SPECIAL GAMES ISSUE!

FDC 50076

ST LOG

THE ATARI ST MONTHLY MAGAZINE

OCTOBER 1989
ISSUE 36

U.S.A. $3.95
CANADA $4.95

U-F-OTO

CAPITAL ST

USE THE FORSLOOK

REVIEWS:
Hyperfont
Laser C 2.0
Baal
War in Middle Earth
Tetris

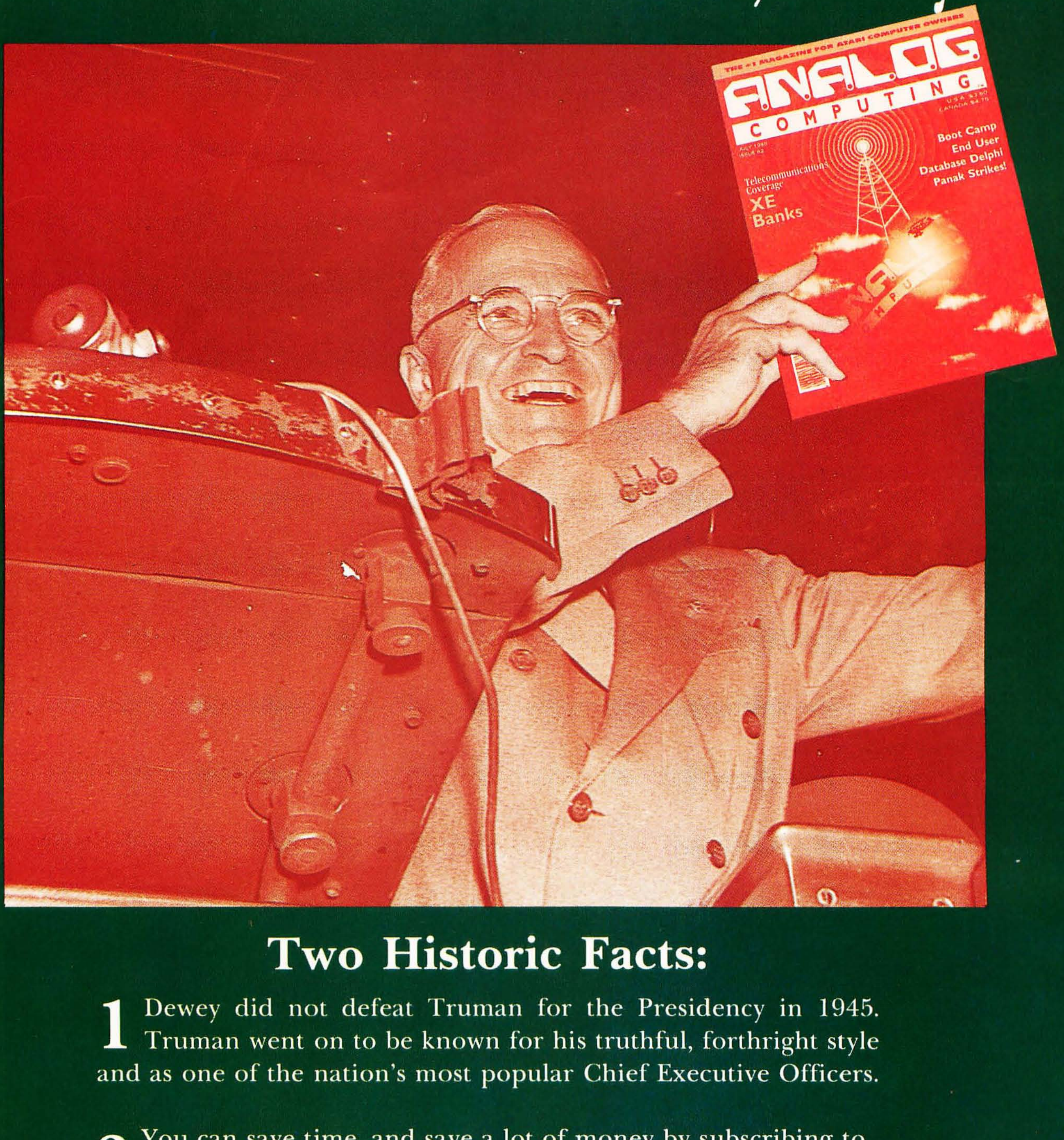
THE ANIMATION STAND

PROGRAMMER'S CALCULATOR

10

74369 50076 1

Give 'Em A.N.A.L.O.G., Harry!



Two Historic Facts:

1 Dewey did not defeat Truman for the Presidency in 1945. Truman went on to be known for his truthful, forthright style and as one of the nation's most popular Chief Executive Officers.

2 You can save time, and save a lot of money by subscribing to A.N.A.L.O.G. Computing Magazine. Save $19 off the cover price with the convenience of having A.N.A.L.O.G. delivered directly to your door before it even hits the newsstands. To order use the handy postage-paid order card located in the back of this magazine!

1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

EDITORIAL

BY CLAYTON WALNUM

As most of you know, this is the last issue of this magazine in its current form. As of next month, ST-LOG will be merged with ANALOG Computing to form a comprehensive Atari-specific publication. If you've read the publisher's letter in the previous issue, you know our reason for the merge: The U.S. Atari market is not large enough to support two Atari-specific magazines from the same publisher. Specifically, advertising, which provides an important portion of every magazine's earnings, is an at all-time low.

The publisher's letter also stated that this month we would give you more details about the new magazine. That task has fallen to me (lucky guy).

The new ANALOG Computing will be much larger than the magazine you're now holding in your hands. It will contain 132 pages, 48 of which will be in full color. A magazine of this size will give us plenty of space to cover the Atari market in full, while still providing the types of features and columns you've come to expect.

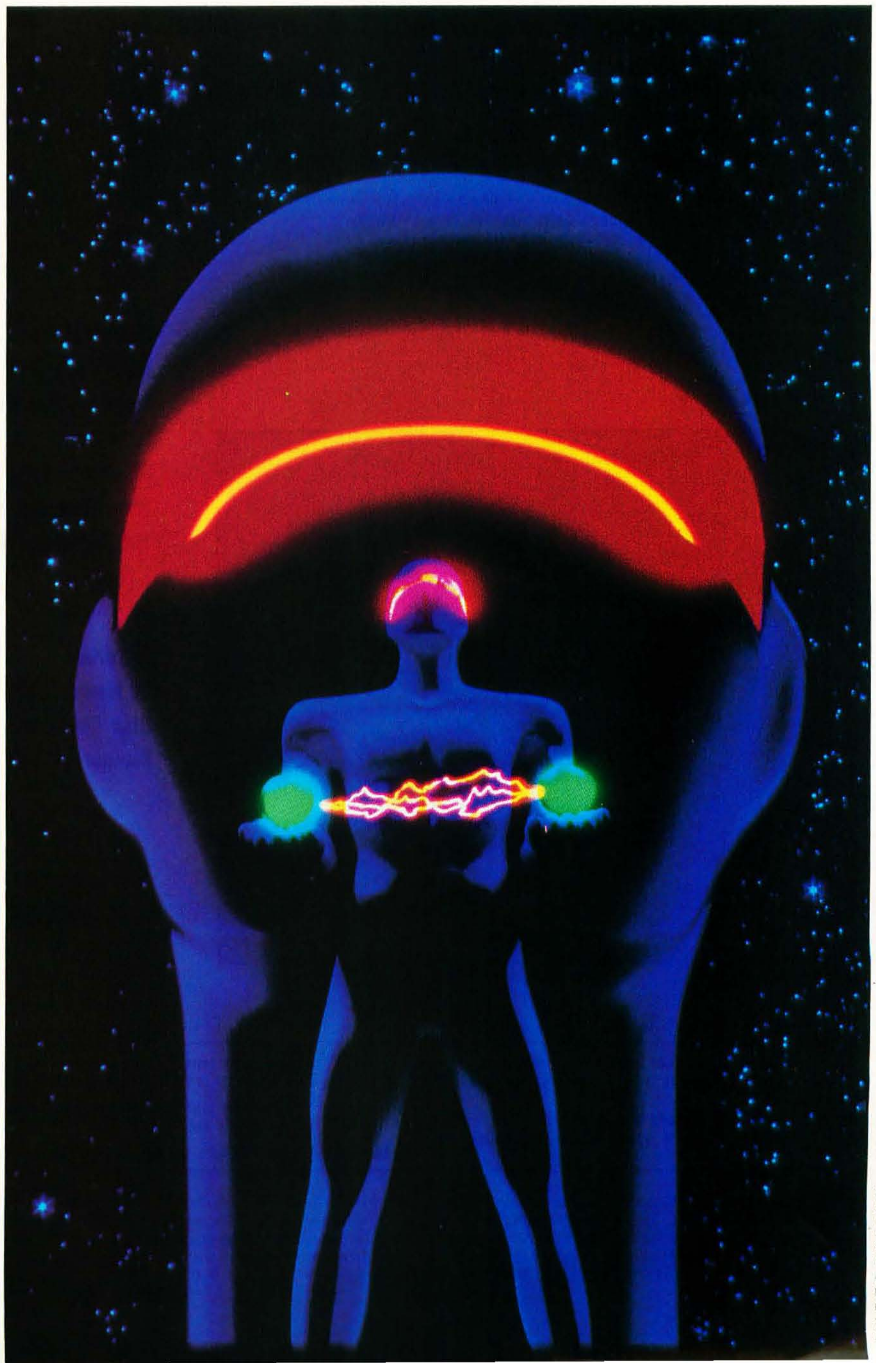
Although we'll still be offering monthly disks, both 8-bit and ST, we've decided not to provide the disk version on the newsstands. We feel that having two versions of the same magazine will be confusing to both buyers and retailers. If you're interested in obtaining the disk each month, we urge you to subscribe. Those who don't wish to subscribe will be able to order the disks by mail. We will be offering a service that will get disks out to you immediately upon the receipt of your order. In addition, we hope to be able to lower the disk price.

