't able to give them as much one-on-one help as I would have liked, so a lot of them were stalled on small notational issues a lot of the time, but they all maintained a cheerful demeanor and apparent interest.

This experience suggests to me that reform instead of revolution might be okay in Europe, or at least in Budapest. Maybe kids' lives are basically okay there, and maybe the schools really are pretty much okay places. I certainly got the impression that Hungarian adults actually like kids.

I was excited by Comenius Logo, a commercial product called by various other names in assorted countries. It's too bad it's commercial, but it combines real support for multimedia glitz (e.g., movies and melodies are first-class objects; they can be members of lists, arguments to procedures, and so on) with real support for debugging (pause, trace, and the like). MSWLogo, a free extension of Berkeley Logo, has a similar spirit; both are widely used in Europe. A team in Greece has made another extension to Berkeley Logo to allow dynamic extension with new sets of primitives. For example, the turtle commands are removed from the core language and are just one of several loadable sets. They showed a "variation" tool that lets you track graphically the effect of changing a parameter to a procedure with sliders.

Disclaimer: I'm a software developer and computer hacker; a classroom teacher probably would give you a completely different report!

—edited by Jeff Richardson

Learning In a Multimedia World

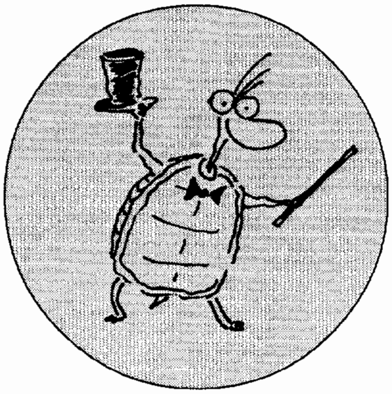
A Residential Workshop for Computer-Using Educators, Presented by Gary Stager and Fred D'Ignazio

by Brenden Morris
St. Paul's Anglican
Grammar School
Warragul, Victoria, Australia
bmorris@sympac.com.au

Multimedia and the multimedia-ready version of MicroWorlds 2.0 offer an ideal environment for students to explore learning in a collaborative, constructivist, student-driven, and opened-ended way. The workshop presented at Avalon in Victoria, Australia, demonstrated not only the software and hardware that teachers and students can use to explore multimedia as a learning tool, but also modeled the learning environment and philosophy that suits the use of this tool. This may well become the classroom model of the future.

See LOGO NEWS (Page 7)

FEATURE ARTICLE



Logo's Return

by ROBERT TINKER

There are major changes in the wind that will bring some important opportunities to the Logo community. Computer costs are finally dropping, with the eMate and WebTV starting a trend that will end with $200 computers in this academic year. Ubiquitous computing will finally be within reach of every student, and this will make it possible for us to finally realize the dreams of vastly better education fueled by technology.

Big visions for education have been articulated by technology advocates for two decades. Pessimists have termed these "faded dreams" that have yet to be realized in any substantial way, and they use this non-result as a way of discrediting the entire idea of using new technologies in education. I believe, on the contrary, that we are on the threshold of realizing these dreams. Computer technology and networking, and knowledge of how to use them in education, are only now sufficiently evolved to have a huge impact on improving education.

> Now that ubiquitous computing is near, Logo can reemerge to take the leading position in education it deserves.

The key barrier to date has been that the technology is too expensive to give learners sufficient exposure for them to master and appropriate these technological tools for their own use. With the recent emergence of low-cost computers specifically designed for students, schools can finally redesign their entire educational offerings on the assumption that every student has continuous, ready access to powerful, networked computers. The implications of this change are momentous, urgently need to be explored, and can be studied only in a real, functioning school district devoted to large-scale experimentation and supported by the best experts available.

The problem posed by information technology is that its best use in education, as it is in business, involves learning and applying tool-like applications such as word processors, spreadsheets, programming languages, and other specialized tools. For maximum benefit, the user must invest substantial time to learn the full set of functions each tool offers and then change one's own thinking patterns to exploit the new options the tool creates. For instance, the power of a word processor cannot be realized just by typing in an essay, running a spelling checker and printing out the result. Only when the user is familiar with outlining and able to move sentences and paragraphs around fluidly, can the word processor really help to conceptualize an essay and help the writer capture its logic succinctly in powerful, expressive words. In addition to very general "productivity" tools, the last two decades have seen the development of dozens of tools designed especially for education, such as Logo, the Geometer's Sketchpad, and probeware, that have proven effective. Each of these require an investment of time to master before returning rich educational dividends.

