

Logo Exchange

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



INSIDE

- The Art of LEGO Design
- ToonTalk and Playground Presentation
- Geometric Skills
- Using Multiple Pages and Animation
- The Y2K Explorer
- AgentSheets
- What's New in MicroWorlds Pro
- Who Wants to Be a Millionaire MicroWorlds Pro Style
- Logo and High-Level Geometric Thinking
- Happy Birthday LX! StarLogoT2000 Released
Logo News

Logo Exchange

Volume 18 / Number 1

Editorial Publisher

International Society for Technology in Education

Editor-in-Chief

Gary S. Stager, Pepperdine University
logoexchange@moon.pepperdine.edu

Copy Editing, Design, & Production

Ron Richmond

Founding Editor

Tom Lough, Murray State University

Design, Illustrations & Art Direction

Peter Reynolds, Fablevision Animation Studios
pete@fablevision.com

Contributing Editors

Dr. Douglas Clements, SUNY Buffalo
Dr. Carolyn Dowling, Australian Catholic University
Alan Epstein, Metasoft
Dr. Brian Harvey, U.C. Berkeley
Daniel E. Kinnaman, Curriculum Administrator Magazine
Dr. Julie Sarama, Wayne State University

International Editor

Jeff Richardson, Monash University, Australia

International Editor Emeritus

Dennis Harper, Olympia, Washington School District

SIGLogo Officers

Jeff Richardson, President
Steve Costa, Vice-President
Hope Chafian, Secretary/Treasurer
Gary S. Stager, Editor

SIG Coordinator

Tom Magness

1999-2000 ISTE BOARD OF DIRECTORS

Heidi Rogers, President University of Idaho
John Vaille, Chief Executive Officer ISTE

Executive Board Members

Lynne Schrum, Past President University of Georgia-Athens
Cathy Gunn, Secretary Illinois Virtual Campus
Michael Turzanski, Treasurer CISCO Systems (MA)
Chip Kimball, At-Large Lake Washington School District (WA)
Jan Van Dam, At-Large Oakland Schools (MI)

Board Members

Larry Anderson Mississippi State University
Marianne Handler National-Louis (IL) University
Kathy Hurley The Learning Company (MD)
Pam Korporaal Norwalk-La Mirada USD (CA)
Cheryl Lemke Milken Family Foundation
David Moursund (ex officio) ISTE
Jorge Ortega Leon County School District (FL)
Marilyn Piper Washington Middle School (WA)
Sue Waalkes Upper Dublin School District (PA)
Peter Wholihan Department of Education (Virgin Islands)
Cheryl Williams National School Boards Association (VA)

ISTE Committees

Lajeane Thomas Accreditation and Standards
Dave Brittain Awards
Cathy Gunn Distance Learning
Michael Turzanski Finance
Paul Resta and Gerald Knezek International
Jenelle Leonard Minority Affairs
Lary Smith Policies and Procedures
M. D. Roblyer Publications

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@stager.org

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.

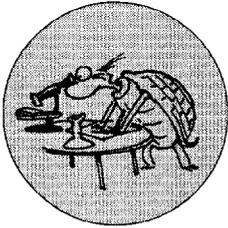
Logo Exchange solicits articles on all topics of interest to Logo-using educators. Submission guidelines can be obtained by contacting the editor. Opinions expressed in this publication are those of the authors and do not necessarily represent or reflect official ISTE policy.

© 2000 ISTE. All articles are copyright of ISTE unless otherwise specified. Reprint permission for nonprofit educational use can be obtained for a nominal charge through the Copyright Clearance Center, 27 Congress St., Salem, MA 01970; 508.750.8400; Fax 508.750.4470. ISTE members may apply directly to the ISTE office for free reprint permission.

POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 480 Charnelton St., Eugene, OR 97401-2626 USA. Periodicals postage paid at Eugene, OR. USPS# 000-554. 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®.

Contents



ARTICLES

- The Art of LEGO Design
Fred Martin 5
- The ToonTalk and Playground Presentation at Logosium
Ken Kahn 13
- The Y2K Explorer
Patrick Mahaey 17
- AgentSheets: End-User Programmable Simulations
as Interactive Media
Alexander Repenning 19
- Use Multiple Pages and Animation Together,
or building games in MicroWorld without tears
Jeff Richardson 25
- What's New in MicroWorlds Pro
Gary Stager 27
- MicroWorlds Tutorial: Functions and Graphing
Gary Stager 29
- Who Wants to Be a Millionaire MicroWorlds Pro Style?
Anonymous and Gary Stager 30



COLUMNS

EDITORIAL

- Happy Birthday Logo Exchange!
Gary S. Stager 2

QUARTERLY QUANTUM

- A Terminal Case of Logo
Tom Lough 3

LOGO NEWS

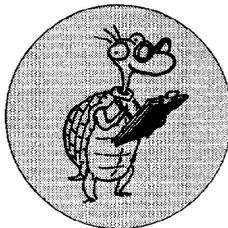
- Logo Exchange Staff 4

STARLOGO STARTERS

- StarLogoT2000 Released
Uri Wilensky 15

LOGO: SEARCH AND RESEARCH

- Logo and High-level
Geometric Thinking
Douglas H. Clements
and Julie Sarama 23





Happy Birthday Logo Exchange!

Happy birthday to you. *Logo Exchange* is celebrating its 18th year of publication. It's hard to believe, but Tom Lough started this proud little publication 18 years ago. *Logo Exchange* is one of the longest-running publications dedicated to any aspect of educational computing. While the *T.H.E. Journal* may be older, one is hardpressed to think of more senior publications. I hope you enjoy reading *Logo Exchange* as much as you have over the past three decades.

Anyone who can remember being 18 years old or is the parent of an 18-year-old can attest to the trials and tribulations associated with that age. Publishing *Logo Exchange* is no different. It seems that a few of our wonderful contributing editors are enjoying their first fall away from home. (Writing columns four times a year without compensation often takes a backseat to other obligations) Teacher Feature and Book Reviews will be back in the next issue. You will receive three more issues of *Logo Exchange* by the North American Summer of 2000.

It is impossible to publish *Logo Exchange* without the generosity, ideas and contributions of Logo users from around the world. Please consider sending tales of your classroom Logo adventures, research articles and Logo project ideas to *Logo Exchange* logoexchange@stager.org. We will help



Dr. Papert surprises Logosium participants

you get the articles in shape for publication in a future issue.

Logosium '99 in Philadelphia was a great success. For the second year in a row, Logosium was the best attended NECC preconference workshop.

Participants enjoyed hands-on workshops, provocative round-table discussions, Philly chessesteaks and a surprise after-lunch keynote discussion led by Dr. Seymour Papert. While we hope to publish a transcript of this rare talk in a future issue, one should remember that all sorts of great surprises are in store at Logosium each year (see photos throughout the issue).

Logosium 2000 will mix great workshops, food, discussions and collabo-

rations in Atlanta on June 25, 1999 (see www.neccsite.org for more information). NECC will also feature a SIGLogo sponsored panel discussion, Logo's Second Millennium.

The Logo community is in great shape. New versions of Logo, such as MicroWorlds Pro and StarLogoT, continue to be developed and exciting work continues in classrooms around the world. This issue features a number of articles detailing the new Logo versions, reports from Logosium '99 and contains a number of tips for beginning Logo users.

SIGLogo is also blessed to have a great new slate of officers. The two Aussies leading our organization demonstrate the international commitment of SIGLogo. Veteran Logo users Jeff Richardson of Monash University and primary teacher Steve Costa of Methodist Ladies' College are President and Vice President, respectively. Outstanding New York Logo-using teacher Hope Chafian is once again our Secretary/Treasurer. Discussions are under way about new SIGLogo publications and our future Web presence. Stay tuned.

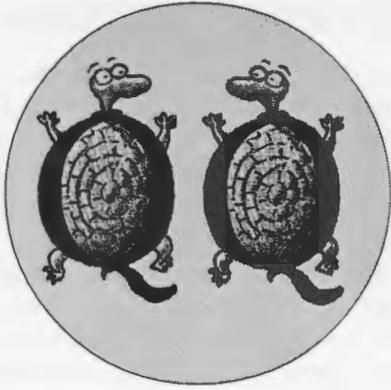
Happy New Millennium!



Gary S. Stager

Gary

Gary Stager, Editor-in-Chief
logoexchange@stager.org



A Terminal Case of Logo

Ah, the mystery of words! It all happened at our welcome-back-to-school luncheon meeting of our university faculty.

During the program, one of the speakers spent several minutes reporting on the accomplishments of the university and then began to highlight the achievements of the faculty. In summarizing, the speaker said that the quality of the university was evident from these achievements and from the high percentage of the faculty with terminal degrees.

Terminal degrees? Well, I knew what the speaker meant to say. But, at that particular moment, I was in one of those moods that—well, you know! I launched out on another mental excursion.

Terminal degree? What? Do you mean something like a train or bus terminal? Like the end of the line? In a way, this is exactly what the speaker meant.

The doctorate is regarded as the uppermost degree in a progression and is often called a terminal degree. It is the final milestone of a long and arduous trail. When you get it, you have arrived! Or so they say.

But there are other terminal degrees besides a doctorate. I think that practically any culminating professional achievement can be viewed in this way—even getting a teaching certificate.

Have you ever encountered teachers who seem to view their terminal degree in this light? They perceive their teach-



Logosium lets participants interact over a great dinner

ing position as their final accomplishment, and life then becomes a holding pattern. “I’ve always done it this way,” becomes some sort of mantra, and turning the pages of the school calendar passes for exercise. We have all seen them, in universities as well as in K–12 schools.

For teachers with that viewpoint, the terminal degree is almost like a terminal illness. Their focus is maintaining their position, and they are not interested in anything else. Their mental version of Logo has no FORWARD command. Too bad.

On the other hand, I have encountered many teachers who tell me, “The more I

learn, the more aware I am of what I do not know.” I have heard this from persons with degrees of every stripe, color, shape, size, and level. What does this mean for a so-called terminal degree?

Well, I will admit that, for me, my doctorate was the end of a line. So I must accept that part of the definition. However, just as trains can depart from as well as arrive at a terminal, we can also view a terminal degree as the beginning of another line—that of a lifelong learner!

It reminded me of my son’s perspective on driving as he approached his

See **TERMINAL CASE** (Page 4)



Logo News

Registration is now open for the 2000 Logo Summer Institutes. Here are the dates, locations, and brief descriptions:

Logo Immersion

New York: June 19-23, 2000

Colorado: July 17-21, 2000

Immerse yourself in Logo training and project building. Use MicroWorlds from LCSi, Terrapin Logo, and other Logo software environments. Choose from among several strands including multimedia, Logo and mathematics, simulations, Logo for young children, and game design.

Logo Robotics

New York: June 19-23, 2000

Colorado: July 24-28, 2000

Design and build cybernetic devices of all kinds and control them using Logo programs. Use a variety of robotics sys-

tems from LEGO Dacta, Terrapin Software, and the MIT Media Lab.

Full information, including registration forms, is on the Logo Foundation Web site at www.logofoundation.org.

I hope to see you in New York or Colorado next summer.

Using Technology to Discover Your World: Projects, Problem Solving, and Presentations with MicroWorlds

A summer institute sponsored by: The St. Paul Public Schools and the Science Museum of Minnesota, June 19-23, 2000, St. Paul, Minnesota.

Come explore learning with MicroWorlds together with other educators expanding their constructionist approach to teaching and learning.

For information contact:

Paul.Krocheski@spps.org

(gasp!) 16th birthday. I believe he views his driver's license as the "terminal degree" of life as a teenager perceives it. I can understand that. I'm old, but not THAT old! In actuality, however, he has to get a learner's permit first.

That's it! The terminal degree (or any degree, for that matter) is sort of like a learner's permit. It marks only the beginning of a period of learning that stretches out for the rest of our lives! It gives us permission to grow, to try, to change, to encounter, to learn new things.

Hmm. Do we view our Logo experience as a terminal degree? Once we have "learned" Logo or "done" Logo, have we arrived?

Of all learning environments, in my opinion, Logo seems to be one of the best representations of the perspective of a learner's permit. After encountering Logo commands and Logo concepts, who can resist the urge to play around with them and learn even more? Who can resist the imperative to share this excitement with others?

What a wonderful opportunity for our students, and for us! But what about . . .

I was startled back into the present by the sound of applause. Hey! Someone really likes my idea! Oops! They are applauding the speaker at our faculty luncheon. Is the program over already?

Well, I have lots of things to think about now. I want to view myself as a lifelong learner, and to think of my terminal degree as a learner's permit for this process. I want to figure out what to do next, and how to do it.

Let's see . . . I have always wondered why that subprocedure did that funny little thing right after the turtle. . .

FD 100!

LX



A lively panel discussion at Logosium '99

**Tom Lough, Founding Editor,
Murray State University
Department of Elementary and
Secondary Education,
PO Box 9, Murray, KY 42071.
phone: 502.762.2538
fax: 502.762.2540
tom.lough@coe.murraystate.edu**



FEATURE ARTICLE

The Art of LEGO Design

by **FRED G. MARTIN**

Editor's Note: This article was originally published in *The Robotics Practitioner: The Journal for Robot Builders*, volume 1, number 2, Spring 1995; trp@footfalls.com

There is a real need for better resources for both fledgling and in intermediate LEGO builders. The plans that the LEGO company distributes with its kits are very good at showing how to build specific models, but not so good at teaching how to design from one's own ideas. At the MIT Media Laboratory, we're working on a project we call the LEGO Constructopedia, a hypermedia resource for LEGO designers that will include LEGO building plans, design principles, textual descriptions, and rendered animations, all interlinked, indexed, and browsable. The project is just beginning and is still in the conceptual stages; this article is my attempt to present some of the content of our proposed LEGO Constructopedia in a more traditional form.

The article begins with an analysis of the structural principles of the LEGO system, continues with a discussion of gears, gear reduction, and geartrains, and finishes with a visual assortment of various building tricks or "clichés." Interspersed throughout are numerous diagrams and sample models to illustrate the ideas being presented. I hope that LEGO aficionados at all levels from novice to expert will find something of interest here.

Two beams are separated by two $1/3$ -height LEGO plates, creating a vertical interval of $1\ 2/3$ units, which is equal to 2 horizontal units. Hence the beams can be locked into place using cross-beams and connector pegs—the way to make your LEGO construction quite sturdy.