Little otherwise is going to change. Essentially, the merging of the magazines will give you more for your money. We will be providing complete Atari coverage in a much larger format for the same price.

As usual, we would like to hear from you. Your input is important to us. If you have any ideas, let us know. If there's something we can do to make the new magazine better suit your needs, drop us a line. We'll give serious consideration to all your comments, and even share some of them in "Reader Comment."

As always, we at ANALOG Computing are looking forward to serving you, our readership, for many years to come. ■

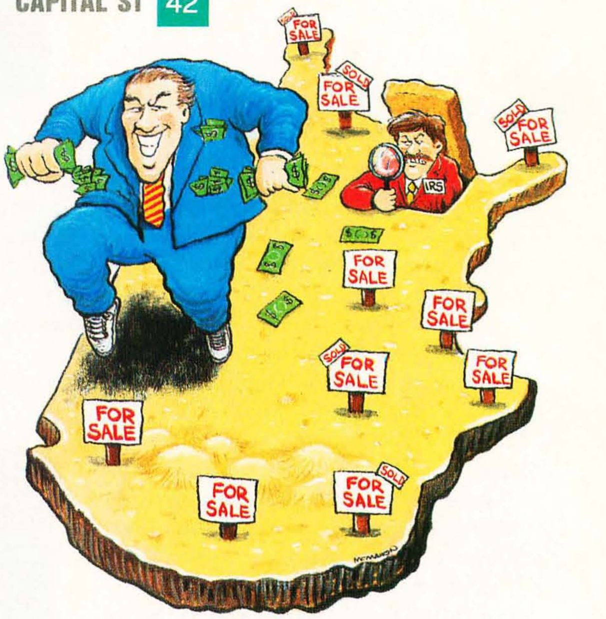
Please send all correspondence regarding this editorial to: ANALOG Computing, *P.O. Box 1413-M.O., Manchester, CT 06040.*



PHOTOGRAPHY: JOSEPH DRIVAS

IN THIS ISSUE

CAPITAL ST 42



U-F-FOTO 10

SYMBOL · GUIDE

C/M This program runs in both color and mono. It is available in type-in form & on the disk.	M This program runs in mono only. It is available in type-in form & on the disk.	C This program runs in color only. It is available only on the disk.
C This program runs in color only. It is available in type-in form & on the disk.	C/M This program runs in both color and mono. It is available only on the disk.	M This program runs in mono only. It is available only on the disk.
	(No image) This article does not have an associated program.	

ST-LOG (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1989 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission of the publisher. U.S. subscription $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscriptions: add $10 per subscription per year. Single copy $3.50 (add $1 for postage). Change of address: six weeks advance notice, and both old and new addresses are needed. POSTMASTER, send change of address to: **ST-LOG** Magazine, P.O. Box 16928, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

OCTOBER 1989
ISSUE 36

ST LOG

STAFF

Publisher: Lee H. Pappas. **Executive Editor:** Clayton Walnum. **Associate Editor:** Andy Eddy. **Art Director:** Andy Dean. **Managing Editor:** Dean Brierly. **East Coast Editor:** Arthur Leyenberger. **West Coast Editor:** Charles F. Johnson. **Contributing Editors:** D.A. Brumleve, Ian Chadwick, Frank Cohen, Maurice Molyneaux, Steve Panak, Marshal M. Rosenthal, George L. Smyth. **Copy Chief:** Sarah Weinberg. **Copy Editor:** Alyson Gould. **Chief Typographer:** David Buchanan. **Typographers:** B. Miro Jr., Quita Saxon, Ligaya Rafael, Larry Gannon. **Editorial Assistants:** Patricia Koury, Catrina Mason, Robin Thompson. **Contributors:** Albert Baggetta, Robert Birmingham, William Busby, James M. Cirile, Frank Eva, Bill Griggs, Pamela Rice Hahn, Darek Mihocka, Kathy Schaefer, M.D., Ron Schaefer, M.D., Todd Threadgill, Donavan Vicha, Scott Wasser, Carl Wiegers. **Vice President, Production:** Donna Hahner. **Production Assistant:** Steve Hopkins. **National Advertising Director:** Jay Eisenberg. **Corporate Director of Advertising:** Paula S. Thornton. **Advertising Production Director:** Janice Rosenblum. **Advertising Production Coordinator:** Maggie Chun. **Corporate Editorial:** Tim Conaway. **Corporate Art:** Pam Carr. **Subscriptions Director:** Irene Gradstein. **Vice President, Sales:** James Gustafson.

U.S. newsstand distribution by Eastern News Distributors, Inc., 1130 Cleveland Rd., Sandusky, OH 44870.

ST-LOG Magazine (L.F.P., Inc.) is in no way affiliated with Atari. Atari is a trademark of Atari Corp.

ADVERTISING SALES

Correspondence, letters and press releases should be sent to: Editor, **ST-LOG**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ST-LOG,** P.O. Box 16928, North Hollywood, CA 91615 or call (818) 760-8983.

Correspondence concerning a regular column should be sent to our editorial address (see Authors below), with the name of the column included in the address. We cannot reply to all letters in these pages; so if you would like an answer, please enclose a self-addressed, stamped envelope.

J.E. Publishers Representatives — Los Angeles: (213) 467-2266.
6855 Santa Monica Blvd., Suite 200, Los Angeles, CA 90038.
San Francisco: (415) 864-3252. Chicago: (312) 445-2489.
Denver: (303) 595-4331.
New York: (212) 724-7767.

Address all advertising materials to: Paula Thornton — Advertising Director, **ST-LOG**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

PERMISSIONS

No portions of this magazine may be reproduced in any form without written permission from the publisher. Many of the programs printed herein are copyrighted and not public domain.

Due, however, to numerous requests from Atari club libraries and bulletin-board systems, our policy does allow club libraries or individually run BBSs to make certain programs from **ST-LOG** available during the month printed on that issue's cover. For example, software from the January issue can be made available January 1.

This does not apply to programs which specifically state that they are *not* public domain and, thus, are not for public distribution.

In addition, any programs used must state that they are taken from **ST-LOG** Magazine. For further information, contact **ST-LOG** at (203) 645-6236.

SUBSCRIPTIONS

ST-LOG, P.O. Box 16928, North Hollywood, CA 91615 or call (818) 760-8983. Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Foreign: add $7.00 per year per subscription. For disk subscriptions, see the cards at the back of this issue.

AUTHORS

When submitting articles and programs, both program listings and text should be provided in printed *and* magnetic form, if possible. Typed or printed text copy is mandatory and should be in upper- and lowercase, with double spacing. If a submission is to be returned, please send a self-addressed, stamped envelope with your manuscript to **ST-LOG**, P.O. Box 1413-MO, Manchester, CT 06040-1413.

COVER PHOTOGRAPHY:
PETE TURNER©

INTERIOR PHOTOGRAPHY:
JOSEPH DRIVAS©

ILLUSTRATORS:
BOB MCMAHAN
GREGG MITCHELL
MIKE CRESSY
WAYNE T. GLADYSZ

THANKS TO:
IMAGE BANK WEST

C-manship:

A Complete GEM Application, Part 3

by CLAYTON WALNUM

This month we'll continue examining the *MicroCheck ST* source code by adding the functions we need to get the menu bar working. We'll also be looking at the code that controls the sliders and arrows in our window. But first . . . .

A Wee Bug

The ST, for some reason, overlooks some runtime errors. For example, a divide-by-zero operation, clearly illegal, doesn't seem to bother the ST at all (at least it doesn't generate any bombs; it does, however, cause an exception #5). Unless you're running the program in question from within the Laser C shell or with a monitor like *Templemon* installed, you'd never know if you got one of these exceptions.

When I was working on this month's listing, I discovered that there are times when the function *calc_vslid()* will try to divide by zero. I didn't notice this before because virtually all of *MicroCheck ST* was written using the older Megamax C, and, although the code was converted to Laser C, for some reason I never ran the program from the shell.

Anyway, you can fix the bug by adding the following piece of code to the beginning of *calc_vslid()*, right above the *wind_get()* call:

```
    if ( line_cnt == 0 )
        line_cnt = 1;
```

Marching Onward

Listing 1 is this month's portion of the *MicroCheck ST* source code. You should load the source code you've typed so far, delete the *do_menu()*, *do_wind_close()*, *do_arrow()*, *do_vslide()*, *do_hslide()* and *handle_button()* stubs (those do-nothing functions that we added for the linker's sake), and add Listing 1. You should leave in the *handle_keys()* stub, and, of course, all the stubs in this month's portion.

When you run the program (after compiling it, of course), you'll notice two big changes: the Quit option of the File menu now works (hallelujah!) and the horizontal scroll bar on the window works.