Becoming fluent with technological tools takes time. Even if there were more computers available in school, there is not enough time in the school day to make the required investment. For most teachers today, we cannot use time at home for this investment because we cannot assume everyone has access to a computer at home, and to pretend otherwise would put students from poor families at an unacceptable disadvantage. The result is that most students exposed to information technologies gain only a very superficial understanding of their capacity, and teachers cannot rely on student facility with technology to change instruction in any major ways. Unless schools can provide computers for home use, home time is unavailable for using computers and, as a result, regular tool use in instruction is almost impossible.

This year, the situation changed dramatically with the introduction of the first rugged, inexpensive, portable, networked computers designed specifically for education. Two commercial entries, the Apple eMate and the NetSchools StudyPro, represent what could well spark the revolution in education technology that visionaries have been predicting. Until this year, computer manufacturers have used the steady improvements of the underlying technologies to produce ever-faster, larger, more sophisticated computers at approximately the same price. The new class of educational computers is unique because for the first time adequate performance is made available at a lower cost in a package designed from the ground up for education. Both computers substitute flash RAM for hard drives and depend on networking for extended memory as well as communication. As a consequence, there is a more rugged computer with no moving parts, longer battery life, instant-on convenience, and lower cost. The result is an affordable portable computer that schools can give to every student.

The educational implications of this new class of computers are enormous, because (1) we can now have regular computer-based home assignments for all students, (2) we can use the technology to engage parents and care-givers in their students' learning, (3) we can assume student fluency in tools, and (4) we can restructure the curriculum to exploit this capacity.

Technology alone, no matter how promising, cannot cause educational change. We urgently need models for the intelligent implementation of this technology to support curricula, meet standards, and then improve the curriculum and set more ambitious standards for the future. What little research that has been done on "ubiquitous computing" supports the idea that huge improvements are possible, but it cannot provide guidance for the environment we can create now, with networking, portability, software, and care in implementation not represented in prior research. We must move quickly to seize the opportunity presented by the new technologies and show how it can enable our vision of a vastly improved education for all.

Information technologies can have four different levels of impacts on the curriculum:

Level 1. *Module replacement.* At the first level, information technologies are used to accomplish the existing curriculum goals by doing them better or to a higher level of student comprehension. For instance, labs based on probes connected to computers have the ability to reach very high levels of student comprehension of material previously mastered by few students.

Level 2. *Addition.* At this level, technology makes it possible to achieve new curriculum goals, usually by adding new material to an existing course. For instance, the TERC Global Lab project adds to existing science courses the possibility of international collaboration on environmental investigations by creating groups of schools linked through telecommunications.

Level 3. *Disciplinary restructuring.* At this level, the capacity of information technologies makes it possible to redesign a course or series of courses within a discipline. For instance, graphing can be introduced far earlier in the math sequence, giving students skills to help speed their understanding of many concepts that might have a graphical interpretation.

Level 4. *Interdisciplinary restructuring.* At this level, technology supports the redesign of courses across disciplines. For instance, if modeling using system dynamics was learned in ninth-grade math, then subsequent science courses could use this capability to address a broader range of science material at a deeper level.

These four levels involve increasing difficulty and educational payoff as illustrated in the Figure. Clearly, the first level is simplest to implement, because it is easy to substitute an improved, technology-based approach to a topic for a less effective one. This strategy has given rise to tens of thousands of small computer programs. The ease of implementation has often outweighed considerations of quality; just because the material is treated with technology does not guarantee that the learning is better. As illustrated, when there is a low level of technology implementation, there is sometimes a negative educational impact that can result in less learning.

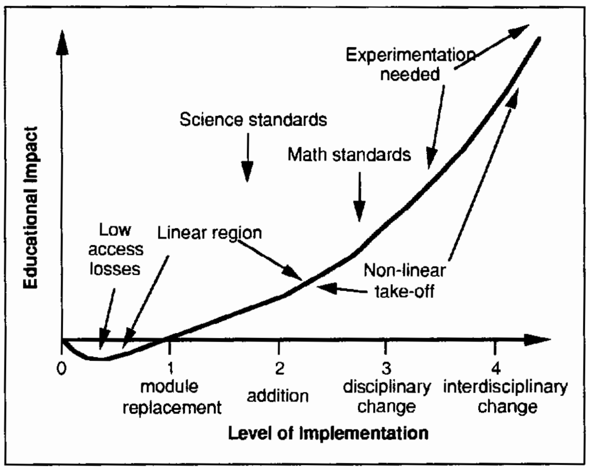


Figure: A representation of the potential educational gains for various levels of use of information technologies.