Structure

The Vertical Dimension Relation

Let's begin by examining the LEGO brick in detail. Most people realize that the LEGO brick is not based on a cubic form. The height of the brick is a larger measure than the length and width (assuming the normal viewpoint of studs on the top). But few people know the secret relationship between these dimensions: the vertical unit is precisely $6/5$ times the horizontal ones. Put another way, a stack of five LEGO bricks is exactly equal in height as a six-stud LEGO beam is long.

The origins of this obscure relationship remain shrouded in mystery, but it has real practical value: by building structures with vertical heights equal to integral horizontal lengths, it is possible to use beams to brace LEGO constructions. This technique is greatly facilitated by the $1/3$ -height plates, which allow a number of vertical spacing possibilities.

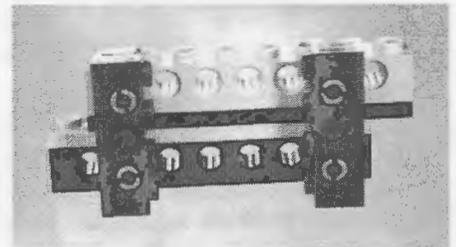


Figure 1. Two beams locked using $1\ 2/3$ vertical spacing relation

The most common trick is to create two horizontal units of space in the vertical dimension by separating two beams with two plates (Figure 1). This $1\ 2/3$ vertical measure is two units of horizontal measure since $1\ 2/3$ times the conversion factor of $6/5$ equals 2. Another useful pairing is $3\ 1/3$ vertical units (i.e., two beams separated by two beams/bricks and one plate), which equals 4 horizontal units (see Figure 2).

In addition to constructing perfect spacings vertically, it's possible to make diagonal braces. A 3-unit horizontal spacing with a 4-unit horizontal spac-

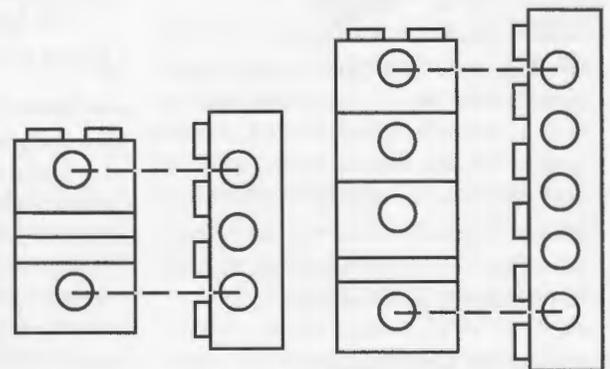


Figure 2: Creating Vertical Spacings with Two-Unit and Four-Unit Horizontal Measures

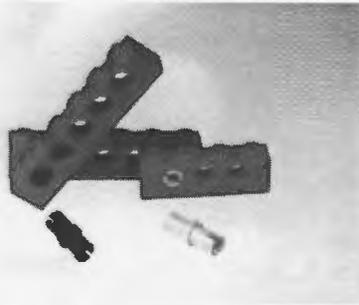


Figure 3: Black Connector Peg Versus Gray Connector Peg

ing vertically yields a 5-unit diagonal by the Pythagorean relation. This is an example of a perfect diagonal spacing, but near-perfect spacings that are “close enough” exist. Experiment, or spend some time thinking about the numbers.

Figure 1 shows the practical application of this dimensional relation: two beams locked together with cross-beams and connector pegs. You can use the vertical spacing trick for at least two purposes. First, use it to lock vertical structures on LEGO machines in place with beams and connector pegs (see more about connector pegs in Figure 3). Second, create vertical spacings that are the right intervals to allow gears to mesh properly (more on this later). This trick will go a long, long way in making sturdy, reliable LEGO designs.

Gearing

Turn on a small DC motor, like the stock LEGO motor, and what do you get? The shaft spins really fast, but with almost no torque (turning force). You can easily pinch the shaft with your fingertips and stop the motor from turning.

Through the magic of gear reduction, this fast-but-weak motor energy can be transformed into a strong but slow rotation, suitable for powering wheels, gripper hands, elbow joints, and any other mechanism. Along with structural issues, building effective geartrains is the other half of the challenge of creating working LEGO machines.

Counting Gear Teeth

Gear reduction is achieved by intermeshing gears of different sizes

The black connector peg vs. the gray connector peg: what is the difference? The answer is that the black peg is *slightly larger*, so it fits quite snugly in the beam hole, while the smaller gray peg rotates freely. Use the black pegs to binding structures together, as suggested by the discussion on locking cross-beams, and use the gray peg when making hinged joints.

with compatible teeth. Figure 4 shows the effect of meshing an 8-tooth gear with a 24-tooth gear. When the 8-tooth gear rotates three times, it has advanced the 24-tooth gear one revolution. Hence this configuration produces a 3-to-1 gear reduction ratio.

More gear reduction can be achieved by meshing gears with greater disparities of teeth count. Using the LEGO 8-tooth and the LEGO 40-tooth gears produces a 5-to-1 reduction. But the more general solution is to gang together—or multiply—single pairs of gear reduction. Figure 5 shows how two 3-to-1 reductions may be ganged

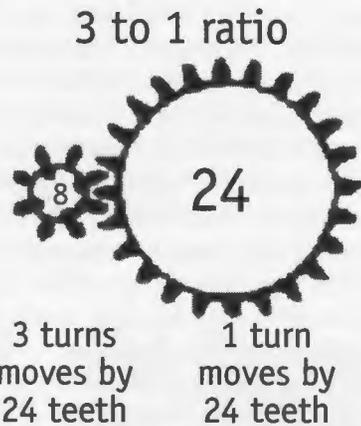


Figure 4: 3-to-1 Gear Reduction Ratio

By ganging together—or multiplying—two 3-to-1 gear reductions, a 9-to-1 output reduction can be achieved. The key is to use intermediary shafts that hold large input gears (e.g., a 24-tooth) and small output gears (e.g., an 8-tooth).

to produce a 9-to-1 reduction, by using a shaft that holds a 24-tooth input gear and an 8-tooth output gear.

The gear ganging concept is the foundation of gear trains. Figure 6 shows a model LEGO gear train that produces a 243-to-1 reduction from the motor shaft to the output wheel. The example is a bit of overkill—this much reduction will produce too slow a final rotation for the typical robot drive train—but it serves to illustrate the point.

When I present gear reduction to kids, I find it difficult to give a direct explanation of why it works. More precisely, by counting teeth it’s evident enough that subsequent gears run slower, but why do they have correspondingly more torque? I generally appeal to a vague “energy must be conserved” line of reasoning. Ultimately, there’s no substitute for holding a live geartrain in your hand and feeling the power as gear reduction does its work.

The Worm Gear

The worm gear is a fascinating invention, sort of a Mobius strip in the world

When the 8-tooth gear rotates 3 times, it advances the meshed gear by a total of 24 teeth. Since the meshed gear has 24 teeth, it rotates exactly once. Hence, this configuration produces a 3:1 ratio of gear reduction: three turns of the input gear causes one turn of the output gear.

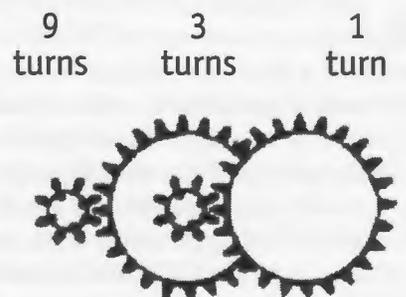


Figure 5: 9-to-1 Gear Reduction with Ganging

of gears. When meshed with a conventional round gear, the worm creates an n-to-1 reduction: each revolution of the worm gear advances the opposing gear by just one tooth. So, for example, it takes 24 rotations of the worm to revolve the 24-tooth round gear once. This forms quite a compact gear reduction—it would take about three gangs of the 3-to-1 reduction, forming a 27-to-1 relation, to do the same work as a single worm meshed with a 24-tooth round gear.

There is a drawback, however. The worm gear uses predominantly sliding friction when advancing the teeth of the round gear. The teeth of round gears are generally designed to minimize sliding effects when they are meshed with each other, but there's no getting around the problem with worm gears.

Thus worm gears create more frictional losses than round gears. At higher torques they have a tendency to cause a geartrain to stall. If your robot's too heavy, a worm gear drive may not work well as its main drive.

Worm gears have another interesting property: they can't be back-driven. That is, if you rotate the gear being driven by a worm, you'll just push the worm gear forward and back along its axle, but you won't get it to turn. Take advantage of this property. For example, if a worm gear is used to raise an arm lifting a weight, then the arm won't fall down after power is removed from the motor.

Figure 7 shows how to mesh a worm gear to a round gear, and Figure 8 illustrates putting two LEGO worm gears onto the same shaft.

Changing Axis of Rotation

In a geartrain with only round gears, all of the axles must be mutually parallel. With the worm gear, the output round gear's axis of rotation is at right angles to the worm's. Two other kinds of gears, the crown gear and bevel gear are available in the LEGO kit for changing the axis of rotation within a geartrain.

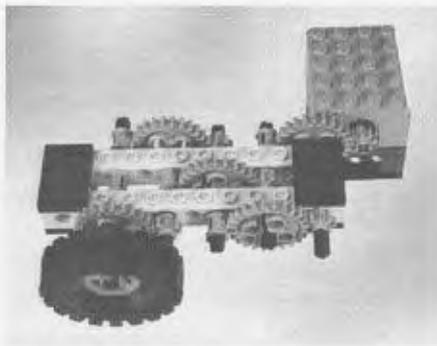


Figure 6: Sample LEGO Geartrain

The worm gear is valuable because it acts as a gear with one tooth: each revolution of the worm gear advances the round gear it's driving by just one tooth. So the worm gear meshed with a 24-tooth gear (as pictured) yields a whopping 24-to-1 reduction. However, the worm gear loses a lot of power to friction, so it may not be suitable for high performance, main drive applications.

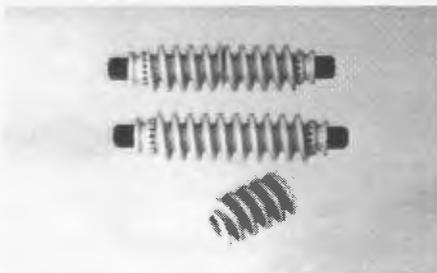


Figure 8: Multiple Worm Gears on One Shaft

The 8-tooth gear, in conjunction with the 24-tooth crown gear, is used to change the axis of rotation in a gear train. In this instance, the configuration provides for a vertical shaft output. Horizontal output also possible.

The Crown Gear

The crown gear is a round gear that is specially designed to mesh at right angles to the standard round gear (Figure 9). In the diagram, the crown gear is shown meshing with the 8-tooth gear. Meshing to the round 24-tooth and 40-tooth gear is also possible, though using the 8-tooth to drive the

A five-stage reduction using 8- and 24-tooth gears creates a 243-to-1 reduction in this sample LEGO geartrain. Note the need for three parallel planes of motion to prevent the gears from interfering with one another. Four 2 x 3 LEGO plates are used to hold the beams square and keep the axles from binding.

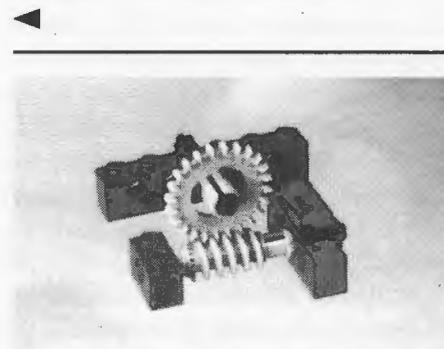


Figure 7: Using the Worm Gear

This diagram shows an arrangement of worm gears. At the bottom is the basic worm gear, two horizontal LEGO units in length. At the top is an unsuccessful attempt to put two worm gears on the same shaft. In the middle is the successful attempt. When placing multiple worm gears on a shaft, the trick is to try all four possible orientations to find the one that works.



Figure 9: 8-Tooth Gear Meshing with Crown Gear

24-tooth crown gear is an effective way to build in a gear reduction while changing the rotation axis.

The 24-tooth crown gear is the same size as the standard 24-tooth round gear, so it can be used as a replacement for that gear when a parts shortage occurs.



Figure 10: The Bevel Gear

The Bevel Gear

The bevel gear is used in pairs to provide a similar function to the crown gear, though without the capability for gear reduction. There are two styles of bevel gear: the older style (Figure 10), which is fairly flat, and a newer style, which is the same diameter but thicker. The old style bevel gear is somewhat flimsy and lossy and is not suitable for delivering larger torques. The new bevel gear is a significant improvement.

Old-style bevel gears can be put to good use by serving as stop bushes (Figure 15).

The Gear Rack

The gear rack is a like a round gear unrolled and laid out flat. When driven by a round gear (the 8-tooth usually works best), it traverses back and forth in a linear motion (Figure 11).

Gear racks can be laid end-to-end to make longer stretches of motion. Underneath a beam driven by gear racks, use the smooth-topped LEGO plates as a surface for the beam to slide on.

Practical Hints

For the remainder of this section on LEGO gearing, I'll present a number of assorted tips to assist in your geartrain designs.

Gear Sizing

It is helpful to know the sizes of the standard gears. This links back to the earlier section on the LEGO dimensional relation-creating unit horizontal spacings in the vertical dimension can be used not only to lock structures into place, but to mesh gears properly above and below one another.

The bevel gears are used to change the angle of rotation of shafts in a gear train with a 1:1 ratio. In this case, they are used to effect a change in the horizontal plane. This picture shows the older-style bevel gears, which have limited usefulness due to their relatively high friction and lack of strength. The newer bevel gears are thicker and perform much better.

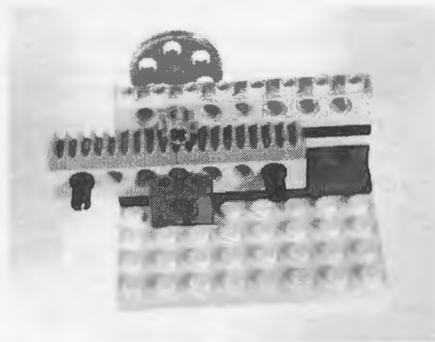


Figure 11: Using the Gear Rack

The 8-tooth, 24-tooth, and 40-tooth round gears all mesh properly along a horizontal beam because they have "half unit" radii. For example, the 8-tooth gear has a radius of 1/2 LEGO units, and the 24-tooth gear, 1 1/2 units, so they mesh at a spacing of two horizontal LEGO units.