In point of fact, the entire menu bar is now working, but since we haven't added the code necessary to perform the functions chosen from the menu (except Quit), most of the menu choices still do nothing. Likewise, all the window controls are now in working order, but since the window is displaying nothing, most of them seem to be nonfunctional.

Now let's step through this month's listing.

Function *do_menu()*

Here we take the MU_MESAG (a menu message) passed from GEM and interpret it, sending program execution to the appropriate function. As you may recall (see the *C-manship* in the June '87 ST-LOG), the object number of the menu title can be found in the third element of the message buffer, and the object number of the selected entry within the menu can be found in the fourth element of the message buffer (in our case, *msg_buf[3]* and *msg_buf[4]*).

To interpret the message, we use nested *switch* statements. The outer *switch* checks *msg_buf[3]* to find out which of the menus was accessed. The inner *switch* statements (one for each of the menus) uses *msg_buf[4]* to route the user's request to the right function.

If you look closely at *do_menu()*, you'll see that every menu and every selection within each menu is represented here. Although the function is long, it is really quite simple. The only other thing of note here is the call, at the end of the function, to *menu_normal()*, which deselects (turns off the highlighting) the menu title chosen.

Function *do_wind_close()*

This function is called whenever the user clicks on the window's close box or selects the Close entry of the File menu. Because *MicroCheck ST* has three modes—edit, search and cancel—*do_wind_close()* has three sections, each of which handles one of the modes.

Since we'll be modifying the window, the first thing we must do is call *wind_update()* to lock the window from any other redraws. (At the end of the function, we unlock the window with the same function.)

If the user is in the search or cancel mode, we need to return to the edit mode. The first two sections of the *if* statement handle these situations. In both cases, the current mode is turned off (returning the program to edit mode), the window name is changed to show that the user is back in the edit mode, the window is redrawn and the menu entries are set appropriately. (Some of them are not available in every mode, so, according to the mode, some are enabled and some are disabled.)

If we're already in the edit mode when the user selects Close, we need to close the account. First, we bring up an alert box, asking the user if he's sure he wants to close the account. If he is, we save the account to disk, blank the window, reset the menu entries (almost everything will be disabled) and reset the window's title bar to show the user that no account is open.

Function *handle_button()*

Whenever the mouse button is clicked on *MicroCheck*'s work area, this function is called. If the mouse pointer was over the check window, then the user either wants to edit a check or wishes to mark it as cancelled.

If the program is not in the search or cancel modes, we call *edit()* to bring up the check dialog for the check he has selected, otherwise we call *canc_chk()*, the function that places the program in the cancel mode and allows the user to cancel transactions.

Function *do_arrow()*

Whenever the user clicks the mouse pointer on one of the window's arrows or slider tracks (not on the slider itself), GEM sends us a WM_ARROWED message (see the *C-manship* in the May '88 ST-LOG). This message comes in eight different flavors (only six of which are of interest to *MicroCheck*). The actual type of arrow message is contained in the fourth element of the message buffer. So in the case of *MicroCheck*, we call *do_arrow()*, passing it the value of *msg_buff[4]*. The user may be asking to move up or down a line, up or down a page, right or left a line (or character, actually) or right or left a page. Since *MicroCheck* allows the user to scroll the window right or left only by a full page, we don't need to worry about the WA_LFLINE and WA_RTLINE messages.

As with *do_menu()*, we use a *switch* statement to route the user's request to the appropriate function.

Function *do_uppage()*

If the user clicked in the portion of the slider track above the slider, *do_uppage()* takes over. Here we simply find out how many lines will fit in the window, subtract that value from the index number for the check displayed at the top of the window to calculate a new *cur_top* and redraw the window.

Function *do_dnpage()*

If the user clicked in the portion of the slider track below the slider, he wants to move down a page. We call *do_dnpage()*, which works much like *do_uppage()*, except that we add *lines_available* (the number of lines that'll fit in the window) to *cur_top* rather than subtract it.

Function *do_upline()*

When the window's Up arrow is clicked, *do_upline()* springs into action, moving the window up one line. Moving up or down a single line is, if it's to be done elegantly, much more complex than moving an entire page. When we move up or down a full page, we have no choice but to redraw the entire window in the normal way, since none of the data we want to display is available anywhere on the screen.

However, when we move up or down a single line, all the data we need, except one line, is on the screen. If we want the scrolling action to be smooth, we can't just redraw the entire window in the conventional way. Instead, we raster (block move) the portion of the window containing data we can use up or down one line, then add the new data at the top or bottom, depending on which way we're moving. These block moves of screen memory are fast and help create the illusion of the window scrolling a line at a time.

If you've forgotten how the raster functions work, please refer to the *C-manship* in the March '87 ST-LOG. There isn't room here for a complete discussion.

So, in *do_upline()* we subtract one from *cur_top* (the index of the check shown at the top of the window), raster a portion of the window, starting with the top line and extending down to the next to the last line, down one position, then replace the top line with the new current top. Simple (well, almost) and elegant.

Function *do_dnline()*

This function works almost exactly like *do_upline()*, except it moves the window down one line instead of up.

Function *do_vslide()*

If the user chooses to use the vertical slider, *do_vslide()* will accommodate him. Handling the sliders is much more complicated than handling the arrows or slider tracks because the user can place the slider anywhere within the track. We need to calculate what portion of the data to display based on the slider's new position. (See the *C-manship* in the May '88 ST-LOG.)

The fourth element of the message buffer contains the new position of the slider. We use this value first to see if the slider has actually moved to a new position. If the slider has been moved, we calculate its position within our "document" (in *MicroCheck*'s case, the list of checks), set *cur_top* to the appropriate check index (the one that'll now appear at the top of the window) and redraw the window, placing the slider in its new location.

Function *do_hslide()*

The horizontal slider works in much the same fashion as its vertical counterpart. The main difference is that in *MicroCheck* we allow this slider to have only two positions: full right or full left. This makes the job much easier, since we don't have to do a lot of fancy calculations. Instead, we use the flag *left* to keep track of the horizontal slider's current position. When the user moves the slider, all we have to do is toggle our flag and change the position of the slider, redrawing the window as we do.

The only complication is that, since we are going to be displaying a different portion of the data, we need to change the labels in the window information line. This is easy to do with a quick call to *wind_set()*.

Function *do_quit()*

When the user is ready to leave the program and return to the desktop, he'll select the Quit selection of the File menu (at least, he will if we wants a safe exit). When he does, *do_quit()* will ask if he's sure he wants to quit. If he is, his data will be saved and the flag *all_done* will be set to true. This flag will then break us out of the *get_event()* loop and return the program to the end of *do_mcheck()*, where all our windows and GEM resources will be deleted from memory.

Putting it in Order

Over the last few months, we've put together a large chunk of source code. Now you might want to rearrange some of the functions so that the higher-level functions are at the top of the program, and the lower-level functions are at the bottom. This "top down" organization will make the program easier to read and trace.

As for me, I'll see you next time. ▪

***Clayton Walnum** is the executive editor of* ST-LOG *and* ANALOG Computing, *as well as the associate editor of* VideoGames & Computer Entertainment.

ST NEWS

GT-Elite Hard Drive

Soon to be available is Future System's GT-Elite hard-drive system, which includes a 30- or 50-megabyte hard drive in combination with a double-sided 3½-inch floppy drive. The GT- Elite comes with its own internal power supply, con-

troller/interface, floppy controller and real-time clock, and is designed to fit underneath a monitor.

The GT-Elite 30 is priced at $899, and the GT-Elite 50 can be purchased for $999.

Future Systems, Inc.
21634 Lassen Street
Chatsworth, CA 91311
(818) 407-1647
CIRCLE #120 ON READER SERVICE CARD.

FAX Surge Protector

The Safe-Fax Surge Protector, from Curtis Manufacturing Company, is designed to protect electronic and telecommunications equipment from power surges. The unit contains two RJ41-type jacks and a single-outlet receptacle that can accommodate such equipment as FAX machines, modems,

telephones and answering machines.

Says Tom Judd, president of Curtis Manufacturing, "Plugged into any standard AC outlet, Safe-Fax helps prevent damage to hardware and assures that data transmission through phone lines is unharmed by surges, spikes and glitches."

The Safe-Fax Surge Protector sells for $29.95 and comes with a one-year warranty.

Curtis Manufacturing Company, Inc.
30 Fitzgerald Drive
Jaffrey, NH 03452
(603) 532-4123
CIRCLE #121 ON READER SERVICE CARD.

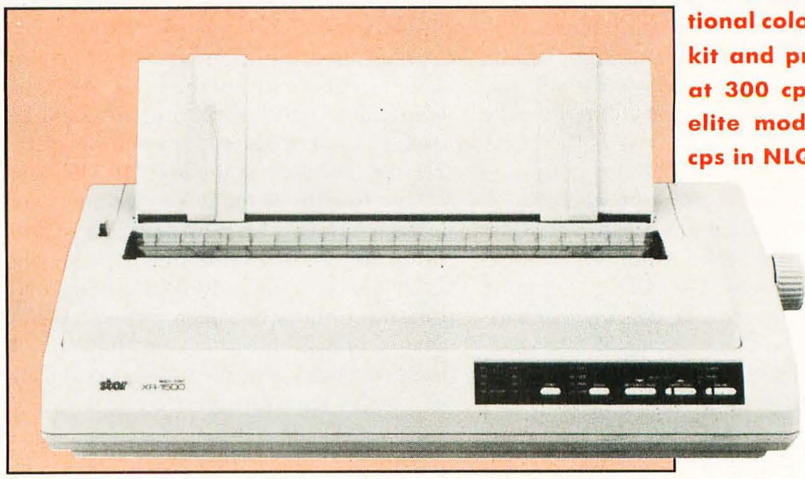
Multi-Font Printers From Star

Star Micronics has announced two new printers featuring eight resident fonts. The two printers, the XR-1500 and XR-1000, will be shipping by the time you read this. They are nine-pin dot-matrix printers that not only have multi-font capability, but can also produce color printouts when using the optional color-printing kit.