Level two is more difficult to implement because curricula represent a zero-sum game: for every topic added, something must be dropped. Making curriculum changes and justifying them to all the concerned educators and parents takes time and effort. This it the level envisioned in the science and more-ambitious math standards. In the Figure, this level is shown as leading to linear gains, because the educational gain is proportional to the amount this strategy is employed. Most current efforts to use technology in education are considered successful if reach this level and actually show some payoff from the technology. We take this level as a starting point and plan to use project resources to go further.

Levels three and four represent the large-scale curriculum changes that are possible only with ubiquitous technologies. Changes of this nature promise the greatest rewards, providing students with far deeper understanding of much more content than is currently expected. The gains are represented in the Figure as non-linear because implementations at one point in the curriculum have a ripple effect, permitting changes in all future studies. Such large-scale changes are difficult to implement in a single class or grade, because their graduates might not have the familiar set of knowledge and because transfer students will face difficulties. As a result, such really large-scale changes have not been practical in the past. Now for the first time, it is possible to dream of the kind of major education improvements that result from level-four implementations.

A mix of tools will be needed, but first and foremost among them should be Logo. Because it is a programming language, Logo is likely to be the tool with the largest potential payoff across the curriculum. Furthermore, there are three decades of research and extensive classroom experience on its use. It is frustrating that there is a perception in the educational community that Logo has been tried and rejected. In my opinion, it has had a bum rap because it was simply too far ahead of its time.

Now that ubiquitous computing is near, Logo can reemerge to take the leading position in education it deserves. We need to prepare for this reemergence by codifying what has been learned, developing increasingly Web-friendly Logo, and developing the capacity for a massive teacher professional-development effort. We also desperately need testbed schools willing to provide leadership in implementing level-four curriculum changes based on a mix of technology tools where Logo plays a prominent roll.

About the Author

Dr. Robert Tinker is the Director of the Concord Consortium. For many years he was associated with TERC where he helped develop the Global Lab, National Geographic KidsNetwork, and the Personal Science Laboratory.

Robert Tinker
bob@concord. org
http://www.concord.org

LOGO NEWS / Continued from Page 4

The workshop modeled a classroom in which the teachers acted not as conveyors of information but as facilitators of student-driven exploratory learning. The students of the workshop decided on projects for exploration using multimedia as the platform and formed groups based on a common interest. The presenters helped by challenging groups to not limit themselves but to be ambitious. Ideas were generated when participants made a suggestion of an area of interest to pursue, ranging among such subjects as "design a simulation of a garden growing," "make a multimedia project about making a multimedia project," or "a presentation about gold"—all very broad and open topics.

As students, we chose a project, and individual focus and enthusiasm were generated immediately, as everyone was working on something they found interesting. Each group decided on a direction, divided the job into a number of tasks, and allocated group members to do each of them. We then set about exploring. We had at our disposal video and video-editing software, digital cameras, microphones, modems and the Internet, keyboards and midi software (even a digital guitar), and MicroWorlds 2.0 to tie it all together.

The institute program allowed for large chunks of time to devote to the task at hand, which was a necessity as most of the learning was through exploration. Groups worked within the same space, and we were encouraged to help each other solve problems. On three occasions during the two-day workshop, we were all gathered together to "demo or die," where all groups demonstrated their progress to-date and told of their trials and tribulations.

Gary and Fred were always on hand, moving among the groups, offering advice and challenges, providing tutorials on some of the hardware and software, and providing encouragement. To break up the time, Gary and Fred gave short presentations on their work and ideas, and led debates on current educational philosophy.

At the end of the two days, we had learned a great deal about using multimedia and had also experienced a learning environment that is ideal for facilitating the type of learning that will allow students to make the most of multimedia as a tool in their own learning.

StarLogo Models Available for Download

The Tufts University project, Making Sense of Complex Phenomena through Building Object-Based Parallel Models, has many StarLogo models available for download on their Web page. They encompass a variety of different traditional content domains including mathematics, physics, chemistry, biology, economics, engineering, art, and the social realm. Many of these models have been designed to be used in schools as so-called "extensible models"—that is, first used for demos, then for exploration, and finally for modification of the underlying assumptions (and code) of the models. These models include lots of accompanying explorations and activities.