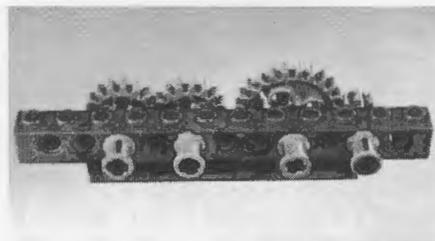


Figure 13: The 16-Tooth Gears

It seems like an obvious point, but using the 2x parts to hold beams parallel is quite important when the beams will be carrying common axles. If beams are not held squarely, the axles will bind and freeze inside the beam-hold bearing supports.



The gear driving the gear rack is often referred to as the "pinion," as in "rack-and-pinion steering," which uses the transverse motion of the gear rack to orient wheels. The 8-tooth gear is a good candidate to drive the rack because of the gear reduction it achieves—one revolution of the gear moves the rack by eight teeth.

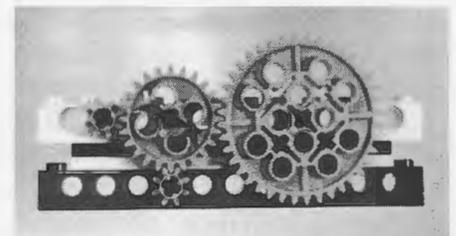


Figure 12: The Half-Radius Round Gears

The 16-tooth gear has a radius of 1 LEGO unit, so two of them mesh properly together at a spacing of two units (left side of diagram). Since an 8- and 24-tooth gear also mesh at two-unit spacing, these respective pairs of gears can be swapped for one another in an existing geartrain—a handy way to change the performance of a geartrain without rebuilding it from scratch.

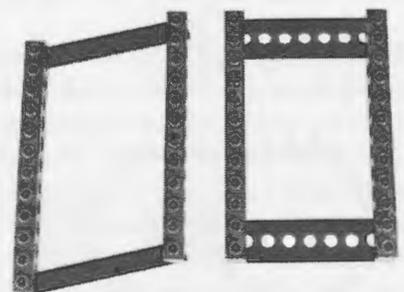


Figure 14: Locking Parallel Beams Together with 2x Parts

The standard 1-LEGO-long stop bush (upper axle, front) is not the only part that can act as a bushing (axle holder). Use the small pulley wheel (middle axle) to act as a half-sized spacer—it also grabs tighter than the full bush. In a pinch, the bevel gear (upper axle, back) makes a great bushing. Finally, the nut-and-bolt parts (lower axle) can be used to make a tight connection (if you can find them).



Figure 15: The Stop Bushes and Other Parts

There are three sizes of pulley wheel: the tiny one, which doubles as a stop bush; the medium-sized one, which doubles as a tire hub; and the large-sized one, which is sometimes used as a steering wheel in official LEGO plans.

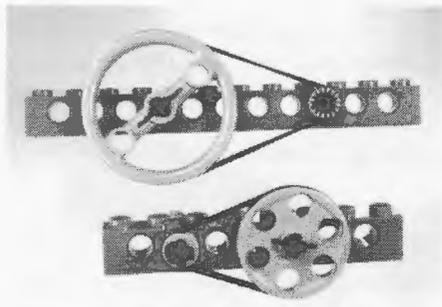


Figure 16: Using Pulley Wheels

Chain link can be an effective way to deliver large amounts of torque to a final drive, while providing a gear reduction if needed. Chain link works best at the slower stages of gearing, and with a somewhat slack linkage. Use the larger gears—the 8-tooth one won't work very well.



Figure 17: Chain Link Drive

On occasion it is necessary to lock a beam to an axle. This figure shows how to use a medium pulley wheel, which rigidly locks to an axle, to hold the beam in place.

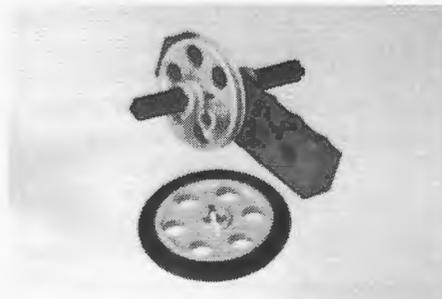


Figure 18: Axle Locked Through Beam Using Pulley Wheel

The special “gear mounter” piece is an axle on one side and a loose connector peg on the other. It can be used to mount gears used as idlers in a gear train—used simply to transmit motion or to reverse the direction of rotation.



Figure 19: The Gear Mounter Part



Of the four round gears, three of them—the 8-tooth, the 24-tooth, and the 40-tooth—have a radius that is a whole number of horizontal LEGO units plus one-half of a unit. Therefore, these three gears form an whole-unit spacing when their radii are added (i.e., when they are meshed together in a geartrain).

For example, the 8-tooth gear has a radius of one-half of a unit, and the 24-tooth, 1 1/2 units, so when properly meshed together, their centers are spaced at two horizontal units. A spacing of two horizontal units is readily available on LEGO beams, or can be constructed using the 1 2/3 vertical spacing relationship discussed earlier. Figure 12 shows how the three half-radii gears mesh with one another.

The example shows the 8- and 24-tooth gears meshed horizontally at two units, and, using the 1 2/3 vertical spacing trick, vertically as well (a common and useful configuration).

The 16-tooth gear, on the other hand, has a radius of one LEGO unit, so it meshes properly only with itself (at the standard horizontal unit spacing). A pair of two 16-tooth gears thus requires a space of two LEGO units, which happens to be the same interval as the pair of an 8-tooth and a 24-tooth gear. Thus these respective pairs of gears may be easily interchanged—a useful trick for adjusting the performance of an existing geartrain without a performing major overhaul (Figure 13).

Odd Gear Spacings

It's possible to mesh gears at odd spacings using diagonal mounting. Generally, the gears work better when slightly too far apart than when too tight, which causes them to bind.

Many combinations are possible when creating a space diagonally, and some of them work. For example, an 8- and 16-tooth gear will function when spaced along the diagonal of one horizontal unit and one vertical unit.

An interesting exercise is to calculate the effective spacings (in horizontal units) of various diagonal measures using the Pythagorean formula, and

then see which come close enough to a pairing of gears to be useful. But I suggest experimentation.

Supporting Axles

It's important to keep axles well-supported in a geartrain. Practically, this means using at least two beams to carry the axles. More importantly, all beams with common axles running through them must be held together squarely. If the beams not held together, the axles will bind and lock up inside the beam-hole bearing mounts.

As suggested in Figure 14, use the 2x parts, not the 1x parts, to hold beams in place. Figure 6, the sample geartrain, uses 2x3 plates to squarely support the beams.

Using Stop Bushes

The stop bushes, or axle holders, are to keep axles from sliding back and forth in their mounts. In addition to the standard full-width stop bush, the small pulley wheel and the bevel gear may be put into service (Figure 15).

Reducing Noise with Pulley Wheels

Sometimes a geartrain will be quite noisy. Usually most of the noise is generated by the very first meshing of gears from the motor. Here is the ideal place to use a pulley wheel drive (Figure 16).

Use the small pulley wheel on the motor shaft, and the medium or large pulley wheel on the driven shaft. The ratio of the circumferences of two pulleys creates a reduction just like the ratio of the gear teeth of a pair of meshed gears.

LEGO pulley drive belts—thin rubber bands—are best when used in high-speed, low-torque situations, because they can't transmit a lot of force. So the first stage is really the best place to use a pulley drive.

Be careful, though, about using pulleys in a competitive situation. They have a penchant for breaking or falling off at the most inopportune moment.

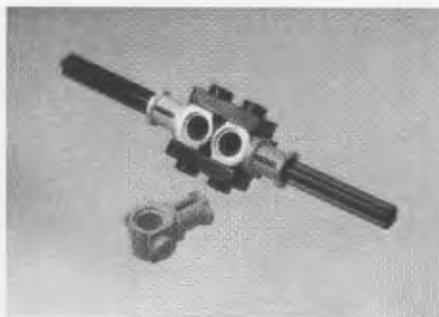


Figure 20: An Axle Joiner

In order to build outward from a vertical wall of axle holes, a smaller beam may be mounted with its top studs in the holes of the beam wall. You will not see this configuration in LEGO's model plans, because the top studs are slightly too big for the axle holes, and a model left in this state will gradually experience solid flow as the stressed plastic expands. The official LEGO solution is to use the "connector peg with stud" parts (see Figure 22), but this method is actually stronger (or at least until the LEGO parts deform).

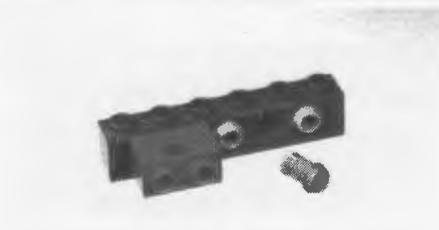


Figure 22: The Connector-Peg-With-Stud Part

The full-size stop bush can be used in one orientation to hold an axle through a plate hole so that the axle can freely rotate. In Figure 24, an additional plate is used to trap the axle, but allow it to rotate freely.

This configuration of parts can be used as a compact axle joiner. LEGO now produces a part designed for this purpose, but in lieu of that part, this is a useful trick.

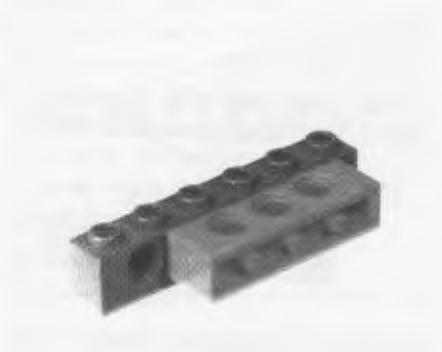


Figure 21: One Method of Building Outward from a Vertical Wall

The recommended way to build outward from a beam wall is to use the connector-peg-with-stud piece, which is a loose-style connector peg on one end and a top stud on the other. This method is somewhat weaker than the method of simply plugging top studs into axle holes (Figure 21) but will not deform the plastic.

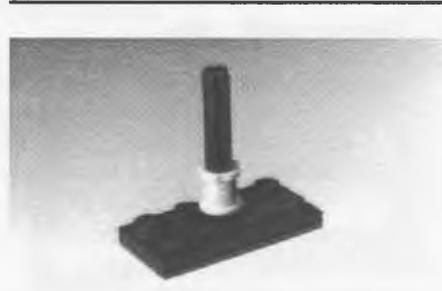


Figure 23: Using a Stop Bush to Retain an Axle



Figure 24: Trapping an Axle Between Two Plates Using Stop Bush

In the other orientation, the stop bush locks between four top studs, perfectly centered over the axle holes in flat plates. This allows the stop bush to lock a plate to an axle.

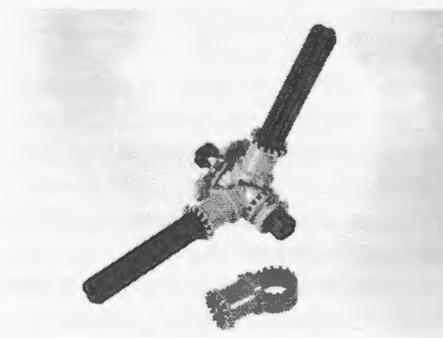


Figure 26: Two Toggle Joints

Here the toggle joint is used to connect two axles at right angles. The small pulley wheel is deployed on the axle that runs through the toggle joint to either lock the axle or allow it to rotate.

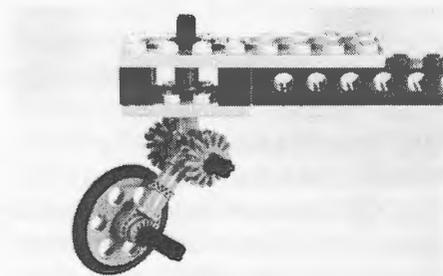


Figure 28: A Caster Design

By using the stop bush to hold an axle in place between two plates, a vertical axle mount can easily be created. Depending on the orientation of the stop bush, it can be made to either lock the axle in place or allow it to rotate freely. In this diagram, the axle is allowed to rotate.



Figure 25: Using a Stop Bush to Lock an Axle to a Plate

The “toggle joint” can be used to lock two axles at a variety of odd angles. The short axle running through the two toggle joints is equipped with stop bushes on either end to hold the joint together.

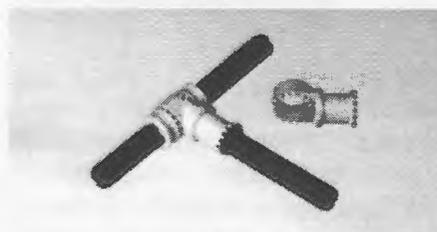


Figure 27: Toggle Joint With Free or Locked Axle

Several clichés are used to construct this caster wheel. The vertical axle is trapped between two plates, but allowed to rotate, using the trick shown in Figure 24. The angled joint down to the wheel is done using toggle joints in the configuration suggested in Figure 26, and the final mounting of the wheel is done using the toggle joint per Figure 27.

Chain Link Drive

Chain link drives are best suited for the final stage of a geartrain—transmitting power down to the axles holding the wheels, for example. This is because they can easily deliver the necessary torques, and they impose frictional losses that are minimized when rotational speeds are low.

Getting the right amount of chain links can be tricky. Generally, a looser chain works better—chains that are too tight will bind up. But too loose a chain will skip when the going gets tough.

Design Strategy

When designing a new geartrain into a model, I find it best to work backward from the final drive, rather than forward from the motor. This makes sense because usually there is a fair bit of flexibility about where the motor is ultimately mounted, but much less in the placement of drive wheels or leg joints (for example).

So start by mounting the axle shaft that will carry the final drive, put a wheel and gear on it, and start working backward, adding gearing until there is enough, and finally mount the motor in a convenient spot.

When designing a vehicle, don't forget about the role of the tire in determining the relationship between the rotational speed of the final drive axle and the linear speed that is achieved. Small tires act as gear reductions with respect to large tires, and this may have an effect on how much gear reduction is necessary. Experiment!