According to Brian Kennedy, director of marketing and sales, "These new printers have the ideal combination of features that users demand in output devices, including speed, multiple fonts, versatile paper handling, durability and reliability. In addition, Star Micronics has designed the XR-1500 Multi-Font and the XR-1000 Multi-Font printers to be the quietest in their price/performance categories, with noise levels of 51 and 50 decibels, respectively, when operating in the quiet mode."

The XR-1500, which features a 15-inch carriage as well as Epson EX-1000 and IBM Proprinter emulation, is priced at $799. The XR-1000, with the standard ten-inch carriage and Epson EX-800 and IBM Proprinter II emulation, carries a price tag of $579.

Star Micronics
200 Park Avenue, Suite 3510
New York, NY 10166
(212) 986-6770
CIRCLE #122 ON READER SERVICE CARD.



The XR-1500 Multi-Font is the first wide-carriage nine-wire dot-matrix printer with eight resident fonts. It features an easy-to-use front control panel, an optional color-printing kit and print speed at 300 cps in draft elite mode and 75 cps in NLQ mode.

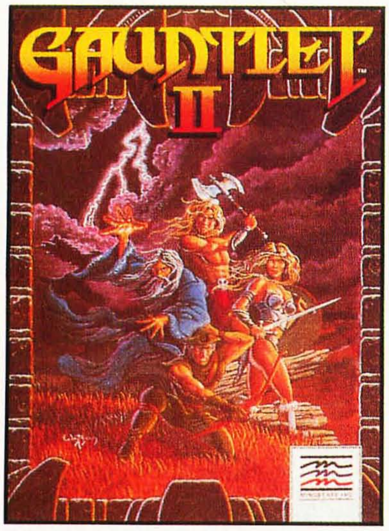
New Titles From Mindscape

Mindscape has announced five new game titles for the Atari ST. First on the list is the long-awaited *Gauntlet II*, which the company claims is an accurate rendition of the popular coin-op machine.

Also coming up are *Fiendish Freddy's Big Top O' Fun*, a slapstick circus game that allows the player to perform six dangerous center-ring stunts; *Harley-Davidson: The Road to Sturgis* (no price available), in which you drive your 1340cc Hog on the road to Sturgis, South Dakota, to perform in a series of contests and events; *After Burner*, Sega's arcade flight simulator; and *Shinobi*, yet another popular Sega arcade title in which you get to try your hand at being a Master Ninja.

Gauntlet II, *Fiendish Freddy's Big Top O' Fun* and *Shinobi* should be available by the time you read this, with the other titles being shipped between the end of 1989 and early 1990. The games are priced at $49.95 each.

Mindscape
3444 Dundee Road
Northbrook, IL 60062
(312) 480-7667
CIRCLE #123 ON READER SERVICE CARD.



Gauntlet II

Cutting-edge programming makes *Gauntlet II* the best home-computer adaptation
yet of an arcade game. Technical wizardry captures
the superlative graphics, sound and excitement of the arcade megahit.

Fiendish Freddy's Big Top O' Fun

Step right up to the greatest show on disk, but watch where you step! *Fiendish Freddy's Big Top O' Fun* offers a satiric, twisted look at six circus events, rendered in slapstick style and brought to life by breakthrough, movie-quality animation and art.

Capcom Enters ST Game Market

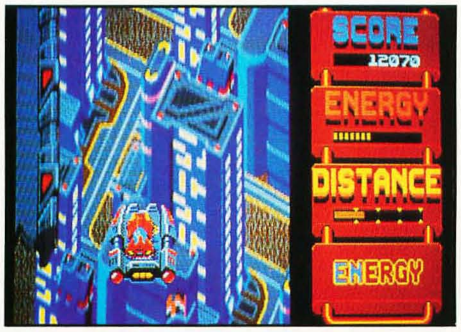
Coming from Capcom, the company that has produced some of the top games for the Nintendo Entertainment System, are three new games for the Atari ST. *Last Duel*, *Side Arms* and *LED Storm* are described as "three futuristic adventures."

In *LED Storm*, the player competes in a jet-car competition that takes him through futuristic scenes, such as elevated skyways and a high-tech metropolis. According to Capcom, "*LED Storm*'s music is so innovative that its score won awards in Europe."

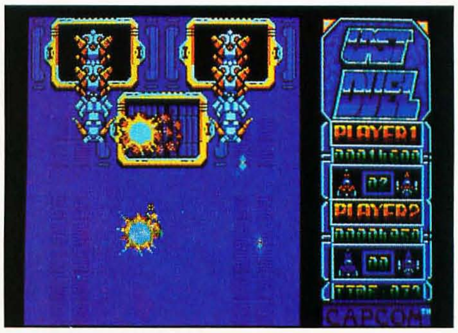
Last Duel plunges one or two players into an enemy stronghold in search of a kidnapped princess. To get around, the players utilize a ship that can transform from a land vehicle to a jet fighter.

Side Arms is described as a space-opera adventure in which the player must defend Earth from hostile aliens using a special space suit outfitted with various deadly weapons.

Capcom U.S.A., Inc.
3303 Scott Blvd.
Santa Clara, CA 95054
CIRCLE #124 ON READER SERVICE CARD.

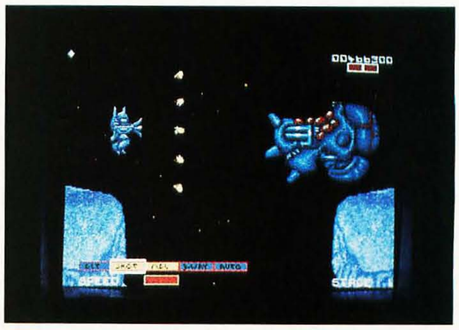


LED STORM



LAST DUEL

"THREE FUTURISTIC ADVENTURES"



SIDE ARMS

U-F-OTO

BY ALBERT BAGGETTA

GAME

LOW RESOLUTION

NOTE: The program listing included here is only for those people interested in programming with GFA BASIC. In order for it to run properly, this program requires a picture file, which cannot be reproduced in listing form in this magazine. The complete program is available on this month's disk or from the databases of the ST-LOG *ST Users' Group on DELPHI.*

In the game *U-F-OTO* (pronounced U-F Foto), you get to play the role of a freelance photographer trying to get work for a big-time newspaper. There are two things you need: a big event and opportunity. In this game you get both. The event is about as spectacular as they get: an invasion by a fleet of alien flying saucers, a la *War of the Worlds.* (Pretty original, huh?) Exclusive photos of this event would shoot any photographer (no pun intended) right to the top of the industry's charts.

About three in the morning the phone rings. You flop over and pick up the receiver. It's Cinnamon, your girlfriend who lives in the adjacent apartment. There's panic in her voice. "UFOs," she screams, "Look out your window!"

Here it is! Your big chance for a scoop photo! You fly out of bed, grab your 35mm camera (loaded with 18 frames of very fast film), pop on your flash (so maybe they will come real close) and run to the window, tripping over everything in the dark. You're amazed. It's perfect. They're all over the place, and here you are with a hot camera, ready to take pictures. Let's see now. ...the news photo lab closes at 4:00 a.m. What time is it now? Holy smoke, it's 3 a.m. Gotta work fast. And they have to be great shots.

ILLUSTRATION MIKE CRESSY

Playing the Game

Double-click on the file PHOTO.PRG on your disk. After the title screen, you will get a partial top view of the neighborhood. If you look closely, you will see some of your neighbors looking at the celestial spectacle. At the lower-right of the screen is your 35mm camera. It has a shutter release on the left, which is operated by pressing the left mouse button. In the center of the camera, at the top, is the simulated viewfinder. This is a single-lens reflex camera, so what you see is what you will get. The viewfinder is represented on the screen by a dotted cursor that flies around the screen area as you move the mouse. As with a real camera, this is where you frame your subject—you are going to be very busy trying to line up some of the saucers as they whiz by. After you align a space vehicle in your finder, click the left mouse button to take the picture. A flash will pop, and your picture will be captured on film.

You will find the film advance on the top-right of the camera. This is used to advance to the next frame. After you take a picture, the cursor will change to a pointing finger. To go to the next frame, point the finger to the center of the film advance on the camera and click the right mouse button. The number in the film window on the back of the camera will advance by one. In case you forget to do this, the camera will start beeping. After a little practice, these movements will become second nature to you.