This project is closely linked to the Connected Probability Project. As a result, many of the models explore what global effects can be achieved using probabilistic distributions and algorithms.

The web page is at:

http://www.tufts.edu/as/ed/cm/index.html

Send Logo News to
logoexchange@moon.pepperdine.edu

BOOK REVIEW / CAROLYN DOWLING

The Cyber Self

Life on the Screen: Identity in the Age of the Internet
Sherry Turkle
1995, Simon & Schuster (New York)
ISBN: 0-68480-353-4

It is now over a decade since *The Second Self: Computers and the Human Spirit* burst into our lives, precipitating what was for most of us a very different way of looking at computers and at ourselves, both in combination and by contrast. Of special interest were a number of issues, brought into focus by the presence of computers, relating to what it really means to be "alive" and to be "human."

During the intervening period, there has been a great deal of change in the attributes and capabilities of computing technology and, consequently, in the use we make of it. The computer of the 1980s was "personal" not simply in our striving for individual ownership of the hardware, but in the relationship between user and machine. For the most part, interactions between computer and user were on a one-to-one basis. In many instances the software interface, which for the majority of users effectively "is" the computer, was deliberately personified, adopting a range of tones and stances according to the purposes of the software and the imagined characteristics of the user. Beyond this level of anthropomorphism lay a number of inadvertent expectations and reactions to computers shaped by the heavy reliance of most interfaces on language and by much of the imagery implicit in the terminology of computing, such as the concept of artificial "intelligence." Small wonder that these artifacts, apparently possessed of at least two of the defining characteristics of human beings, could be seen as occupying an ambiguous and complex position relative to their users.

Underpinning the argument of the book is a rich and subtle understanding of the extent to which a culture of simulation is increasingly permeating our lives

The computer of the 1990s is a very different kettle of fish. The relationships currently enjoyed by users are less commonly with the computer as an entity in itself than with a range of other users, or with information, accessed through the computer but physically remote from the desktop. The once personal, personified machine, its functionality residing unequivocally within the boundaries of the plastic container atop the individual desk or lap, has been transformed into a conceptually transparent conduit or gateway to rich and complex environments of interaction. It is these new worlds that provide the context for Turkle's more recent work, in which the focus shifts from the general issue of what it means to be "human" to the vexed but eternally intriguing question of what constitutes "reality" and, most particular, the "real" self.

Underpinning the argument of the book is a rich and subtle understanding of the extent to which a culture of simulation is increasingly permeating our lives, its practical manifestations both encouraged by and giving support to various strands of contemporary philosophy and theory, in particularly aspects of postmodernism. It is an extremely "personal" book in the most satisfying sense, in that its insights derive from a unique blend of direct personal experience and depth of scholarship in what many would justifiably regard as fairly diverse areas.

The book falls into three fairly discrete but thematically linked sections within which individual chapters address in considerable depth the new directions and styles that characterize recent developments in computing. While the first two sections provide a wealth of fascinating and extremely accessible discussion of a range of concepts and practices sometimes regarded as a trifle esoteric, it is perhaps predictable that the casual reader (including most of my tertiary level students) makes a beeline for section three, On the Internet, arguably less "technical" in focus and of most obvious relevance to their personal lives. This section addresses the issue of the special opportunities provided by the Internet for a person to construct and enact a multiplicity of identities. While the traditional beliefs

See BOOK REVIEW (Page 30)

TEACHER FEATURE

John St. Clair

by JIM MULLER

In 1984, Apple Computer Inc. made a bold move to advance the use of computers in California schools. They gave an Apple IIe and Apple Logo to each public school in the state. This is how John St. Clair was introduced to Logo. As he describes it, "it motivated me to learn more about computers."

Logo fan(atic)s around the world are certainly glad John became motivated. He has become an active Logo spokesperson serving on the Board of Directors of the Global SchoolNet Foundation and moderator of Schl-sig-logo and Logo-L listserv. In his "spare time," he is technology teacher and mentor at Vina Danks Middle School, Ontario, California.

How did you get started using Logo?

I started using Apple Logo with my kindergarten students in 1984 because that was the only software that was delivered with the free computer from Apple. I was amazed at how creative my kindergarten students were at drawing geometric designs using a single-key Logo procedure that I wrote for them.