If a geartrain seems to be performing badly, there are a few things to check. Make sure the stop bushes aren't squeezing too hard—there should be some room for the axles to shift back and forth in their mounts. Check that all beams holding the axles are squarely locked together. The most common cause of poorly performing geartrains is beam mounts that aren't square.

To test a geartrain, try driving it backward. Remove the motor, and gently but firmly turn the final drive wheel or shaft. If there isn't too much friction,

all of the gears in the train will start moving, with the motor's gear spinning around rapidly. If your geartrain can be readily back-driven, it's a sure sign that it's performing well.

LEGO Design Clichés

This section presents a miscellaneous assortment of ideas in a visual fashion. I've come to call these LEGO ideas "clichés" because I hope that they become common, everyday knowledge, rather than secrets held by some small group of LEGO experts. I find myself inventing them time and again, on the spot, when working with kids helping them with their LEGO designs. Part of my intention in writing this article is to collect these clichés and share them with others.

Browse through this collection and perhaps you will find one or more of these techniques useful in your own LEGO designs.

Closing

I hope that this article inspires others to contribute LEGO design clichés from their own vocabulary to a larger collection of resources for LEGO builders. By the time this article is printed, we will open a World Wide Web site representing successive versions of our LEGO Constructopedia, presenting these ideas in a hypermedia format and soliciting the contributions of others.

Acknowledgments

Steve Ocko guided my early work with the LEGO Technic system, and continues to inspire me with his LEGO creativity and his dedication to developing rich learning environments for kids. Mike Plusch and Randy Sargent, both LEGO geniuses, shared numerous ideas and insights with me, which I've incorporated into this article.

The LEGO Group is a continuing sponsor of our work at the Media Laboratory. They have provided both generous research funding and valued intellectual discourse on a wide range of topics related to children's learning and effective use of the LEGO materials.

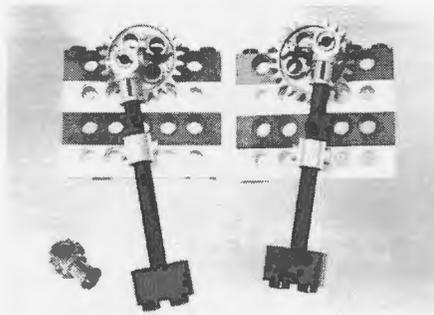


Figure 29: LEGO Legs

How many of the design clichés can you find in this robot? Look especially for the vertical spacing trick used to provide rigidity to the robot structure. The robot, a ping-pong ball collector, was designed by the author and Brian Silverman. Sitting on top of it is the Programmable Brick, a robotics controller for kids recently developed by the author and his colleagues at the MIT Media Laboratory.

The "piston rod" part (shown in the left foreground) is used twice in each mechanism to create a LEGO leg. By using a chain drive or gear linkage to lock legs in sync, a multi-legged creature can be designed.

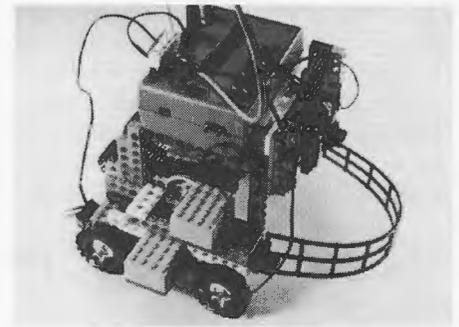


Figure 30: LEGO Ping-Pong Ball Collecting Robot

We are deeply grateful to their support of our work. Any criticism I make about their products should be taken in the spirit of making a great thing even better.

About the Author

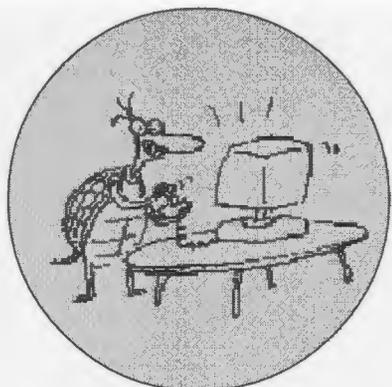
Fred Martin has been developing educational robotics technologies at the MIT Media Laboratory since 1986, and is a co-founder of the annual MIT LEGO Robot Design Competition. Fred is interested in robotics for fun and as a vehicle for exploring sensing, control, and engineering design. He has been teaching robotics to kids, teenagers, college students, and adults for nearly 10 years and hasn't gotten tired of it yet.

Currently a Postdoctoral Fellow at the Media Lab, Fred is working on a robotics textbook aimed at undergraduate engineering students. Tentatively titled *The Art of Robotics: A Hands-On Introduction to Engineering*, the book is scheduled to be published in the fall of 1996 by Benjamin/Cummings. This article will form the basis of an

expanded chapter on LEGO design in the text.

Fred welcomes any comments on this article. He can be reached at fredm@media.mit.edu. The URL for the LEGO Constructopedia Web site is: <http://el.www.media.mit.edu/groups/el/Projects/constructopedia>

Fred Martin
MIT Media Laboratory
fredm@media.mit.edu
LEGO Constructopedia Web site:
<http://el.www.media.mit.edu/groups/el/Projects/constructopedia>



The ToonTalk and Playground Presentation at Logosium

by KEN KAHN, Animated Programs

At the Logosium I had the privilege of presenting Toontalk (www.toontalk.com) and the Playground Project (www.ioe.ac.uk/playground) to audiences ranging from the curious to Logo old-timers (including Seymour Papert and Cynthia Solomon). I began by trying to motivate ToonTalk as my attempt to have the best of computer programming and video games.

Programming is such a creative open-ended activity but is hard to learn while video games have tremendous appeal and are very learnable. The idea behind ToonTalk [so called because you are “talking” to the computer in (car)toons] is to create a video game world inside of which you can do real full-blown programming.

There are several ways one can learn ToonTalk besides being taught by a human expert. One is to explore the game world. You start off in a helicopter and like most video games the controls are easy to discover.

Soon you have landed and are walking around. You see Tooty, a tool box character, following you around. You enter a house and are greeted by Marty the Martian. Marty welcomes you and explains that he'll stay around to help. (Marty can speak or use text balloons or both.) You sit down and Tooty opens up and you find numbers, text pads, boxes, nests, robots, trucks, bombs, and scales inside. Also Dusty the Vacuum,



Ken Kahn (far left) presents ToonTalk to a star-studded audience

Pumpy the Bike Pump, and the Magic Wand come out. You can experiment with these objects (with or without Marty's help). Eventually you learn how to train robots to accomplish nearly any task, to give birds messages to deliver, the load up trucks to construct new houses, etc. You do real programming, not by typing text or assembling a diagram, but by working through concrete examples in an animated fashion and then generalizing.

This kind of exploration works well for some children, but for many some-

thing else is needed. The most passive, least constructionist, alternative is to sit back and watch movies demonstrating how things work and some neat programming techniques. ToonTalk includes 10 such demos with narration and subtitles. These demos range from an introductory tour to how to compute factorial or Fibonacci numbers, to how to do something silly like build a town of exploding houses, to implementing a bank account, to building a Ping Pong game.

The most unusual way to learn

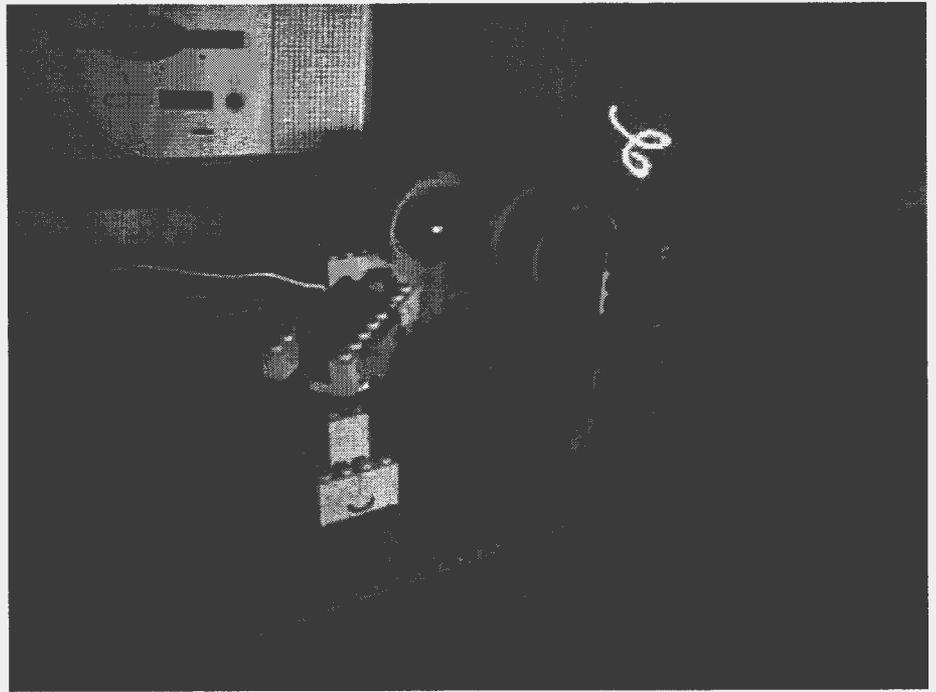
ToonTalk is to play the puzzle game. At the Logosium I chose to introduce the audience to ToonTalk this way when I found a child in the audience who was willing to volunteer to play the puzzle game. The boy had never seen ToonTalk before and I wanted to give the audience a sense of how a child could learn a lot about ToonTalk and programming on his own. The puzzle game starts with a picture book background story. Marty has crashed after rescuing houses on a sinking island. He is hurt and needs your help to fix his ship. The first level (17 puzzles) deals with trying to get numbers working again in the ship's computer. Later levels focus on manipulating words, time, sensors, among other things.

The boy (unfortunately I've forgotten his name) started with the first task, which was to construct a data structure with 1 and 2 in it. (Like [1 2] in Logo.) He goes the house next door and sees a box with 2 holes and the number 1 and 2. All he needed to do here was to figure out how to pick up and drop objects. The next puzzle requires a 4 and he was given a 2 and a 2. He quickly guessed that to add them he needs to drop one number on the other. The puzzles increase in difficulty such that by the end of the first level the player has trained a robot to construct a box with powers of 2 in it. This is roughly equivalent to

```
TO POWERS :N
OUTPUT SENTENCE :N POWERS
:N+:N
END
```

The equivalent ToonTalk program terminates when the number become too large for the computer, leaving the box with powers of two on the floor. (Later puzzles introduce termination conditions.)

The demo by the boy went very well and he was able to make good progress without help from me or the audience. My only worry about having spent so much of my presentation having a complete ToonTalk beginner demo the



An amazing laser project built out of LEGO by one of Logosium's student presenters

puzzle game is that some of the audience might have left with an impression that the puzzle game is the important thing about ToonTalk. It is only one way to learn ToonTalk. The important thing is that children can do real programming while inside a game world.

As time was running out, I discussed the Playground Project. This is a three-year research project funding by the European Union. Its partners include the Institute of Education at the University of London, the University of Stockholm, Logotron (www.logo.com), Comenius University, and Cnotinfor, Coimbra, Portugal. The goal of the project is to enable children to "play with rules" by constructing video games. Games can be constructed by modifying existing games, pulling multiple games apart and combining them in unique ways, or by assembling them from components. The components are open in that they can be inspected, edited, or even made from scratch by children. The project is currently working with children from 6 to 8 years old. They are building two Playgrounds—one on top of ToonTalk (hence my involvement) and another on a brand new Logo called Open Logo

being built in Slovakia. The project has made very good progress in its first year. I urge you to visit www.ioe.ac.uk/playground to learn more. LX

About the author

ToonTalk was designed and built by Ken Kahn who, after earning a doctorate in computer science from MIT, spent 20 years as a researcher in programming languages, computer animation, and programming systems for children. He has been a faculty member at MIT, the University of Stockholm, and Upsala University. For more than eight years, he was a researcher at Xerox PARC. He has made several animated films that have been shown in film festivals, theaters and on cable TV. In 1992, Ken founded Animated Programs with the mission to make computer programming child's play. He received a patent covering the underlying technology of ToonTalk (US Patent Number 5,517,663).

Ken Kahn, Animated Programs
kenkahn@toontalk.com
www.toontalk.com



STARTING WITH STARLOGO

StarLogoT2000 Released

by URI WILENSKY

The StarLogoT development team at the Center for Connected Learning and Computer-Based Modeling (CCL) is pleased to announce a new release of StarLogoT, StarLogoT2000. Like our previous releases, this version of StarLogoT is available only for the Macintosh.

A multi-platform Java implementation of StarLogoT is under development here at the CCL. The new version, called N-Logo, is now in alpha test. N-Logo will run on PCs, Macs and Unix machines. We are sorry that the N-Logo release has been delayed. We expect to distribute a beta version of N-Logo in first-quarter 2000. N-Logo will include all StarLogoT features and add many significant new features. StarLogoT models will be automatically converted when opened in N-Logo.

Due to the extensive feedback we received from users of StarLogoT2.0, we have worked hard to enhance the documentation. We have made many improvements to the HTML reference manual and Apple quick guide. The StarLogoT Development Reference manual has many new sections. We have also added a library of short code examples that function as mini-tutori-

als. StarLogoT2000 also introduces new features and sample models. We expect that these changes will make StarLogoT2000 the easiest to use and most powerful version of StarLogoT yet.

This release of StarLogoT2000 is, nominally, a beta release as we plan to extend it in the spring. We do recommend, however, that all users upgrade to StarLogoT2000 at this time. It has many significant improvements and, in our testing, it has been at least as stable as StarLogoT2.0.

New Features

- Turtle, patch and observer arrays. Arrays are now fully supported in StarLogoT.
- Turtle and patch command centers. These command centers are located inside turtle and patch monitors. These command centers allow the issuing of commands to one turtle or patch at a time. Monitors are divided in half horizontally. The lower section is a command center for its associated individual turtle or patch. This new feature can be of great value in testing and debugging.