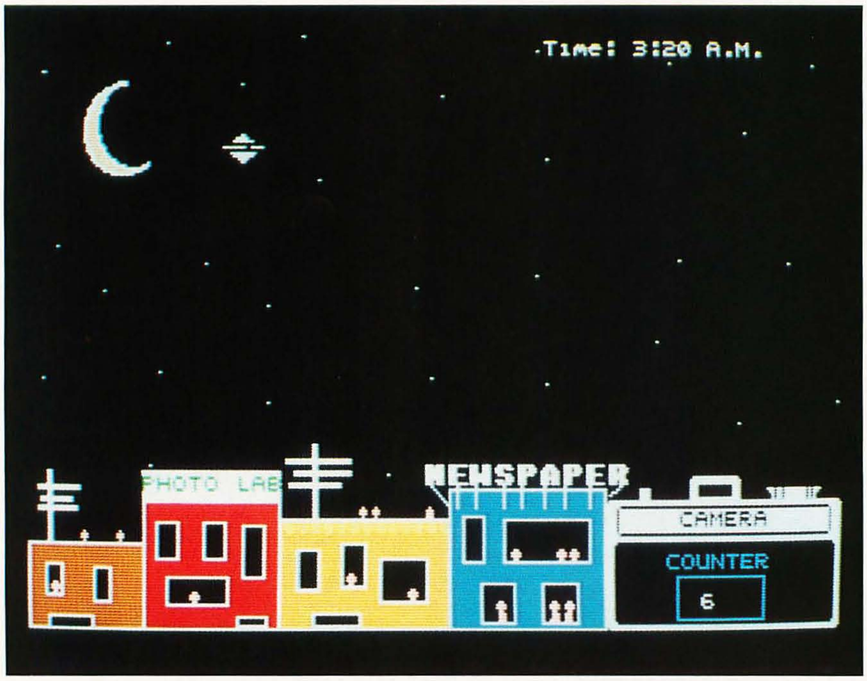
Now this all sounds easy, and it would be simple to come up with some good photos of the saucers—if it were not for a few built-in difficulties of the game. First of all, there is the varying speed of the saucers; you never know how fast they will be traveling. Also, you cannot always tell when they will enter and where they will enter from. Even when you get a bead on one, it does not stay on a steady path. To further complicate matters, the saucers can sometimes detect that someone is watching them (and maybe even trying to photograph them), so they set up an unexpected shield, which makes them invisible. At times, this aggravating habit of theirs can ruin a great shot.

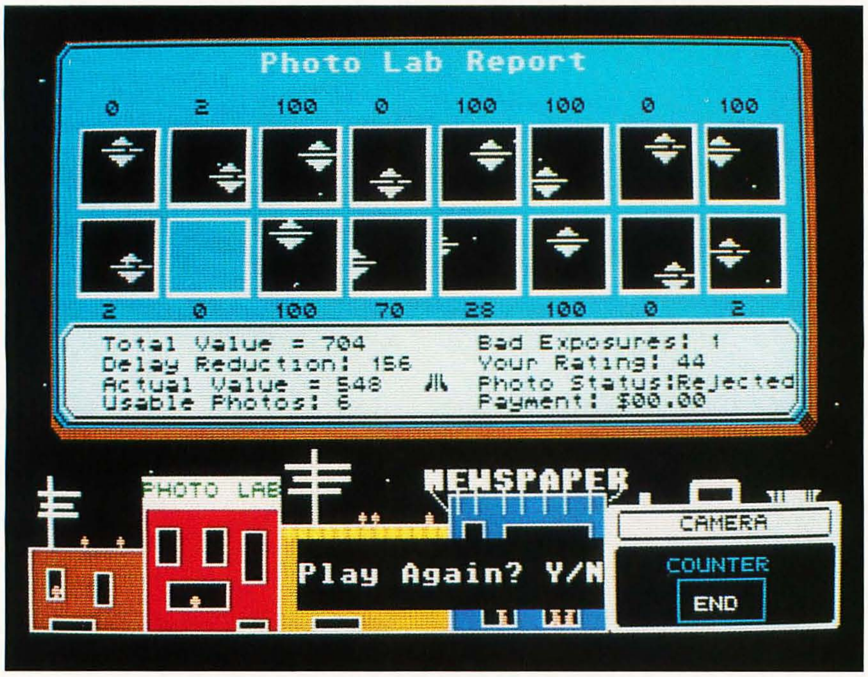
Sometimes, after a certain number of passes, the saucers can detect that you are taking pictures. Since the inhabitants frown upon this, they have set their computers (Ataris, no doubt) and their infrared shutter-releases to activate any cocked shutters. So don't be surprised if your camera goes off accidentally, giving you a scare and a bad exposure.

You should also keep conscious of your deadline. You have to finish the photo session by 4:00 a.m., or the newspaper won't even look at your photos until next time. And you know that there is no deader news than old news. So keep an eye on the clock in the upper right-hand portion of the screen. If it registers 4 a.m., forget it. The game will end.

This clock, by the way, is ticking all the time—even while the film is waiting to be advanced, so don't dawdle. Sometimes it is difficult to overcome the time factor because you do want to get your best pictures for the paper, and the ships do not always cooperate with you.

Are there any other problems? Yep. Let's say you get 16 photos to send to the newsroom photo lab. The paper must judge these as usable for the publication in terms of their quality and your dependability as a photographer. So they send them through a special scanner that examines them, pixel for pixel, and each is rated in points of quality and usability





```
Total Value = 704            Bad Exposures: 1
Delay Reduction: 156         Your Rating: 44
Actual Value = 548           Photo Status:Rejected
Usable Photos: 6             Payment: $00.00
```

Play Again? Y/N

(don't bother to try and figure it out—these people are an esoteric group and won't like your prying into their business). In any event, your pictures will be displayed at the lab and the values will be revealed under each print as part of the final report.

You might have a few good pictures, and you would think that would be enough to make the paper anxious to get you as a staff photographer. But this newspaper is very finicky. They accept only photos from photographers whose ratings (based on picture quality) are high. So you might have many good photos, but a low rating will get you rejected. If you clock below a 50 rating, you can forget fame and fortune.

What can cause a low rating in your pictures? Several factors: 1) poor-quality photos, 2) slow reaction time, 3) bad exposures, 4) lens reflections and 5) flashback. Miss the ship, getting a picture of space junk or buildings, etc., and you won't get many points. The amount of time you take to click the shutter is measured, and this amount is also deducted from your final score. Bad exposures (forced by the passing ships) won't get you any points. Fake ships can also dent your score, and if this doesn't frost you, nothing will. You might have what looks like a good picture of a saucer, but which is merely a reflection in your lens. The scanners in the newsphoto room will detect a reflection and will give you no points for the picture (maybe a few for background objects), no matter how complete it seems to you. A quick eye and a fast finger will allow you to identify reflections and prevent capturing them. Hint: Look for ships with small dots on them—these are lens images trying to fool you.

Finally, there is the problem of flashback. Sometimes your flash will reflect off a passing ship, causing a golden glare to cover parts of the ship. This in turn will cause an imperfection in the photo, rendering it useless. After you have delivered your film to the lab (you will be notified of your success on-screen) and your film is processed and analyzed, you will get a full on-screen report, just below your prints. The data is presented as follows:

1. Your total score.
2. Your delay factor (subtracted from the total score).
3. Your actual score (this is used by the news office to evaluate you).
4. The number of usable photographs, if you are accepted.
5. The number of bad exposures you got.
6. Your rating.
7. Your status: rejected/accepted.
8. Your payment for this job—if accepted.

U-F-OTO is a game of strategy and pure luck that can be enjoyed by children of all ages. You can accumulate more photo points by waiting for the real ships, as opposed to the meager points you will get for anything else you capture. On the other hand, as you wait, those delay points are being deducted from your total point value. Delaying too long might also jeopardize any chance of getting a good picture because the alien ships might cause your camera to prematurely expose a frame of film.

Because of these factors, the game is tantalizing enough to play over and over. After a while, it becomes a matter of pride to try to become a successful photographer. ■

***Albert Baggetta** is an English teacher and a professional guitarist. He lives in Agawam, Massachusetts, with his wife, Beverly, and his two children. He frequently can be found wandering the* ST-LOG *SIG on DELPHI.*

How much is YOUR time worth?

Turbo ST turns your 520/1040 ST into a speed demon. Check out **Turbo ST** at your local dealer or download our free demo from your favorite bulletin board.

Upgrades to version 1.6 are only $5 with your original disk. For more information, call (407) 657-4611, or write to SofTrek, P.O. Box 5257, Winter Park, FL 32793.

CIRCLE #101 ON READER SERVICE CARD.

TOOLS FOR

One aspect of C is that the function libraries supplied with different C compilers are not all created equal. Some libraries contain functions not included in others.

BY KARL E. WIEGERS

When you begin programming in a new language, you look for commands and functions to perform the kinds of operations you've become accustomed to in other languages. This helps ease the pain of transition into the new language. However, not all high-level languages are created equal. The absence of familiar and useful commands is highly frustrating to someone struggling to work with a new language.

I encountered this frustration when I began programming in C not long ago. If you come from a BASIC background, you're used to all sorts of functions for performing character-string manipulations, like LEFT$, RIGHT$ and MID$. I do a lot of programming in REXX, a nice interpreted language available on IBM mainframe computers. REXX also has a wide variety of string-handling functions, including many not available in even the finest BASICs. Face it: I was spoiled by the versatility of REXX when I began coding in C. What's a boy to do?