What did others in your school think?

Actually, I was the only teacher in my school district who was enthusiastic about using Logo. I enjoyed reading *Logo Exchange* articles to get ideas and support for what I was doing. To get more people interested, I used more user-friendly versions of Logo, as they became available such as Apple Logo II and LogoWriter.

What did or do you see in Logo? What got you so interested?

There are two things I like about Logo. Students with low academic abilities are able to be successful using Logo and students with high academic abilities are challenged to be independent learners. I now work with all levels of students including limited English- speaking students and those learning disabled students who can't read. It is very gratifying to see "wanna-be" gang members get excited about making the turtle draw a car or a truck, to see them truly earn an "A" grade in my class. I always hope that some of those students will remember what it was like to succeed in my class and give that same effort towards making a better life for themselves.

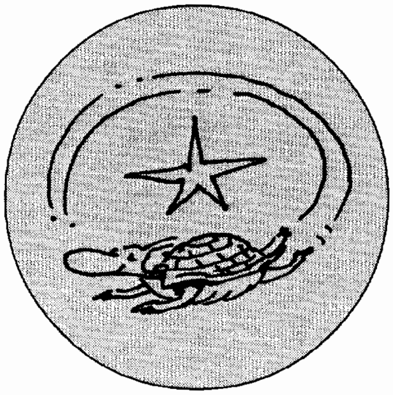
What have you done to get others involved?

Out of necessity, I have become a great believer in on-line communication. After I became a technology mentor-teacher in my district, I became the sysop (systems operator) of the local FrEdMail BBS. I pushed for a Logo newsgroup topic on FrEdMail so that I could get in touch with other teachers who used Logo. Through this group, I did several LogoWriter project exchanges with teachers from Nebraska and Pennsylvania. When FrEdMail incorporated as a non-profit foundation in 1991, I became a founding member of the Board of Directors. It is now known as the Global SchoolNet Foundation (GSN). When Michael Tempel, President of the Logo Foundation, and Al Rogers, Executive Director of Global SchoolNet Foundation, asked me to be a co-moderator with Michael of the Logo-L listserv, I was delighted. There are now close to 300 Logo subscribers worldwide, including those from the Logo newsgroup. I handle technical difficulties, read and cross-post messages from comp.lang.logo (the Logo Usenet newsgroup). *To join Logo-l send subscribe logo-l in the body of an e-mail message to majordomo@gsn.org*

What happened to your Logo work as you developed this electronic network?

In 1992, I moved from teaching at a K-1 school to teaching at a middle school with sixth- to eighth-grade students. I currently spend all day in a computer

See ST. CLAIR (Page 30)



Self-Organizing Behavior

StarLogo is used primarily for modeling systems that have no leaders. Some examples of these leaderless, or self-organizing, systems in our everyday world are bird flocks, traffic jams, and termite and ant colonies. StarLogo provides a useful means for students and researchers (and we are all researchers of one sort or the other!) to study and manipulate models of these systems to understand their dynamics.

To help familiarize yourselves with the modeling power of StarLogo, we'll create a program step by step to model a colony of termites. The intent is to introduce some turtle and patch commands, as well as an understanding of programming, in StarLogo. You will also learn to make simple changes to StarLogo programs.

Tutorial

Before we go on, be sure to obtain StarLogo and install it on your computer. [At the time of this writing, only Macintosh computers are supported.] See the end of the column for ways to obtain StarLogo.

As we go along this tutorial, we will ask you to type some things on the computer. These will be in **Bold** letters, so it will be easy to figure out when and what to type.

Start up the StarLogo program, and select the "New" menu command from the "File" menu (File/New) to start a new model.

We'll be doing some of our work in the Command Center and some in the Procedures window. You can find each of these in the "Windows" menu if they are not visible. We'll start by using the Command Center.

The first thing we need to do is create some "turtles," which we will call termites. (We'll call them turtles, too, because that's what StarLogo calls them).

An easy way to create turtles is to use the create command: crt 100, which will create 100 turtles on our screen. Type **crt 100** into the Command Center window, followed by a Return (or Enter).

You might be asking, "Where are all the turtles?" because there is now only one spot in the center of the Graphics window. Well, they're actually stacked up on top of each other. Also, they won't really look like turtles, they will look like colored spots. [In future columns, I'll describe a way to use turtles with unlimited shapes.]