- A large library of code examples that illustrate the use of many of the StarLogoT primitives and idioms.
- A “Read Info Window” button (in green) is now located at the bottom of the interface window. This button links directly to the open model’s info window.
- Considerable enhancements to the documentation, both the HTML User’s Guide and the Apple Quick Reference.
- A glossary to accompany the primitives documentation in the User’s Guide.
- A revised set of sample models.
- More than 30 new models in the Connected Models package: www.ccl.tufts.edu/cm/models
- Turtle and patch versions of `< slider-variable > -max`, `< slider-variable > -min`, and `< slider-variable > -inc`. Previously, these were accessible only to the observer.
- New commands: “ask-turtles” and “ask-patches”. These commands are provided for code consistency with breed syntax and for forward code migration.



The Logo Foundation's Michael Tempel makes last-minute preparations for his Game Design presentation

New Bug Fixes

- Fixed bug with importing patches that caused patch variables to become misaligned.
- Fixed bug that caused the diffuse command to overflow.
- Fixed bug that caused occasional listener error windows to pop up when histogramming.
- Fixed bug that caused histograms set to the color black to render incorrectly.
- Graphs using auto-plot no longer have the tops of their plots clipped.
- Fixed bug that caused some switch values to be overwritten.
- Fixed bug that caused observer random seeds not to be correctly reset.
- Fixed Apple Guide formatting errors.

To download StarLogoT2000, visit our Web page: www.ccl.tufts.edu/cm and click on the StarLogoT button.

To provide feedback to the development, documentation and model building team, please send a message to: complex-feedback@listproc.tufts.edu

To report bugs please send a message to: bug-StarLogoT@listproc.tufts.edu

Included Models

StarLogoT2000 is distributed with two folders of models.

The Sample Models folder contains a set of models that are exemplary and demonstrate good coding practice. This package is a subset of the entire Connected Models package. The entire set of Connected Models isn't distributed with this release, but is available for download at: www.ccl.tufts.edu/cm/models/

The Code Examples folder contains a library of models that are intended to demonstrate common StarLogoT idioms. These are very simple models that can serve as an introduction to StarLogoT and as building blocks for your models.

Models available on the Web

The Connected Models package has many new models such as: cooperation, altruism, divide the cake, gas chromatography, voting, random billiards, income distribution, bank reserves, mammoths, percolation, sphere, boiling and wandering letters . . . It also includes updates to old models such as slime, speakers, wolf-sheep predation, rumor mill, planar transformations, ants, follower, firefly, nuclear reaction,

painted desert challenge and many others. To download the entire Connected Models package or to download individual models, visit the StarLogoT models page: www.ccl.tufts.edu/cm/models/

We also invite you to contribute your own models to our Contributed Models page. You can access the Contributed Models page at the top of the Connected Models page: www.ccl.tufts.edu/cm/models/. It will ask you for a username and password. The username is StarLogoT and the password is StarLogoT. At this page, you can also browse and download models contributed by other users. Please feel free to use this community exchange in any way you like. Models do not have to be polished or finished to be uploaded. Please also feel free to ask members of the StarLogoT community as well as members of the CCL staff to comment on your models.

Our Web page (www.ccl.tufts.edu/cm) as well as the models page (www.ccl.tufts.edu/cm/models) will be getting frequent updates over the next few months, so keep those bookmarks.

If you are a member of the complex-users mailing list, we will continue to send you notices of updates and future releases.

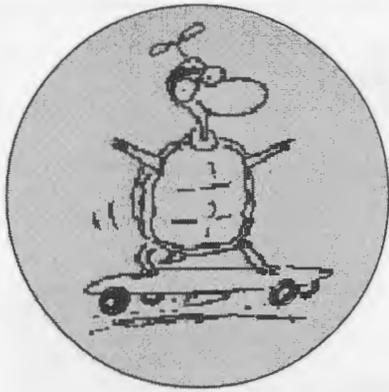
If you would like to subscribe to complex-users, send a message to: listproc@listproc.tufts.edu

The body of the message should say: subscribe complex-users your-first-name your-last-name

To be removed from the complex-users announcement list, send a message to: listproc@listproc.tufts.edu

The body of the message should say: unsubscribe complex-users **LX**

**Professor Uri Wilensky, Director
Center for Connected Learning
and Computer Based Modeling
www.ccl.tufts.edu**



The Y2K Explorer

by PATRICK MAHANEY

Some three years ago, my colleague and I built a car that consisted of a Lego chassis, and a color QuickCam. It was a good idea and broke new ground because of its tilt-and-rotate features but ultimately slow and could not drive on much more than a wood floor. Thrilling as it was, we wanted more out of it. It lacked the performance necessary to do *real* exploring. (The world isn't flat after all, and neither are the other planets.) By this time, Patrick already had a high-performance off-road R/C car. R/C stands for Remote Control. This means the vehicle can operate without being physically connected to a control unit. Instead, it uses radio waves. By now, my other colleague, Moris Behar, took an interest in the project to create a better explorer using my R/C car. The problem was how to include all of the other bells and whistles of the previous car.

After a month or so of hard after-school labor, our new explorer was born. As mentioned before everything sits on my R/C car. This enables it to travel fast, and have all the power it needs to climb hills, and deal with all sorts of debris in its path. Next, we mounted our newest investment: a grayscale pinhole camera. This is a very small but high-resolution camera.



Patrick (left) with Dr. Seymour Papert (right) and friends at Logosium '99

The entire thing is about half the size of a credit card, and weighs about the same. This we attached to our own Lego chassis on top of the car. The camera was mounted to a rotating "swing" which allowed the camera to tilt up and down.

From here, we decided to make our invention even better. The off-road car part with the tilt camera was good, but it still lacked something. . . . A view! Normally, the camera would only be able to see at ground level. This makes

it difficult to see the road hazards ahead. The answer would be to have the camera elevate it's position. To do this, we engineered what looks to be a crane boom onto the car. It consists of an arm about the length of the car, which can be raised perpendicular to the ground. From this position, the camera can still tilt, and the car can still drive without concern for it flipping over. The biggest problem with the elevating arm was getting the power to move it up. A regular motor

would need an immense amount of gear reduction to move it. The answer to our problem was pneumatics. Pneumatics are relatives of hydraulics which are seen on many construction vehicles that make the "joints" move. Instead of using oil in hydraulics, pneumatics use air pressure.

Our car has two pneumatic rams with an onboard compressor. A motorized valve regulates the amount of air allowed into each ram. This way we have quick power to elevate the arm, and a controlled decent back to ground level.

All of the Lego motors are once again powered through the Lego Dacta Control Lab. This means that they are user-friendly on screen controls (which we programmed) to control the Lego parts of the roamer. Every part of the Lego chassis (camera, pneumatics) can be controlled on screen. We even made a program that allows a user to program various features of the car without knowing anything about the logo language. (We strive for simplicity.) The more advanced features include direct keyboard controls (much like a game) for the serious user.

The only setback to this entire project is that the car must drag about 50 feet of ethernet cable behind it. This makes it possible to control the motors. There are just so many functions that this car does, it would be difficult and costly to make the car truly wireless. LX

About the Author

Patrick Mahaney is a student at the High School For Environmental Studies in New York City. He has designed and built many projects for his school as well as in his free time. He has presented his work at Parent-Teacher Conferences, the Museum of Natural History in New York City, and twice at the UFT "Kid Connections" Conference. He has also written an article for Logo Exchange, in the summer 1997. His most recent appearance was at the Logosium '99 conference in Philadelphia.

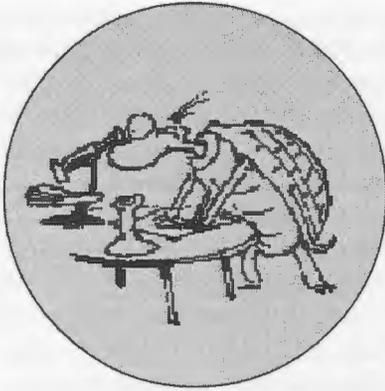
More memories from Logosium '99



Constructing with LEGO



Seymour Papert leads an inspirational discussion



AgentSheets

End-User Programmable Simulations as Interactive Media

by ALEXANDER REPENNING

What is the role of Logo or more generally of programming in education? Relatively cheap but powerful computers that exist today have enormous potential to revolutionize the way we learn. Nevertheless, at a time when the potential of information technology is captured with omnipresent diagrams of impressive exponential growth describing the number of Web hosts, computer speed, memory size, etc., one wonders why there is so little consequence to education. Information technology opportunities are typically limited in making education more efficient. Drill-and-practice tools, course management software, and distance learning Web sites with video contents certainly do have their educational role. However, this kind of technology contributes little to the constructionist vision of employing computers as active and engaging educational media.

Today the role of programming in education is unclear. There are numerous pragmatic reasons to keep the idea of programming as learning vehicle from spreading. In the eyes of many educators, Logo's visionary quest for "Mathland" has led to "Programmingland" instead. Without compelling evidence for transfer of programming to general problem-solving skills time

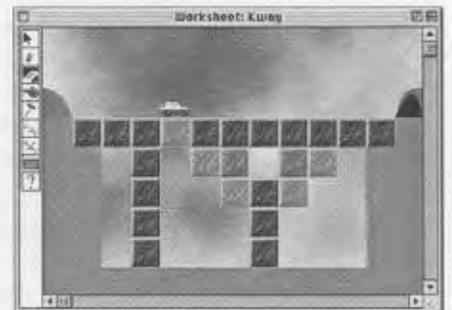
spent with information technology gets absorbed with search engine drills, word processing and maybe some Photoshop experience. Is the idea of programming as constructionist vehicle dead or relegated to a splinter group of kids attending Friday afternoon computer clubs?

I do not think that programming in education is dead, but there is a strong need to re-conceptualize what programming is, why it is used, and how it can be supported as educational activity from a technical as well as from a social perspective. End-User programmable simulations are very powerful interactive media, which—in the

sprit of constructionism—encourage learning through design activities. This article quickly introduces the AgentSheets® simulation authoring tool and discusses some end-user programming approaches from a philosophical Logo perspective.

AgentSheets: a Simulation Authoring Environment

The AgentSheets work started in 1989 with the goal of creating a versatile dynamic computational medium allowing a wide range of people to explore, comprehend and communicate complex ideas. From a more recent perspective AgentSheets can be conceptualized



AgentSheets application called the Bridge Builder. The objective of the Bridge Builder simulation is to design a bridge that requires the minimum number of bricks. By removing as many bricks as possible students learn about basic bridge design. The interactive aspect of this simulation as medium is that the bridge can be modified while the simulation is running. Static (bricks on bricks) as well as dynamic (cars on bricks) structural tension is visualized as red color feedback.

as an authoring environment to design interactive SimCity-like simulations and games. Interactive simulations are useful for a vast range of applications. For instance, interactive simulations can empower biologists to explore the growth of cells, network managers to analyze the flow of information, city planners to play through complex growth and pollution scenarios, and school children to experience the fragility of food webs.

AgentSheets application called the Bridge Builder. The objective of the Bridge Builder simulation is to design a bridge that requires the minimum number of bricks. By removing as many bricks as possible students learn about basic bridge design. The interactive aspect of this simulation as medium is that the bridge can be modified while the simulation is running. Static (bricks on bricks) as well as dynamic (cars on bricks) structural tension is visualized as red color feedback.

The main technologies upon which AgentSheets is based are agents, spreadsheets, and Java generators:

- **Agents:** Agents appear as small, movable pictures on the computer screen, and have associated behaviors that define how they interact with their environment. For instance, in Bridge Builder the car as well as the bricks are agents. The Logo equivalent of an agent is a turtle. Agent behaviors are end-user programmable, and AgentSheets features a number of end-user programming approaches (some described below), which allow users without a programming background to create agents with complex behaviors. Agents can be programmed to perceive mouse clicks, sound and keyboard input, the existence of other agents, attribute values, messages, and even get input information by querying Web pages (e.g., get the current temperature from a weather Web page). They can be programmed to move,

change their appearance, play sounds, play MIDI instruments, play video clips, compute formulae, speak text through voice synthesis, send messages to other agents, open up URLs, and do various other things.

- **Spreadsheets:** Agents are placed together in a grid, called an agent-sheet or worksheet, where they interact with one another. Similar to a spreadsheet, the grid structure of a worksheet enables spatial communication between the elements it contains. For instance, a bridge is simply constructed by placing bricks next to each other.
- **Java Generators:** Computational artifacts embedded into interactive media need to be easily shareable. AgentSheets includes the Ristretto™ technology. Ristretto provides a unique way for users without any Java background to instantly turn their simulations into interactive Web pages containing Java applets or JavaBeans. In contrast to other simulation authoring tools Ristretto generated simulation applets do not require any browser plug-ins or separate players. Simulations converted into JavaBeans can be combined with other educational components including spreadsheets, data bases, data plotters (e.g., SimCalc), and geometry tools (e.g., Geometers' Sketch pad). For more information on educational components, see ESCOT (www.escot.org).

End-User Programming and Logo

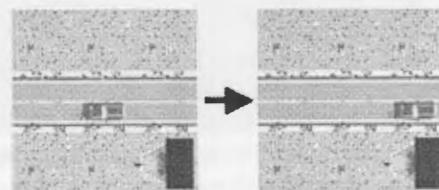
How are agents programmed? AgentSheets is similar to StarLogo but has a stronger emphasis on end-user programming approaches that explore new programming paradigms. For instance, AgentSheets pioneered the combination of graphical rewrite rules with programming by example. In the left figure, a car agent is programmed to keep

moving on a road when seeing a green traffic light simply by moving the car one road piece ahead. The result of this manipulation is stored as Before/After rule. The advantage of this kind of programming (later also adapted by KidSim/Cocoa/Stagecast) is that users do not need to have any kind of programming background since all they do is to interact with objects on the screen directly.

When tested in real classroom situations researchers including ourselves found the graphical rewrite rule approach used in AgentSheets and Cocoa not well suited for building simulations such as ecoworlds simulations. On the one hand, kids picked up basic programming skills remarkably fast compared to Logo programming. Within minutes and without any programming background they were able to make animals do simple things such as moving around. On the other hand, trying to move on and adding more reality to their simulations they quickly got completely stuck. With programming affordances strongly favoring the control of character positions the graphical rewrite rules are more conducive to build animations instead of simulations.

AgentSheets current programming language called Visual AgenTalk is still a rule-based language but it includes programming concepts found in more traditional object-oriented programming approaches to support the design of complex simulations and not just animations. Even so Visual AgenTalk has a low threshold allowing simple behaviors such as the cursor control of a pacman (Figure right) to be represented effectively.