The obvious solution is to build some tools by writing C functions to duplicate the REXX and BASIC string-handling functions I know and love. Actually, the process of tool-building is a good way to learn a new language, although it is not without pain. To help you avoid the same kind of pain (although most of you have never heard of REXX), in this article I present C versions of seven of my favorite REXX-type functions (COPIES, LEFT, RIGHT, SUBSTR, WORDS, WORD and OVERLAY), plus another function that's used by my new RIGHT function (STRREV).

YOUR C CHEST

The New Functions

The names of these functions are pretty much self-explanatory, but I'll give you brief descriptions here. The source listings are extensively commented in case you want to see how it's all done.

The function *copies()* takes two arguments, a pointer to a character string and an integer. It returns a pointer to a string of concatenated copies of the supplied input string, with the integer argument specifying the number of copies to make. For example, if you called *copies()* with arguments of "Now" and 4, the string created would be "NowNowNowNow." The original string is not changed.

The function *left()* returns a pointer to a character string containing the leftmost part of a supplied input string. Its arguments are a pointer to the input string and the number of characters from the left to keep. If you ask for more characters than are in the original string, the output string is padded on the right to the desired length with blanks. The call *left("Now hear this," 6)* will return "Now he." The original string is not changed.

The function *right()* does the same thing, but for the right part of a string. The call *right("Now hear this," 6)* returns "r this." This function can be used to right-justify a string in a field longer than the original string since padding blanks are placed on the left. The original string is not changed.

The function *strrev()* reverses the characters in a supplied input string; the original string is lost in the process (the function could easily be changed so this doesn't happen, if you like). This function is supplied as a library function with some C compilers.

The function *substr()* performs like the MID$ function in BASIC, returning a pointer to a substring of a supplied input string, beginning at some character position and extending for a specified number of characters. The call *substr("Now hear this," 3, 5)* returns "w hea." The original string is not changed.

The function *words()* counts the number of blank-delimited words in a supplied input string and returns a short integer. The call *words ("Now hear this")* will return the value three.

The function *word()* returns a pointer to a character string containing a specified word in a supplied input string. The call *word ("Now hear this," 2)* will return "hear." The original string is not changed.

Finally, the function *overlay()* is a bit more complex. It overlays one character string (the guest) upon another (the host), beginning in a specified character position in the host string and extending for a specified number of characters of the guest string. If the guest string is shorter than the requested length, it is padded with a supplied pad character. The call *overlay ("see," "Now hear this," 5, 4, "K")* would return "Now seeK this." Again, the original strings are not changed.

Now let's look at some aspects of the C language that influenced why and how these functions came into existence.

Some Pointers on C Strings

C does not have a character string data type as such. Instead, a string is represented by a one-dimensional array of single-byte characters terminated with a null character (shown as \0). Array manipulations in C often are most easily handled by using pointers; this is certainly true of the character-string operations used in these functions. A pointer to a character string can be initialized by assigning some string constant (any sequence of characters enclosed in double quotes) to the pointer variable. Here's a sample variable definition:

```
char *charptr = "Some literal string";
```

You can also define a pointer to a character string by allocating the desired number of bytes using a function like *malloc()* or *calloc()*. Be sure to leave room for the extra null character at the end of the string and be sure that the value returned from *malloc()* is in the form of a pointer to a character. Here's an example of how to allocate storage for a ten-byte character string using a pointer variable called *charptr*:

```
char *charptr;
charptr = (char *) malloc (11);
```

Note that I've performed a cast to make sure that the value returned from *malloc()* is of type pointer to *charptr*. This cast may not be required with all C compilers, but it is with Laser C.

Another aspect of C is that the function libraries supplied with different C compilers are not all created equal. Some libraries contain functions not included in others. For example, a common C function is *strtok()*, which tokenizes (breaks apart) a character string based on specified delimiter characters. If the delimiter is a blank character, *strtok()* provides an easy way to separate a string into its individual words. The function *strrev()*, which reverses the characters in a string, is also found in some, but not all compiler libraries. I used both *strtok()* and *strrev()* in my initial versions of some of these string-handling functions.

The listings included in this article were written for the Laser C compiler. Unfortunately, the Laser C library contains neither *strtok()* nor *strrev()*. Hence, I wound up writing my own *strrev()*. I also wrote new *word()* and *words()* functions that didn't require *strtok()* at all. So much for the vaunted portability of C programs. If you do end up programming for multiple systems, check the function libraries carefully beforehand to avoid compiler-specific code.

MOVING?

DON'T MISS A SINGLE ISSUE

Let us know your new address right away. Attach an old mailing label in the space provided below and print your new address where indicated.

DO YOU HAVE A QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:

☐ New subscription. Please allow 4 to 8 weeks for your first copy to be mailed.

☐ Renewal subscription. Please include a current address label to insure prompt and proper extension.

☐ 1 year—$28.00. This rate limited to the U.S. and its possessions.

☐ Payment enclosed. ☐ Bill me.

P.O. BOX 16927, N. HOLLYWOOD, CA 91615

Name	
Street Address	
City	State ____ Zip ____

ATTACH LABEL HERE

(IF LABEL IS NOT HANDY, PRINT OLD ADDRESS IN THIS SPACE.)

Notes on Null

In C programs, null often is returned by a function to indicate that an error has taken place. For example, in these functions I'm returning a null if an attempt to allocate memory fails or if some other error condition is encountered during processing. The calling program is responsible for testing the value returned by a function and handling nulls appropriately.

The symbol "NULL" usually is defined in a header file with a value of zero, like this: *#define NULL 0.* If a numeric result is returned, there's no problem. But most of these functions return a value of type pointer to character. What happens when you try to print something at address zero? Well, the *printf()* function in some C compilers is smart enough to print the literal value "NULL", or just a backslash character, which clearly shows that something evil has happened. Not so with Laser C. Instead, you get an addressing (or bus) exception, which shows up as two bombs. Not fatal (as in reboot the computer), but definitely unhealthy for the program.

The examples in listing CHARTEST.C are carefully crafted to avoid returning a null. However, it is safer to test the value returned from each function for null before using it, something like this:

```
char *string = "Sample string";
char *answer;
answer = strrev ( string );

if ( answer == NULL )
   printf ( "Something horrible has happened." );
else  printf ( "Reversed string is: %s\n", answer);
```

Sample Programs

Two source code listings accompany this article. CHARTEST.C (Listing 1) is a demo program that exercises the eight string-manipulation functions. It illustrates the syntax and the results obtained, with several examples for each function.

The second program listing contains the source code for all eight of our new functions, all strung together. This file exists as CHARFUNC.C on your ST-LOG disk. In practice, you don't want to simply compile this entire file and link-in the resulting object file with your own program. That way you might be including object code for unused functions, thereby making your .PRG file larger to no useful purpose. You're better off including just the object files you need.

Laser C provides a convenient way to manage a collection of your own functions like this through its archiver/librarian utility called AR.TTP. AR.TTP lets you combine several object files (or any other sort of files) into a single archive file, which has an extension of .A. You can execute the AR.TTP program from within the Laser C shell by simply typing the command you want into an empty window, highlighting the entire command with the mouse and pressing Control-Return. Here's a typical command line:

```
F:\UTILS\AR.TTP RV \REXXOID\CHARFUNC.A
\REXXOID\STRREV.O \REXXOID\RIGHT.O
```

This sequence assumes that the AR.TTP file is in folder UTILS on Drive F:. The R function says that I want to replace or add a file to the archive file, and the V (verbose) function tells me what is going on as the operation progresses. The name of the archive file is CHARFUNC.A, and it's in folder \REXXOID. I'm adding files STRREV.O and RIGHT.O to this archive file, both of which also are in folder /REXXOID.

It takes a bit of getting used to, but the archive function can really cut down on file clutter. It also simplifies the linking process, as you can specify archive files in the list of object files to link together. Only the required members of the archive file will be extracted and used during linking.

One Last Thing

While testing these functions from within the Laser C shell environment, sometimes I encountered execution errors, like the dreaded exception 02 (equal to two bombs from the GEM desktop). Fortunately, Laser C provides a graceful exit from most such failures. I'd wrack my brain trying to find out what was wrong, but to no avail.

Imagine my relief when the very same program ran *just fine* from the desktop, although it failed in the shell. I guess the shell can't completely simulate the real-world GEM environment your program may encounter. The moral is to test all programs from the desktop as well as the shell before concluding that they are either right or wrong. ▪

***Karl Wiegers**, Ph.D., spent the 70s learning how to be an organic chemist, then spent the 80s wrestling with computers. He is now a software engineer in the Eastman Kodak Company Photographic Research Labs. He hasn't selected a career for the 90s yet.*

VideoGames & Computer Entertainment™

12
ALL COLOR ISSUES
ONLY $19.95
Save over $15 off the cover price!

- GAME REVIEWS
- ARCADE ACTION
- STRATEGY GUIDES
- TECHNICAL REPORTS
- COMPUTER SOFTWARE

VideoGames & Computer Entertainment™
P.O. BOX 16927, N. HOLLYWOOD, CA 91615

☐ **Yes!** Sign me up for 12 issues for only $19.95—I'll save over $15!

☐ Payment Enclosed — Charge My ☐ VISA ☐ MC	NAME________
# ________ EXP________	ADDRESS________
SIGNATURE________	CITY________ STATE________ ZIP________

Your first issue will arrive in 6 to 8 weeks.
WATCH FOR IT!!