Try typing **fd 10** (with Return) into the Command Center. And there they are! Because they all have a slightly different compass heading, their moving forward creates a circle.

Also notice that you typed **fd 10** only once, and all 100 turtles moved at the same time! Each command you type is used by all the turtles.

Let's put these first few commands into a setup procedure, which is a collection of instructions, such as the ones you have typed in the Command Center, with a name attached. To use a procedure, all you have to do is type its name.

All procedures are entered in the Procedures window. Find that window and type:

```
to setup
crt 100
fd 10
end
```

Remember that the word "to" begins every procedure. Unlike commands typed in the Command Center, procedures typed into the Procedures window are reusable until changed or deleted.

Let's try out our first procedure. First type **ca** into the Command Center to clear all. Now type **setup**, and watch our turtles appear. Typing setup caused the setup procedure to run.

But what if you don't want to have to type ca every time before starting over? Incorporate that into the setup procedure, too:

```
to setup
ca
crt 100
fd 10
end
```

Now our screen will be automatically cleared each time we type setup. Try it.

To make this even easier, let's create a setup button. Then, when we press the button, it will do the procedure setup without our having to type anything in the Command Center.

Click on the Interface window (or use the Windows/Interface menu item), and choose the blue button icon in the palette. Now click in the empty Interface window to make a button.

When a dialog window pops up, type **setup** in the box where it says logo instruction. Then click OK.

Now you have a setup button that when pressed will perform your StarLogo procedure called setup, which will create 100 turtles and move them forward. Try pushing the button.

Now that we have 100 turtles (or termites), let's give them something to do. Why don't we create objects called woodchips. Then we can give the termites certain rules about what to do with the woodchips.

Let's make all of our termites red and our woodchips yellow so we can tell them apart.

To do that we say: setc red to set the color of the termites to be red. Type **setc red** in the Command Center.

Now add that to your setup procedure:

```
to setup
ca
crt 100
setc red
fd 10
end
```

We're now ready to create the woodchips. Let's use yellow-colored patches for the woodchips to contrast them visually from the termites. This will be only a little trickier than creating turtles.

What we want is one out of every five (20%) patches to have a woodchip. We'll use a random number to do this. A random number is one that the computer picks, kind of like out of a hat. We're going to ask the computer to pick a number between one and 100. If the number is below 20, then we'll drop a yellow woodchip; otherwise, the patch will stay black indicating that it is empty.

So, let's put these ideas together to create the woodchips. Type this in the Command Center:

```
if (random 100) < 20 [setpc
yellow]
```

This says that for each patch, if the number you get from "random 100" is less than 20, do what is inside the square brackets. The other 80% of the time, do nothing and leave the patch black. Unlike setc, which is used for turtles, setpc sets the color of the patch. StarLogo understands that because setpc is a patch command, that this entry applies to all the patches instead of the turtles. As with the turtles, each patch executes this command line in turn.

Add this command to the setup procedure, which now looks like this:

```
to setup
ca
crt 100
setc red
fd 10
if (random 100) < 20 [setpc
yellow]
end
```

Now when we press the setup button, we see red termites and yellow woodchips.

We've completed the setup procedure for our program and learned something about making a lot to termites, coloring things, and using procedures.

So now let's make our termites interact with the woodchips. To do this, we need to make them move around and do something.

Let's have the termites search for woodchips and place them into piles. There are three rules that each termite needs to follow to successfully do this:

- If you see a woodchip, pick it up
- Search for another woodchip (presumably in a pile of at least one)
- Find an empty space near that pile and put this woodchip down

We'll start with a go procedure that will call these three rules. Type the following into the Procedures window, after the setup procedure (for clarity, leave a blank line between the procedures):

```
to go
;;search-for-chip
;;find-new-pile
;;find-empty-patch
end
```

The two semicolons mean: Ignore everything after the semicolons on this line. We use them because we haven't yet written these three procedures, and StarLogo will complain until they exist. Later, when the procedures are written, we'll remove the semicolons. [Semicolons are also very useful for embedding comments and notes in a program.]

Let's write the procedure called search-for-chip. In search-for-chip, we want the termites to first find a woodchip. How about this: If the termite moves over a yellow woodchip, then it removes it from the patch.

To do this we type this in the Procedures window after the go procedure:

```
to search-for-chip
if pc = yellow [stamp black
jump 20 stop]
end
```

What this means is that every termite checks whether the patch it is on is yellow, and if it is, turn it to black, jump far away, and stop executing this procedure.