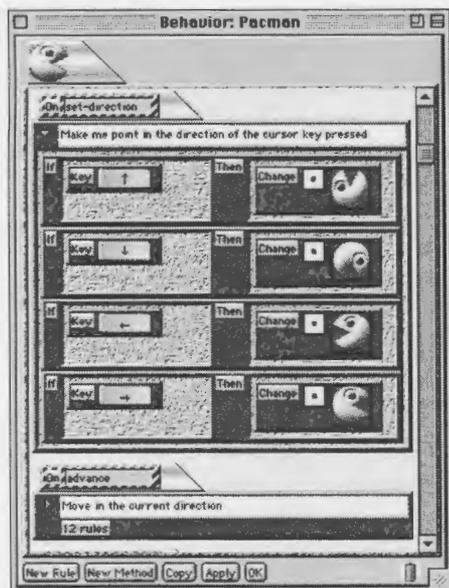
The user interface of the Visual AgenTalk programming language amplifies many aspects of the exploratory programming philosophy found in Logo:



Bricolage: The kind of bottom-up building process typically found in end-user programming is supported by what we call a tactile programming interface. In Visual AgenTalk each program fragment such as conditions, actions, rules, and methods can be tested at any time in any context. Will this condition be true for this agent? To test this a user simply drags the condition onto the agent getting feedback if the condition is true or not. The same holds true for rules and methods. For instance when a rule is dragged onto an agent the rule will be tested with audio/visual feedback shown in the rule. If the rule cannot fire then first false condition in the rule blinks indicating the reason the rule cannot fire. If all conditions are true each action will be executed step by step showing which action has what effect.

To test conditions and actions users do not have to write a complete program. They can drag and drop any condition or action from a condition/action palette directly onto an agent without first having to define any rule.

Each condition or action can have many parameters such as the direction in which an agent should move, and the color it should assume. Each of these parameters are direct manipulation widgets such as popup dialogs. At the same time they prevent users from making syntactic programming mistakes and illustrate valid parts of the design space.



More from Logosium '99



No trip to Philly would be complete with a lunch of pizza, grinders, and cheesesteaks!

Modifiability: At any point in time any aspect of the simulation can be changed including the look of the agents, the behaviors of the agents, and the arrangement of agents in the worksheet—even while the simulation is running.

Procedural Abstraction: A set of rules, called a method, is given a name by the programmer. This method name can be used anywhere, anytime to invoke the method. For instance the “set-direction” method of a Pacman game contains four rules testing if a cursor key is pressed. If so, appearance of the pacman is changed. The ability to define method names allows users to build their own language and abstractions of complex problems.

Social Constructionism: The Behavior Exchange is a Web-based forum supporting collaborative simulation design. New AgentSheets users can browse the catalog of agents and download interesting agents. More advanced AgentSheets users upload their own agents. The Behavior Exchange has been

used effectively in elementary schools to build Eco systems by teams of students.

The use of tactile programming in AgentSheets has been quite successful resulting in thousands of simulations built by an extraordinary range of users including elementary school kids as well as professional NASA scientists. I very much welcome a discussion of programming issues for kids programming and invite you to download a free demo version of AgentSheets at www.agentsheets.com.

Acknowledgements

AgentSheets funding includes NSF DMI-9761360, RED925-3425, DMI-9901678, and DARPA CDA-940860.

Dr. Alexander Repenning
President and CEO
AgentSheets Inc.
6525 Gunpark Dr., Suite 150,
Boulder, CO 80301
alexandr@agentsheets.com
www.agentsheets.com

RELEASE:

AgentSheets 1.4b3

We are happy to announce the release of AgentSheets 1.4b3:

www.agentsheets.com/download.html

This will be the final beta version of the software before the final release. The b3 version of AgentSheets available for download expires on February 28, 2000. It includes many new features and improvements some of which are outlined below:

1. **Simplified Authoring:** Twist Down Methods. Each method features twist down control to show or hide rules. This significantly simplifies the editing of complex behaviors containing a large number of rules.
2. **Documentation Stickies:** Methods can now be documented in the Behavior Editor with stickies.
3. **Better Simulation Control:** Simulation Properties are global properties that can be read and written by all agents. A Simulation property editor allows users to inspect and to change the values of simulation properties while the simulation is running. The Beans version of AgentSheets (at the time of this release only available to ESCOT testbed members) exports Simulation properties as JavaBeans properties. Ristretto™ generated JavaBeans can be combined with other components such as spreadsheets, plotters, graphers, etc.
4. **Web-Based Project Documentation:** Projects may include Web-based documentation. This documentation will be offered to users when project is opened in Macintosh application version, and when project is loaded as Ristretto™ Java applet in browsers. A simple documentation consists of a "readme.html" file provided by the user or can also include a "documentation" folder contained in the project folder. The "documentation" folder, in turn, may contain any kind of acceptable web content including HTML files, images, etc. Ristretto™ copies documentation into the generated Web pages to be included with the Java applet.
5. **Faster Dispatcher:** The AgentSheets agent dispatcher spends even less time with overhead activities. The result is that simulations run between 10% and 20% faster than in AgentSheets 1.4b2.

For a complete list of new features and fixes, please refer to

www.agentsheets.com/download/update14b3.html

AGENTSHEETS 1.4b3 CD

You can now order the AgentSheets = AE 1.4b3 CD with over 200 MB of interactive tutorials, manuals, movies, many more projects, Ristretto applets, language extensions, third-party contributions, and research papers. The b3 version of AgentSheets on the CD expires on April 30, 2000. As with the 1.4b2 CD, the content would take more than nine hours to download with a 56k modem. The CD is also Windows 95/98 readable: Applets, manuals, and tutorials also work in Windows. The Windows version of the AgentSheets authoring software is not yet released and not on the CD.

Content of this CD:

www.agentsheets.com/AS1_4b3CD_contents.html

Order Information:

www.agentsheets.com/download/order-form.html

MACWORLD 2000: AgentSheets Inc. Announces the official release

AgentSheets Inc. will be announcing the official release of the final version of AgentSheets at the MacWorld 2000 Expo in San Francisco, CA, Jan. 5-8. Come visit us at booth #3558.

AGENTSHEETS THINKS DIFFERENT: Paper in Apple's Learning Technology Review

An AgentSheets article was recently published in the SimCorner of Apple's Learning Technology Review journal at

www.apple.com/education/LTReview/fall99/agentsheets/index.html

MOVIES:

New Tutorial Movies about AgentSheets

You can find our new instructional videos on how to use AgentSheets on the b3 CD or on the Web at

www.agentsheets.com/movies.html.

These include a getting-started movie to help you define looks and behaviors for your agents following the step-by-step description of the creation of the Virus Attack simulation.





Logo and High-level Geometric Thinking

by **DOUGLAS H. CLEMENTS AND JULIE SARAMA**

In a previous issue of *Logo Exchange* (Vol. 17, #4), de Los Santos and Patton discussed at length the learning of geometric skills with Logo. Nicola Yelland, along with colleagues, has spent years investigating how children learn mathematical ideas in Logo. She also frequently addresses gender issues. In this column, we describe one of her recent works (Yelland & Masters, 1997).

A Geometric Logo

*Turtle Math*¹ was designed as an environment to support the learning of geometry. We've described it in previous columns. For Yelland's study, the main advantage is its structure, which helps students maintain relationships between symbolic and visual external representations. But students may lose the connection between the two. *Turtle Math* encourages immediate mode programming—students type commands into the command center and see them immediately enacted. A tool (leftmost on the toolbar) transforms this set of commands into a procedure in the "Teach" window with one step. Further, any change to commands in either location, once accepted, are reflected automatically in the drawing. In this way, command window code constitutes what we call a proleptic procedure. The Logo code in the command window stands half-way between traditional immediate mode

records and procedures created in an editor, helping link the symbolic and visual representations.

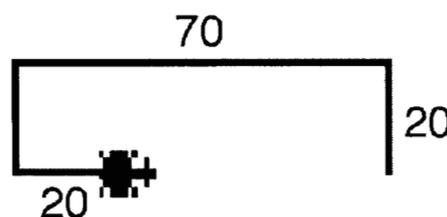
The *Turtle Math* tasks form a teaching and learning sequence. The curriculum unit used, from the *Investigations in Number, Data, and Space* curriculum, was called *Turtle Paths* (Clements et al., 1995)

Case Studies

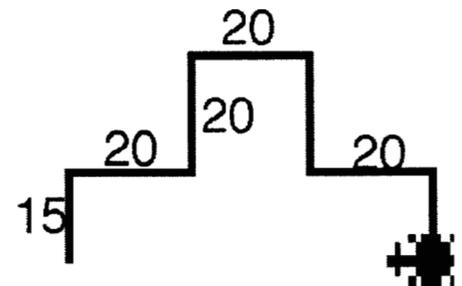
Yelland and Masters (1997) engaged seven pairs of 7- and 8-year-old Australian students in geometry activities using *Turtle Math*. They were interested in knowing how they would perform and what strategies they would use working in one of three gender pairs. Some pairs were girls, some boys, and some a girl and a boy.

Working on the Activities

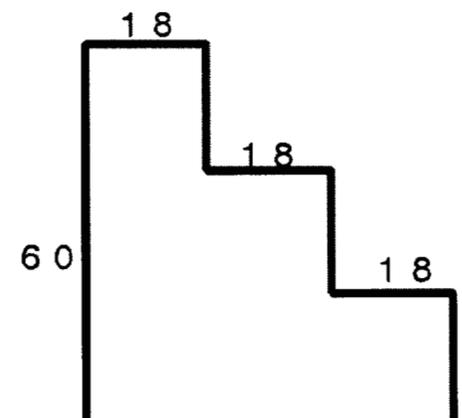
One set of activities was called "Missing Measures." Children were asked to provide the Logo code that would complete six geometric figures. Most of the beginning items were easy for them. They seemed to "just know" that for the final side of the following figure it had to be 50 more.



On the fourth item, "The Factory," illustrated below, all the students initially believed the missing distance would be 40. When they entered that amount, they were surprised that it did not work, but soon recognized that the gap was 60 because of "the third step."

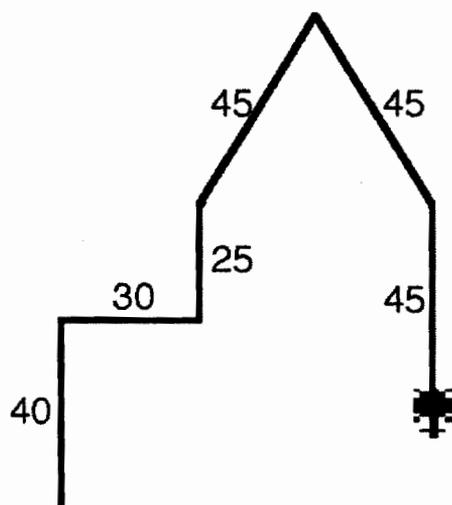


The fifth item, "Steps," was much more challenging. They all thought that the vertical measure of the step would be 18. Most pairs of students thought that the final step would be 20, another thought it would be 25. All justified it because it looks "a bit bigger" than the other steps.



All pairs also thought the base would be 60 because the first side was 60. This was true even though they had successfully matched opposite sides in previous problems. When they tried out their solutions, they adjusted them visually so that no line was “sticking out.” For example, they changed the final side to 56, 55, and finally 54.

When the researcher told them the vertical step should be 20 because the total height was 60, they merely said that was another way to do it! The researcher prompted them to think about the base as the same as the three horizontal steps and they say “Oh, yes!” and worked it out by adding (not multiplying) to arrive at 54.



The final item was called “House.”

On the final side, most pairs entered 45 and then added on 20. They reasoned that it had to be 65 because it was the same as “40 and 25 over there.” One pair just entered 65. So, it seems that when visual and numeric information could be combined easily, the students adopted this strategy. However, when numbers (such as three 18s) could not be so easily calculated, they reverted to a visual approximation strategy. These numbers were not multiples of 5 or 10, and there were three of them.

The base of the house was more challenging. This was probably because one of the distances to be combined had to be deduced. One pair of students successfully reasoned this

out, the other pairs combined guessing and reasoning. They used trial and error. Here the *Turtle Math* environment was important, because it allows students to change the code and immediately see the change in the graphics without starting from the beginning. “This design feature is one of the most empowering features” (p. 91); it encourages meaningful mathematical exploration.

In summary, the Missing Measures activity encouraged students to think about relationships of geometric figures and connect these properties to numbers and computation. The opportunity to share their strategies with each other was particularly important to their learning. They would appear to benefit from even more work such as this; for example, they “knew” multiplication, but were not yet applying it in situations where it would have been useful.

There were many other activities, but this provides the flavor of the students’ work. We now turn to the pre- and posttest results.

Pre-Post Test

One item on the test asked children what the lengths of the sides of a rectangle, square, and equilateral triangle would be if the perimeter of each were 12. At the pretest, only three of the 14 students could complete these calculations. At the posttest, all the children correctly answered all parts of these item.

More students also correctly completed different missing-measures problems. The most interesting result, however, was that many more provided mathematical justifications.

Conclusions

These students showed a high level of engagement and learning in the *Turtle Math* environment. They engaged in complex problem solving that required them to integrate previously acquired mathematical knowledge with strategies in a new and dynamic way. The graphic manifestation of their thinking allowed them to reflect on that

thinking and modify it. The students were able to make sense of the many mathematical ideas by literally playing with them on the screen. The use of technology enabled the students to build and test ideas for themselves. Finally, the mathematics was motivating and meaningful for the students because it involved Logo, nontrivial problems, and using mathematics in an active manner.

Gender and Logo

Yelland and Masters, in other articles, discuss how the various gender pairs acted differently. Indeed, Yelland and other researchers have conducted numerous studies that investigate gender roles in using and learning with Logo. We’ll be reviewing this body of research when you hear from us next. LX

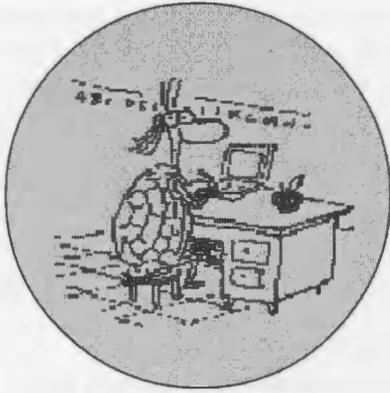
References

- Clements, D. H., Battista, M. T., Akers, J., Woolley, V., Meredith, J. S., & McMillen, S. (1995). *Turtle paths*. Cambridge, MA: Dale Seymour Publications.
- Yelland, N. J., & Masters, J. E. (1997). Learning mathematics with technology: Young children’s understanding of paths and. *Mathematics Education Research Journal*, 9, 83-99.