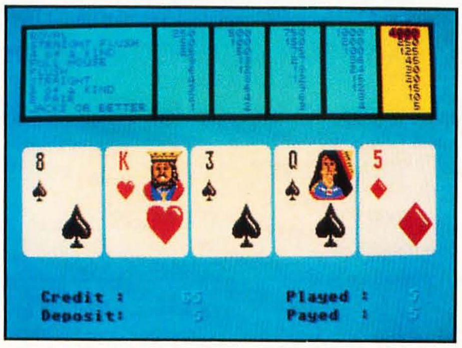
Money back on unused subscriptions if not satisfied!
Foreign—add $10 for postage.

Offer expires November 26, 1989 CJWYA

PD PARADE

GEORGE L. SMYTH

Last month I decided to celebrate my raise with a trip to Atlantic City. Round-trip bus fare from Washington, D.C., is $25, which is reimbursed by $25 worth of tokens upon arrival. Of course, my initial destinations, and those of my tokens, were the computerized slot and poker machines. After three hours at the machines, I had done extremely well—I only lost $12. The remainder of my time was spent waiting for one of the nickel machines to become available. At home I gamble in a different way, and this month's disk includes two games I have grown fond of: *Realistic Video Poker*, which will enable you to bone up on your casi-



no video poker skills, and *Dominoes* from Mulsoft, which will allow you to practice your hand-eye coordination.

Scott Foust has made available, via the shareware media, a game whereby you can lose your shirt in the comfort and privacy of your own home. *Realistic Video Poker*, Version 2.2., is a well-written, nicely presented clone of the popular casino video poker game. It features excellent graphics, fast play and an alternative to travel time and cost.

After entering "M" to receive $20 in quarters, a 15-second "commercial" for the shareware package is displayed in the hope that the owner of the disk will decide to support the programmer for his hard work.

The upper-left portion of the screen shows the possible winning consequences for a royal flush, straight flush, four of a kind, full house, flush, straight, three of a kind, two pair and a pair of jacks or better. The results are displayed for stakes of one through five coins across the top.

The bottom of the screen keeps track of the essentials—the amount currently credited to the player's account, the number of coins deposited and the number of coins played and won during the last round.

The player determines the number of coins to play by pressing the letter "C," which drops a single coin. Each time this key is pressed, the column at the top representing the potential earnings is highlighted in yellow. If the player decides to enter less than five coins, pressing the space bar deals the cards. Entering the letter "F" plays the maximum, five coins.

Upon receiving the hand, the player must determine which cards to hold and which to discard. This decision is made by entering the corresponding card number, one through five, of the cards to hold. In deference to the mouse, this method is faster, easier and closer to that used by the machines in Atlantic City. When the space bar is pressed, the cards not chosen to be held are flipped over and replaced. If the remaining hand yields a pair of jacks or better, the player wins. The amount of winnings is reflected by the strength of the hand and the number of coins played, as highlighted at the top of the screen.

The reward for supporting his efforts (sending in your donation) will come in the form of Version 3.0, which promises to include digitized sound, Double-Down, New Deck, Cash-Out to scoreboard and a report of all hands played, as well as the removal of the 15-second commercial.

I found this poker game to be entertaining and addicting, both essential elements of a good game. Having reviewed most of the commercial poker software on the market, I can say that Scott Foust has bettered several of these expensive games.

As this month's theme is games, I had to include a favorite of mine and my two children, the monochrome game of *Dominoes* from Mulsoft. This game is a test of both fast thinking and joystick skill.

The game begins with a blank screen and two dominoes drawn across from each other. The action starts as both dominoes are set in motion, leaving a trail of dominoes on the screen. The computer controls one, and the player's joystick controls the other. The object becomes apparent rather quickly: Lay the dominoes in such a pattern that the computer runs out of room to place dominoes first.

The dominoes can be placed in any direction, as long as a position already claimed by another domino is not challenged. If the computer is forced to move onto an already-occupied position, the player is moved to the next level and awarded 500 points in addition to the number of dominoes laid down. This forces the player to consider not only how to force the computer to run out of room, but also how to lay as many dominoes as possible before this happens. If the computer forces the player into this situation three times, the game ends.

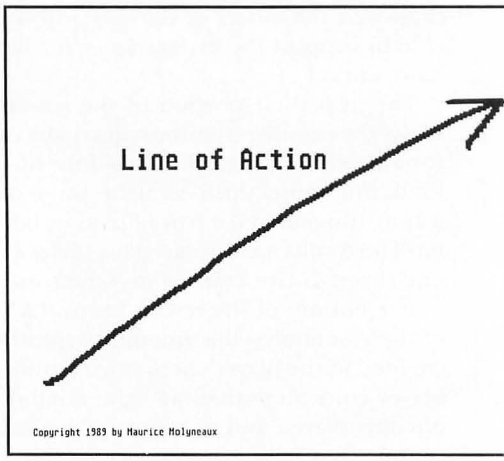
The first two levels are quite simple, and an opportunity is given to laugh at some of the computer's poor decisions. By the third round, the dominoes are definitely being set down faster and reaction time needs to improve. As the fifth round begins, it becomes much easier to make mistakes in spite of some of the computer's questionable moves. Reaching the seventh round forces more respect for the computer. I have only once passed the eighth level, as the dominoes are drawn at a pace I am not able to maintain.

If you want nonviolent action that begins simple and ends impossible, *Dominoes* will help you fill a good deal of your time.

Hopefully, you will have a chance to try both of these great games from this month's disk. Remember to remember the shareware programmers! ■

***George L. Smyth** holds a degree in psychology from West Virginia University and is currently employed as a programmer. He is the author of a series of tutorials on programming in Forth.*

THE ANIMATION STAND
TECHNIQUES



MAURICE MOLYNEAUX

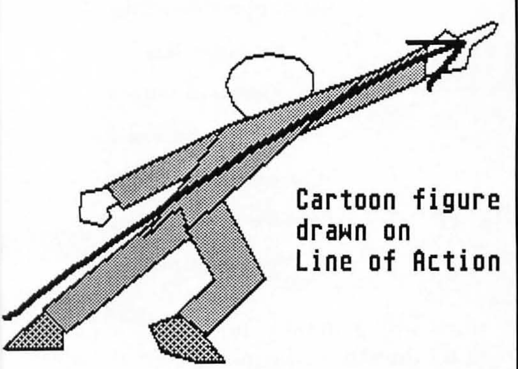
By now you should be into the first stages of your animation or just about ready to begin in earnest. To help you through the actual animation stage, this time the subject will be specific animation techniques. We'll start off covering a few basics and then go into more advanced topics, such as how to control the viewer's focus.

How-to

How you actually go about creating an animation depends on what software you're using and what you are animating. Here are a few software-specific pointers:

CAD-3D: I'm guessing you're using *Cyber Control* (at this point it's the only really decent way to generate a *CAD-3D* animation). Write the control script with options to allow you to "Watch" or "Record" the animation and also select whether to do either in wireframe or solid mode. You should also be able to run a solid/shaded test in draft mode, which takes less time than final draft. The first time out you should probably run through the entire animation in wireframe mode. If it looks good, record the wireframe (it is quite possible to create a wireframe animation too big for your ST to reload later, so be careful if it's complex) and then view it with a playback system (like ANIMATE4.PRG or *Cyber Paint*). Settle on any changes you feel necessary, make adjustments, then rerun the wireframe test. When all is right, do a test of a few frames in solid mode to see that the lighting and such looks correct. Then you can either generate a solid rough-draft mode animation for a final test or just go for the final-draft animation.

FIGURE 1



Line of Action: **The simplest way to illustrate this is to try to place an object (usually a character) so that the action it is undertaking follows along a single line and its form tends to follow that line.**

Cyber Paint: If you're doing anything more complex than using the video-paintbow-type effects provided by the program, such as actually drawing a series of frames, you may want to consider drawing only the moving elements at this point and leaving out the background (if any) because it will be easier to make changes without a background. If you had to move an arm to a different position, you might have to redraw the arm and then wipe out the old arm. If a background was present already, you'd probably have to redraw the part of the background which would show through the "hole" left by the original arm. It's better to animate the moving elements first and then underlay the background picture or animation. This permits easy changing of the background as well. Be careful with the defocus and antialias (*Cyber Paint 2.0*) effects because they can blur small details out of existence. If you have to animate a starship flying through space, antialias the ship before you add the stars, or you may find that instead of little bright dots,you'll have a galaxy full of little dull gray specks.

Film Director: When creating groups, stages, frames and so forth, rather than creating each from scratch, it's often much faster to copy an existing element and modify it. For instance, you can copy one group of parts that make up a robot and change only those cels that need adjusting.

If you want to get more complex than using the "boxed" action that groups and actors will provide, you can try animating individual collections of patterns one frame at a time. While this takes a lot longer, it allows you complete freedom to improvise and adjust the actions of your on-screen characters and objects.

In Character

I could write several long articles on the subject of character animation, but as this is a limited series, we'll just stick to a few useful basics. If one thing communicates attitude and expression better than almost anything else in a character, it's the eyes. Very small changes in their shape (often determined by the lids) and in the position, size, and color of the iris/pupil can greatly change the apparent feelings a character is experiencing.