So far, we have search-for-chip changing yellow woodchips to black to indicate that they have been picked up and are no longer on the ground.

But what if the termite isn't on a patch with a woodchip? We don't want those termites just standing around doing nothing! (What would Henry Ford say?) Let's get the termites to move while they search.

First, we'll write a simple moving-around procedure. We'll call it wiggle:

```
to wiggle
fd 1
rt random 50
lt random 50
end
```

This procedure just moves the termite forward one unit and then wiggles by turning randomly a little to the left and to the right.

Let's also change our search-for-chip procedure to do the wiggle before checking for a woodchip, and to keep going as long as it hasn't found one. Find the search-for-chip procedure in the Procedures window and change it to look like this:

```
to search-for-chip
wig
if pc = yellow [stamp black
jump 20 stop]
search-for-chip
end
```

Now click on the Interface window. WHOOPS! I'm sorry. I made you make an error, and a pop-up box shows you what it is: StarLogo doesn't know anything about a procedure called wig. Sometimes errors will occur during programming, and a pop-up box will help you figure out what went wrong.

Click OK in the pop-up box, and change wig to **wiggle** in the search-for-chip procedure. That should fix the bug in our program.

Before, when we made the go procedure, we put semicolons in front of the procedures until we wrote them. Well, now we've written the search-for-chip procedure, and we can remove the semicolons from in front of it. The go procedure should now look like this:

```
to go
search-for-chip
;;find-new-pile
;;find-empty-patch
end
```

Test our go procedure by typing **go** in the Command Center. This will result in the termites moving around and turning many of the yellow woodchip patches to black. Of course, the red termites will be covering the black patches, so you won't be able to see them yet.

So far, so good. Save our work up to this point. Use the File/Save As menu, and type your name as the file to save. Click OK. It's a good idea to save your work regularly, even as you are working.

Now get the woodchip-carrying termites to keep searching for other woodchips. This procedure will look a little like search-for-chip:

```
to find-new-pile
wiggle
if pc = yellow [stop]
find-new-pile
end
```

All we had to do for find-new-pile was tell the termite to wiggle and, if it sees a yellow woodchip, to stop executing this procedure. Otherwise, keep looking (do find-new-pile again).

Test our latest procedure by typing find-new-pile in the Command Center. The termites will move around looking for another yellow woodchip and then stop.

Good, we've written the second of our three rules, so we can remove the semicolons from the find-new-pile call of the go procedure. It should now look like this:

```
to go
search-for-chip
find-new-pile
;;find-empty-patch
end
```

Now, we're ready for the last step: finding an empty patch to place the woodchip.

We know a patch is empty if it's color is black; that is, if pc = black.

To drop the woodchip, we make the black patch turn yellow by stamping it yellow. Here's the last rule:

```
to find-empty-patch
wiggle
if pc = black
[stamp yellow
stop]
find-empty-patch
end
```

Here we tell the termite to wiggle, and if it finds an empty (black) patch, stamp the patch yellow (the same as dropping a yellow woodchip) and stop. If not, keep searching for an empty patch.

After they've found an empty patch and dropped the woodchip, we want them to get-away and look for more woodchips. If we don't do this, the termite will likely pick up the same woodchip and start all over again in the next round.

Let's write a simple get-away procedure to accomplish this:

```
to get-away
seth random 360
jump 20
if pc = black
[stop]
get-away
end
```

Seth random 360 sets the heading of the termite to be some random compass direction from zero to 360 degrees. jump 20 is similar to fd 20, but much faster, more like teleporting. If a termite lands on a black patch, stop executing the procedure. Otherwise, keep jumping to get away.

Add get-away after it puts the woodchip down in our find-empty-patch procedure:

```
to find-empty-patch
wiggle
if pc = black
[stamp yellow
get-away
stop]
find-empty-patch
end
```

Test our last rule by typing **find-empty-patch** in the Command Center. The termites will move around looking for a patch to drop their yellow woodchips and jump away.

We also must change the go procedure because all the rules are now written:

```
to go
search-for-chip
find-new-pile
find-empty-patch
end
```

Now all our procedures are complete and tested!

Unfortunately, our go procedure tells each one of our 100 termites to move a woodchip to another pile, but only one time. What we need is a button to run the go procedure over and over again.

See STARTING WITH STARLOGO (Page 32)