This paper was supported in part by the National Science Foundation under Grant No. ESI-9730804, “Building Blocks-Foundations for Mathematical Thinking, Pre-Kindergarten to Grade 2: Research-based Materials Development.” Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

¹*Turtle Math* is the name of this version of Logo as a stand-alone product (available from the authors of this column); it’s also known as *Geo-Logo* when sold with fewer activities as part of an Investigations in *Number, Data, and Space* unit.

See GEOMETRIC (Page 32)



Use Multiple Pages and Animation Together

or building games in MicroWorlds without tears

by **JEFF RICHARDSON**, president of ISTE's SIGLogo

Many *Logo Exchange* readers will be familiar with the capacity of MicroWorlds to enable building simulations and games. Some terrific examples have been made by children at The Spence School in New York City. Hope Chafian and Michael Tempel have documented some of this work at:

<http://el.www.media.mit.edu/logo-foundation/VGW/>

<http://el.www.media.mit.edu/logo-foundation//LU/v7n2/games.html>

Their workshop was one of the highlights of Logosium last June in Philadelphia.

Children (and adults) working on projects like this often hit a particular type of bug. In a multiple-paged project, a nice feature is to have a page change invoked by the result of some process . . . an animation whereby a turtles crosses into a defined area, or a specific number or text is entered from the keyboard.

So far so good, but the bug arises when animations or processes continue to run, invisibly in the background after the page change has taken effect. Here's a simple example:

A Turtle, in the shape of a Fish, is swimming along on a page called SEA. Part of the graphic background of this page is a cave mouth, with the color of

the cavemouth programmed to open another page, CAVE, when the Fish passes across it.

The Fish is carrying this Logo code (instruction):
`FD 3 RT RANDOM 3 FD 3 LT RANDOM 3`
Many Times.

The cave mouth is colored *brown*, and *brown* on the palette for the SEA page is programmed for:

Double click on the color *brown* in the drawing (graphics) center and set

the instructions as follows:
 Turtle: **CAVE**
Once.

Now the Fish, once activated with a mouseclick, will swim along until it eventually 'enters' the Brown cave mouth, and when it does the SEA page will be closed and the CAVE page will open. This'll work well enough, and you can create the illusion of the Fish going on to further adventures in the cave with another Fish-shaped turtle on this page.



SIGLogo President Jeff Richardson chats with Logo pioneer Cynthia Solomon

Meanwhile, the original Fish continues to swim blithely along on the SEA Page, even though you can't see it anymore, giving rise to all sorts of unanticipated, and unwanted effects.

Many people instinctively try using the MicroWorlds Primitive STOPALL to deal with this, discovering in the process that it is a very blunt instrument. What's needed in this situation is something that will stop the Fish, but let the rest of the project keep running. A quick fix is to add CLICKOFF to the programming for the color *brown* on the SEA page so that the Fish stops, then

the Page turns. Even better is to use the opportunity to tidy up the SEA Page, and restore it to state that you want it to have when it is next opened.

So, instead of running CLICKOFF CAVE, the color *brown* could be programmed for the Turtle with this procedure:

```
to change
clickoff
ht
setx -270
cave
end
```

This will stop then hide the Fish, move it back to the left of the Page, and then close the Page and open the Cave page.

There's at least one bug left with this. When the SEA page is reopened, the Fish will be sitting waiting to be clicked on to swim off, but it will be invisible. . . .

I'll leave it to you to devise your own solution for this one. 

About the Author

Jeff Richardson is International Editor of *Logo Exchange* and President of SIGLogo. Jeff may be reached at jeff@rmit.edu.au.

Quick Classroom Ideas:

LEGO® Design Brief

Build a Temperature Sensitive Vending Machine

by Gary Stager

Description

In October 1999, the Coca-Cola Company announced plans for the testing of a new vending machine that would charge more for a can/bottle of soda when the temperature outside rises above a certain number of degrees. Logic would suggest a lower price would be charged in cooler weather.

Challenge

Build a LEGO vending machine with Control Lab or the RCX (programmable brick) that accepts coins as payment before it dispenses a product. (LEGO bricks can be used to represent soda cans.) Increase your profits by charging customers more during weather conditions likely to increase thirst.

The machine should:

- Dispense a product
- Count the number (and perhaps denomination) of coins being deposited
- Require more coins when the temperature exceeds a specific number of degrees and reduces the cost when the temperature dips below a specific number of degrees. You probably need a way of telling the customer how many coins to deposit.

Add Next and Previous Navigation Buttons to Your MicroWorlds Pages

by Michael Tempel, President, Logo Foundation

Here's a trick that allows you to put a **next** and a **previous** button on each page of your MicroWorlds project. You'll need these two procedures on your procedures page:

```
to next
getpage word `page sum 1 bf bf bf bf first
pagelist
end
```

```
to previous
getpage word `page difference bf bf bf bf
first pagelist 1
end
```

Then you will need to create two buttons on each page, one with the instruction **next** and the other with the instruction **previous**.

This assumes that your pages are named **page1**, **page2**, **page3**, ..., **page10**, ... etc., which is what they will be named automatically when you create them using **newpage** command or **New Page** on the Pages menu. Don't put a **next** button on the last page or a **previous** button on page 1, or you'll cause an error.

Here's how it works. **Pagelist** reports the list of the names of the pages in the project with the current page being first in the list. The four **bfs** (**butfirst**) remove the word "page" from the page name; **sum** or **difference** does the arithmetic on the number that's left, adding or subtracting 1. **Word** sticks the word "page" and the new number together and hands the result to **getpage**, which gets the appropriate page. 



What's New in MicroWorlds Pro

by **GARY S. STAGER**

LCSI recently released an exciting new multimedia, simulation and interactive Web-authoring package, MicroWorlds Pro.

MicroWorlds Pro (Win 95/98) is the first major revision of the award-winning MicroWorlds software since 1993. While it is a completely new software package, all of your existing MicroWorlds projects will run in MicroWorlds Pro.

While MicroWorlds Pro was developed to support the learning of secondary school students through the addition of new programming features and more advanced project ideas, it is an excellent tool for learners of all ages. A principle of Logo has always been that it has no threshold and no ceiling. MicroWorlds Pro is no exception.

The following are a few reasons why MicroWorlds Pro may be the Logo you've been waiting for.

New and Improved Interface

All of the multimedia object tools available in MicroWorlds are available in MicroWorlds Pro. MicroWorlds Pro has sports a greatly improved WYSIWYG interface in which beautiful projects may be created and debugging is a breeze.

There are four tabs in MicroWorlds Pro: procedures, processes, graphics and objects. These tabs sit alongside your project, unless you hide them.

Procedures reside in a tab (window) right alongside of your page. This way you can affect cause and effect without flipping the page.

The processes tab lists all of the processes currently running so you can keep track of what's happening "behind the scenes." The ability to choose a normal, slow, and slower speed at which your project runs makes debugging more clear.

The objects tab shows a list of every page in your project and every object on that page. Right-click on one of those objects and you can edit it.

The graphics tab contains your turtle costumes and painting tools in one area. Shapes and graphic tools are no longer kept in separate areas.

Right-click on any object or even the page background and you can edit that object!

Better Graphics

MicroWorlds Pro has room for a lot more turtle costumes and all sorts of new paint brushes. The shapes editor allows you to rotate a costume by any number of degrees. You can even magnify the drawing area in the shapes editor.

Perhaps the coolest thing about the new graphics tab is that if you select a shape and click on a turtle, that turtle will wear that costume. If you click on a shape and then click on the page, that graphic will be automatically stamped on the page. You can resize the image before it becomes a permanent part of the page background.

MicroWorlds Pro makes it easy to import shapes from other projects. Choose Import-Shapes from the File menu and a graphical interface pops up allowing you to select a shape from

one file and choose where it will be stored in the current shapes list.

In MicroWorlds you can now rotate a shape by a specific number of degrees. This makes it easier to create turtle costumes that will turn under program control.

Little kids and impatient "hard" programmers can animate a turtle by shift-clicking through a variety of turtle costumes. Right click-Animate to make the turtle go.

Instant Web Publishing

With the click of the mouse MicroWorlds Pro automatically exports your project for play via the Web plug-in AND creates the HTML file necessary for playing that project on the Internet or Intranet. This HTML file may be opened in a Web-authoring program if additions to the page are desired.

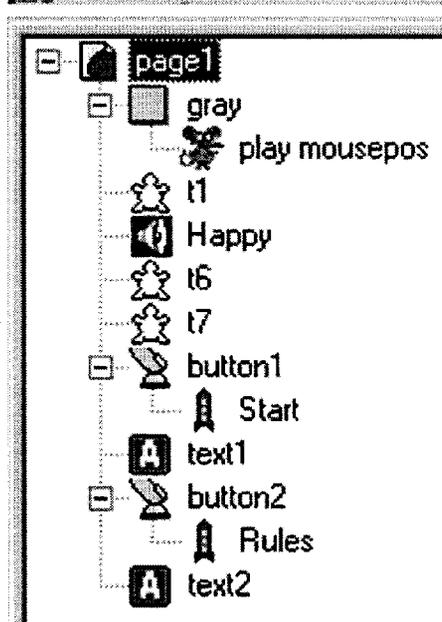
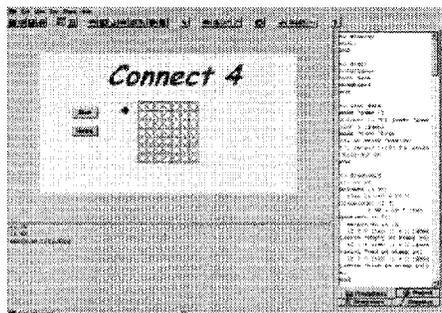
No other program offers P-12 students such powerful opportunities for publishing interactive work on the Web! Web-based projects no longer need to be static.

MicroWorlds Pro Talks with Excel

MicroWorlds Pro can write data to an Excel spreadsheet for further manipulation and graphing. It can also read data from an Excel spreadsheet opening up all sorts of simulation possibilities based on real data. Future issues of *Logo Exchange* will explore these exciting capabilities.

MicroWorlds Pro Can Spell

Right click on a text box and Micro-



Worlds Pro uses the Microsoft Word dictionary to check your spelling.

New Sophisticated Programming Functionality

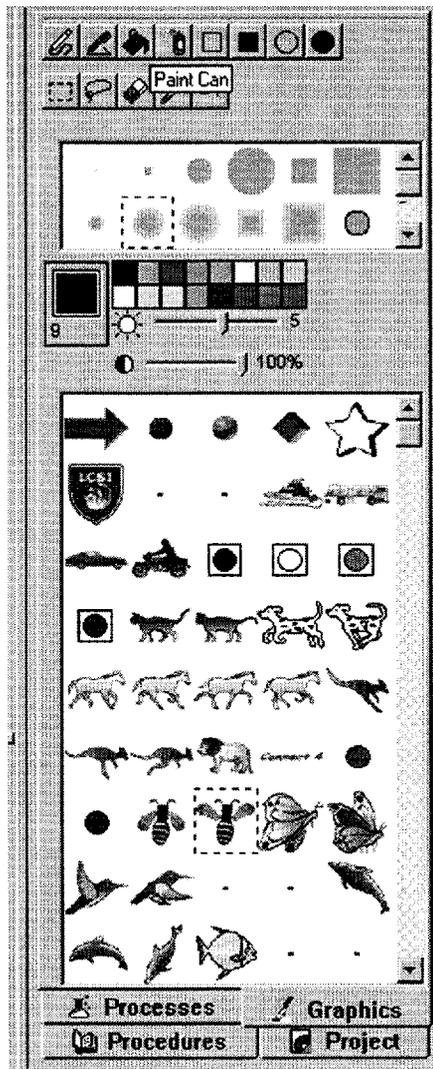
A variety of new data structures and primitives are part of MicroWorlds Pro. Many more processes may run in parallel as well.

Your Pages Scroll

If you can justify doing so, MicroWorlds Pro allows you to create pages larger than your viewing area. Since your pages scroll, don't be surprised when your turtles disappear temporarily. MicroWorlds Pro offers a variety of preset project sizes from the File-New Project Size menu. Keep your projects small if you plan on Web publishing.

Online Interactive Help

The MicroWorlds Help system is built in HTML and features interactive ex-



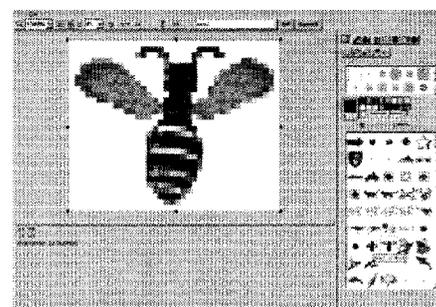
amples of how primitives and programming concepts might be used.

New Documentation

MicroWorlds Pro comes with two terrific new books, *Learning with MicroWorlds Pro* by Logo Exchange founder Tom Lough, and *MicroWorlds Pro Tips and Tricks*. The *Tips and Tricks* book explains a few dozen MicroWorlds skills and concepts through short clear examples. Tom Lough's tutorial contains a variety of project ideas geared towards secondary and post-secondary users. The projects include presentations, games and genetic simulations.

It's Great for Little Kids

Don't let the documentation and marketing scare you. The improved more



concrete user-interface and graphic tools makes MicroWorlds Pro an outstanding tool for even the youngest children. For example, you can hatch a turtle, put a shape on the turtle, and, from the contextual menu, choose animate. Voila! Your turtle is now animated!

I can hear what some of you are thinking. Where's the thinking? The answer is two-fold.

1. If you just need turtles to move and don't care about their direction or velocity, then this allows you to move on to higher-level challenges.
2. Everything in MicroWorlds Pro is transparent. Edit that turtle and see the `fd 5 wait 1` instruction for that turtle. Feel free to modify the instruction if you wish.

Shift-click a series of shapes and then the turtle and that turtle will use flipbook animation.

All of your old Logo activities will work just fine, probably even enhanced, in MicroWorlds Pro.

Give MicroWorlds Pro a try. Share your experiences and innovations with us and we'll feature them in future issues of *Logo Exchange!*



MicroWorlds Tutorial: Functions and Graphing

by **GARY STAGER**

This set of activities introduces a number of mathematical operations to middle/high school students and then challenges them to write programs for creating XY tables and coordinate graphs. Reproduce these pages and use them with students.

Things to try

What is the difference between the following reporters?

INT
ROUND
ABS
MINUS

What do each of the reporters above do? Substitute numbers after each of the reporters and see if you can explain what each does. For example,

Show int 3.5
Show round 5.6
Show abs -4.2
Show minus -7.6

Important information

You can find the negative form of a number in the following two ways:

Σ Multiply a number by -1
Σ Use the minus reporter before the value

:x is an input to the following procedures and holds a value

```
to negative :x
show :x * -1
end
```

```
to negative2 :x
show minus :x
end
```

Create a data (X|Y) table

Σ Create a text box to contain your data
The following procedure will plug-in dif-

ferent values for :x with a fixed function and print both the :x and :y values in the text box.

This procedure is creating a table for the function, $y = \text{int}(x)$

```
to table :x
print (sentence :x int :x)
table :x + .01
end
```

The .01 is called the increment. You can change this value to increase/decrease each :x value. Subtracting from the value for :x would be called a decrement. Test this procedure by typing something like

Table -5 in the command center

You may wish change your table procedure to contain the maximum number you wish to test as an :x value in the following way. **DO NOT CREATE A NEW PROCEDURE**, merely change the existing table procedure. The increment increases the value for :x, but the maximum (limit) number remains the same (constant).

```
to table :x :maximum
if :x > :maximum [stop]
print (sentence :x int :x)
table :x + .01 :maximum
end
```

table -10 10

Something for you to try

Can you write new procedures like table to calculate the following functions?

$y = -x$ $y = -\text{int}(-x)$

Making a better tool procedure

Type run [show 50 * 2]

What do you think that the run command does?

```
to bettertable :x :function :maximum
if :x > :maximum [stop]
print (sentence :x run :function)
```

```
bettertable :x + .1 :function :maximum
end
```

Run the procedure by trying something like:

```
bettertable -10 [:x * :x] 10
```

Can you add an input for the increment?

A not-too-smart graphing program

The same idea used in bettertable may be used to graph a function.

```
to graph :x :function :maximum
if :x > :maximum [stop]
setpos (list :x run :function)
dot
graph :x + .1 :function :maximum
end
```

```
to dot
pd fd 0
end
```

setpos is a command which takes a list as input, i.e. .. setpos [-45 32] and puts the active turtle there. List is a reporter whose job it is to force to (or more) things together into a Logo list and report them.

The dot procedure may be used to draw an x or larger dot on a point by stamping a shape you design.

Try graph by attempting something like the following:

Graph -10 [int :x] 10

Did it work?

Challenge

Can you create a graphing procedure that does the following?

Σ Graphs a function
Σ Begins at the furthest possible point on the screen
Σ Automatically stops when the x value gets too high





Who Wants to Be a Millionaire MicroWorlds Pro Style?

by ANONYMOUS and GARY STAGER

Editor's Note: The terrific MicroWorlds project and the description of its code were written by a 15 year-old student. For purposes of confidentiality that student's name may not be used in this publication. I will explain the context for the project and a bit about how it works. The student wrote the program itself and the explanation of that program.

The game show Who Wants to be a Millionaire is a phenomenon sweeping several continents. A group of students I work with decided to produce their own videotaped version of the popular game show in December 1999. At my suggestion the game would involve questions regarding 20th century history to coincide with the end of the century.

The television show uses computer graphics to display the questions posed to contestants. The students decided to choose questions on a computer and project the questions and four possible answers on a screen to be filmed by one of two digital video cameras. The other camera would be trained on the host and contestant. For this reason, the program detailed in this article displays questions and possible answers. A human operator makes a decision about the accuracy of the question and asks the computer to flash the correct answer. One command (flash) is typed when the contestant answers correctly. In this case the contestant's winnings double. If the lose command is typed, the correct answer is flashed and the contestant loses some or all of his/her money. If you don't understand the rules of the game, ask a colleague or student.

It would not be difficult to modify this program to make "the home version" of the game to be played by a user against the computer. A few extra buttons turns this display program into a trivia game.

Enter MicroWorlds Pro

A good game show requires a large collection of questions and this game also requires four multiple-choice answers per question. Traditional Logo data structures like lists, reporters and global variables make entering, storing and manipulating large collections of data cumbersome. MicroWorlds Pro provides simple primitives for reading and writing to Excel spreadsheets. Therefore my student decided to use a spreadsheet as the container for the questions and possible answers.

Anatomy of the Data

This project uses Excel much like a database. Rows represent records and columns contain questions, answers, the correct answer and a field for keeping track of used questions. You can of course use the following format to use your own questions and answers regarding any theme you wish.

Enter the following data in the following columns:

- Column A - Question
- Column B - Answer A

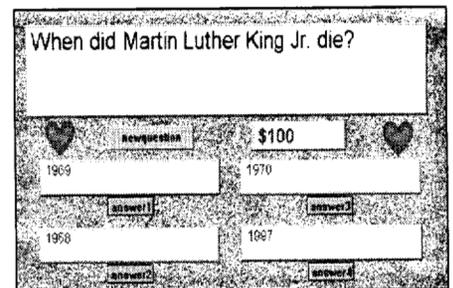
- Column C - Answer B
- Column D - Answer C
- Column E - Answer D
- Column F - The correct answer
- Column G - The checksum value (written by MicroWorlds, not entered by the designer)

Enter your data in the format described and be sure to save often.

The MicroWorlds Pro Program

You will need five text boxes on a MicroWorlds Pro page. They are Challenge (since question is a primitive), Answer1, Answer2, Answer3, Answer4. You may also wish to add small buttons for Flash and Lose. These buttons are pressed based on whether the contestant answers the question correctly or incorrectly. Decorate the rest of the page any way you like and be sure to save the project.

The following are the procedures used in this game show. Enter them in the MicroWorlds Pro procedures tab.



```

to startup          The procedure's name
carefully [openworksheet "gamedata "sheet1] []
;tells MicroWorlds Pro to open the file in Excel called "gamedata" and not to do anything with it, to just
leave it open. If it is already open, it does nothing else.
make "magic random 9999 ;makes the computer choose any number from 0 to 9999 and call it the "magic"
number. This number is entered in the seventh column when a question has been used. This allows Logo to check
to see if a question has already been used.
clear          ;clear all the textboxes
end           ;end of the procedure's directions

```

```

to newquestion
make "choice 1 + random 107 ;picks out a random question from the 107 of them in the Excel spread-
sheet - the number used is the number of questions in your spreadsheet
ifelse :magic = getcell :choice 7 [newquestion stop]
[setcell :choice 7 :magic] ;if the magic 'checksum' number appears in the 7th column of a row/question,
then a new question is chosen. Otherwise, the magic number is put in column 7/g
setchallenge getcell :choice 1 ;puts one of the questions in column 1 in the spreadsheet into the
textbox named "challenge"
setanswer1 getcell :choice 2 ;puts the entry in the 2nd column of the same row into the textbox named "answer1"
setanswer2 getcell :choice 3 ;puts the entry in the 3rd column of the same row into the textbox named "answer2"
setanswer3 getcell :choice 4 ;puts the entry in the 4th column of the same row into the textbox named "answer3"
setanswer4 getcell :choice 5 ;puts the entry in the 5th column of the same row into the textbox named "answer4"
end

```

```

to clear
ask [challenge answer1 answer2 answer3 answer4]
[ct] ;talks to the textboxes named "challenge", "answer1", "answer2", "answer3", and "answer4"; clears all
the text in the mentioned textboxes
setmoney "$100 ;puts the value of $100 into the textbox named "money"
make "questionnumber 1 ;tells it to set the number of questions asked to 1
end

```

```

to flash
if answer1 = getcell :choice 6 [blink "answer1];if the entry in the textbox "answer1" is the same
as the entry in column 6 in the spreadsheet, to make it blink
if answer2 = getcell :choice 6 [blink "answer2];if the entry in the textbox "answer2" is the same
as the entry in column 6 in the spreadsheet, to make it blink
if answer3 = getcell :choice 6 [blink "answer3];if the entry in the textbox "answer3" is the same
as the entry in column 6 in the spreadsheet, to make it blink
if answer4 = getcell :choice 6 [blink "answer4];if the entry in the textbox "answer4" is the same
as the entry in column 6 in the spreadsheet, to make it blink
make "questionnumber :questionnumber + 1 ;increase the counter keeping track of the question number
setmoney word "$ item :questionnumber prizelist ;makes it so that when the program picks a new question to
display, the amount in the textbox, "money" goes to the next amount listed in the "prizelist" procedure and
puts a "$" in front of it
end

```

```

to blink :textbox ;names the procedure and is referring to a textbox input by the user
ask :textbox[
repeat 5 [showtext wait 2 hidetext wait 2]
showtext] ;talks to one of the textboxes and show the text for 2 units of time and then hides it for 2 units
of time, repeats that process 5 times, and ends by showing the text again
end

```

```

to prizelist
output [100 200 300 500 1000 2000 4000 8000 16000 32000 64000 128000 250000 500000 1000000];outputs all the amounts
of money used in order of appearance
end

```

```

to lose
flash ;tells the correct answer to flash
if :questionnumber > 10 [setmoney "$32000 stop] ;if the question's amount is less than the 10th amount of money listed (32000), to set the
amount in the textbox, "money" to $32000 and stop
if :questionnumber > 5 [setmoney "$1000 stop] ;if the question's amount is less than the 5th amount of money listed (1000), to set the
amount in the textbox, "money" to $1000 and stop
SETMONEY "$0
end

```

The challenge

Can you turn this program into a game playable by others? What sorts of other similar games can you design? Can you think of ideas for using other sorts of

data collections to fuel MicroWorlds Pro simulations, games or graphical representations?

Please share your creations with *Logo Exchange* for future publication. 

About the Authors

Douglas H. Clements, Professor at the State University of New York at Buffalo, has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. Through a National Science Foundation (NSF) grant, he developed a K-6 elementary geometry curriculum, *Logo Geometry* (published by Silver Burdett, & Ginn, 1991). With colleagues, he is working on the previously mentioned NSF research grant and is finishing a second NSF-funded project, *Investigations in Number, Data, and Space*, to develop a full K-5 mathematics curriculum featuring Logo. With Sarama, he is co-authoring new versions Logo for learning elementary mathematics. One, *Turtle Math*, is currently available from LCSi. Sarama and Clements are co-PI's on the aforementioned *Building Blocks* project, which is developing mathematics software for preschool to grade 2 children.

Julie Sarama, Ph.D., is an assistant professor at Wayne State University, where she teaches mathematics content courses for pre-service teachers and research courses for graduate mathematics education students. She has studied teachers' use of computer innovations and students development of mathematical constructs while working in computer microworlds. She is co-author of several *Investigations* units and of *Turtle Math* and has designed and programming new versions of Logo and other computer microworlds. She is co-principal investigator on the new *Building Blocks* project.

Douglas H. Clements
SUNY at Buffalo
Dept. of Learning and Instruction
593 Baldy Hall
Buffalo, NY 14260
Clements@acsu.buffalo.edu

Julie Sarama
Wayne State University
Teacher Education Division
Detroit, MI 48202
Sarama@coe.wayne.edu

Communicating with Excel

MicroWorlds Pro allows users to write (store) and read (retrieve) data in an Excel spreadsheet. This creates opportunities to use the abundant collections of tab-delimited data available online and store large quantities of data created by MicroWorlds in an organized shareable format. For example, a Logo probability experiment could store large quantities of numerical data in Excel and geographic or meteorological databases could be used to create visual simulations.

Note: You must have Excel installed on your computer for this to work.

MicroWorlds and Excel work together with just three primitives—two commands and one reporter.

Openworksheet is a command that takes two inputs—the name of an Excel file and the specific worksheet you wish to use. You must already have a spreadsheet file saved even if it is empty.

```
openworksheet "probability
sheet2
```

setcell takes three inputs—row, column and the value or formula you would like to put in a specific cell.

```
setcell 3 5 75
setcell 4 6 [= a2 + c3]
```

getcell is a reporter that takes two inputs—row and column—and reports the quantity found in that cell.

show getcell 2 3 will display the contents of cell c2

A few tools

The game show procedures use the row and column format of MicroWorlds since it's easy to choose a row randomly and then look at the data in each column. Other types of projects might need a different way of dealing with data.

I know what you are thinking. "You don't refer to spreadsheet cells this way. Spreadsheets use cell labels like, c3 or f9."

Well, here are a few procedures for communicating with Excel data in more conventional ways. Putcell uses a cell reference and value as input and sets that cell to the value. Cell reports the contents on an Excel spreadsheet cell.

```
to putcell :cell :value
setcell bf :cell convert :cell :value
end
```

```
to convert :thing
ifelse (ascii first :thing) > 96
[output (ascii first :thing) - 96]
[output (ascii first :thing) - 64]
end
```

```
to cell :cell
getcell bf :cell convert :cell
end
```

Try typing the following in the command center of MicroWorlds Pro.

```
putcell "c3 75
putcell "d6 [= c3/b5]
```


ISTE BRINGS THE WORLD OF TECHNOLOGY CLOSER TO YOU.

By drawing from the resources of committed professionals worldwide, ISTE provides support that helps educators like you prepare for the future of education.

As an ISTE member, you benefit from a wide variety of publications, national policy leadership, and our work with Teacher Accreditation.

You also enjoy exciting conferences, global peer networking, and graduate-level Distance Education courses.

So if you're interested in the education of tomorrow, call us today.



iste International Society for
Technology in Education

Teachers Helping Teachers Use Technology in the Classroom

WE'LL PUT YOU IN TOUCH WITH THE WORLD.



iste

International Society for Technology in Education

Administrative Office

1787 Agate Street, Eugene, OR 97403-1923 USA

Non-Profit
Organization
US Postage
PAID
ISTE