A few hints: Try to avoid having the pupils "floating" in the middle of the whites of the eyes (if any), as it makes for a blank and unappealing expression. Such a "dead" look is useful in special circumstances, but is otherwise unattractive and does more harm than good. It's usually much better to keep the pupil toward the edges of the whites, rather than in the center. Use eyelids to heighten expression, and perhaps even eyebrows. Move the pupil in conjunction with these to strengthen an expression.

Furthermore, try to make the character's eyes "contact" what it is looking at. If something floats in front of the character's nose, have his eyes cross as he focuses on it. If the something passes above the character, have its eyes track it as it goes across. It will make the character look more alive than just pointing the pupils in the general direction of the object.

Beyond that comes the subject of characterization through motion, which Chuck Jones describes as nothing more or less than acting through a cartoon character. To quote Jones, "Just like Marlon Brando is Marlon Brando by the way he moves, not by what he looks like, in the same sense Bugs Bunny looks a certain way, but his actions make him what he is. . .characters have different personalities and different ways of moving. You can exemplify that because you can imitate Bugs Bunny."

In using this technique, you not only work out the actions the characters go through, but also define who and what the character is and determine how he will act in the circumstances in which you have placed him. If he's cocky, the way he carries himself will convey that. If he's insecure, specific postures and movement patterns will make that clear visually, without the need for explanatory dialogue and/or actions.

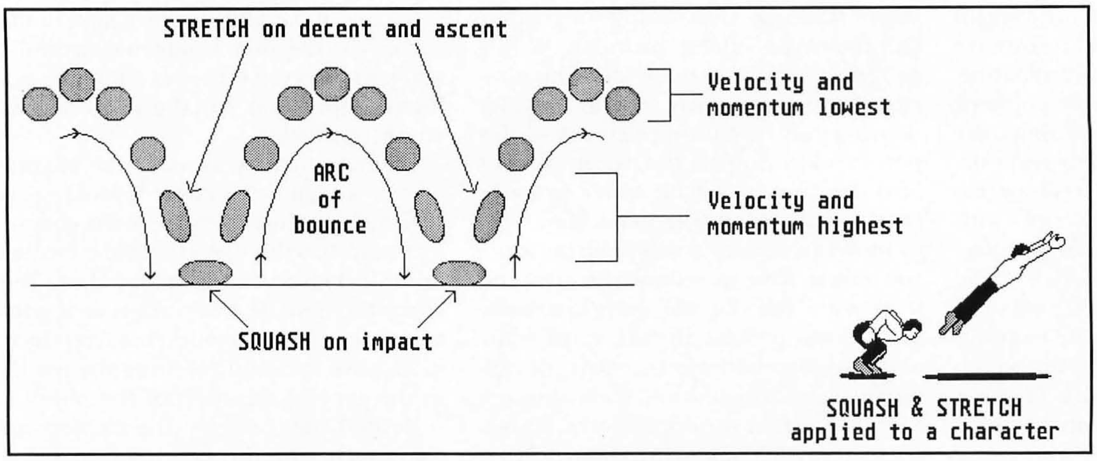
ABCs—Animation Basics With Computers

We have limited space, so we'll just hit these briefly. Specific examples appear in the accompanying illustrations.

Line of Action: The simplest way to illustrate this is to try to place an object (usually a character) so that the action it is undertaking follows along a single line and its form tends to follow that line.

(Figure 1). This has the benefit of making the thrust of the action easier to follow and clearer for the audience. It is also best to try to silhouette the pose so that the action is clear and out in the open. If you can blacken-in a character so he/she is just a solid outline but you can still tell what's going on, it's probably a good pose.

Anticipation: This means having a small lead-in action or a slight hesitation before a main action so the viewer will be prepared for the larger action and not miss it. The small action draws the viewer's attention to the focus of the scene so that


STRETCH on decent and ascent
Velocity and momentum lowest
ARC of bounce
Velocity and momentum highest
SQUASH on impact
SQUASH & STRETCH applied to a character

F I G U R E 2

he isn't looking at the wrong thing and misses the main action when it occurs. Such "anticipation" can include actions as diverse as a pitcher winding up before throwing a ball, a car "gathering itself up" prior to racing away, a purse snatcher coyly extending his hand before grabbing his target or a guy curling up into a ball before snapping into a human exclamation mark upon seeing a gorgeous girl.

Squash and Stretch: One of the most basic animation principles; it means treating objects as if they were flexible—as if made of putty. They always keep the same volume, but their shape changes depending on the action they are undergoing. This can apply to almost any image, from rubber balls to human beings. Hold one arm straight out to the side of your body, then bend your elbow and bring your hand back over your shoulder. Notice that when your arm is extended it looks thinner, and when flexed the muscles and flesh bunch up. This is a simple example of squash and stretch in the real world. Aside from such flexing within an object, the effects can be played out on entire figures. Generally, the "squash" happens at some point of impact (like a foot pressing on a pedal), and the "stretch" as the object makes a sudden move (like a rocket launching itself into the air).

The ball in Figure 2 demonstrates this, as does the little carbon figure. You can use squash and stretch for many types of actions; it's not limited to bouncing alone. Earlier, in "anticipation," I mentioned a boy curling up into a ball before straightening out into a reaction pose (see "Take" below). His curling up can be considered the squash, and his straightening the stretch.

Oddly enough, adding a little judicious squash and stretch can increase the impact of an action, even on items you would think it would look utterly wrong on. For example, a gun barrel could shorten and fatten slightly as the trigger is pulled, and then narrow and stretch out as the bullet emerges. If you're careful not to go so far that it becomes too obvious, you could even apply it to a giant boulder falling and crashing. If you did it right, the boulder would seem to fall faster and hit harder than if you just drew the rock with a rigid, unchanging shape.

Arcs: Stand up, put your hands at your sides, look straight ahead, then suddenly raise one hand and point at something. Now, analyze the path your hand took going from its resting position to pointing. Was it a straight line? Unless you're a mutant, chances are the path was curved. The fact is, few actions in real life are in straight lines. Usually they curve. A hand going from one position to another won't go in a straight line, it will follow an arc. You should remember this when animating, because while the shortest distance between two points may be a straight line, it looks unnatural and just plain lousy in animation. Sure, such lines are fine for starship flybys or cars zapping down a road, but for arms, legs, cars turning corners and so on, the action generally looks better if you make it follow an arc.

Slow In/Slow Out: This is sometimes also known as "ease-in/ease-out." Once again, bring your arm up and quickly point at something. Do it again, but this time pay attention to how fast your arm is moving at the various points of the action. The slowest movements are at the beginning and end of the action, with the fastest action happening in the middle. Try kicking or turning around. The same occurs. Whenever you make a motion, at first there is some buildup in speed (slow in), and at the end some "coasting" as you apply your body's brakes (slow out). Again, this is something you should apply to your animation. The speeds of various actions are generally not constant. The faster the action, the more severe the slow in/out effect. On an average arm swing, you might have four positions at each end of the movement, and only three poses between them. On a fast action, like a roundhouse punch, there may be a cluster of positions at the beginning and end of the action, with only a single in-between position. As with everything else, overuse of this technique can be bad, but using it where it seems natural will usually improve your animated action.

A lot of would-be animators don't realize that it's not hard to inadvertently draw the viewer's attention away from the real focus of the scene. Having a character dancing through the background would draw the viewer's eye away from the characters in the foreground.

The "Take": This is usually used for comic effect. Let's backtrack to our old example of a boy reacting to the sight of a gorgeous girl. The anticipation mentioned earlier prepares the audience for what comes next; and in this example, what comes next is a wild, overblown reaction, also known as a "take." The term isn't as unusual as it might sound. We've all heard of "double takes," right? Anyway, a take is an extreme and exaggerated reaction to something. If a character is startled and her eyes and dentures shoot out of her head (they invariably return to their original positions), that's a take. The aforementioned boy's body stiffening into a human exclamation mark is also a take. It's easy to overdo these, so they should be used sparingly and only where appropriate. As a rule, wild reactions rarely work well with realistic characters.

To make a take funny, the reaction

should be unexpected. When an ant turns around and finds an anteater breathing down his neck, the obvious take would be for him to stiffen or have his eyes bug out. A funnier way would be to have him look at the anteater, look at the audience, smile to himself as he jerks a thumb at the anteater as if it were a joke, and then unexpectedly launch into some wild reaction.

Weight and Momentum: This covers a number of things, mostly related to animating objects as if they were real and actually had weight. If you want to do abstract movement, this subject has little bearing. You can animate objects and characters as if they had no real mass and weight, but their movements would not seem real. A little reality can help in animation, though you probably don't want to go to the hyper-realistic extreme that studios like Disney have locked themselves into.

Weight can be indicated with as little effort as having a character squash a little bit as he comes down on each foot when walking or running, or by having a heavy object move like the real thing. If you pushed a metal chest across the floor, the overall speed and pauses in the action would be different than if you pushed an office chair on wheels. The metal chest is much more massive and more difficult to move. Likewise, someone carrying an anvil should walk as if he's supporting a great weight. The anvil is just a drawing, but the way it is moved lends it a sense of weight.

Weight has bearing on momentum. If you animated two cars, one traveling at low speed and the other going fast, with both having to make left turns, the slow-moving car would execute it easily and gracefully. Conversely, the fast-moving car would careen around the turn in an ungainly arc, probably leaning into the turn because it has to overcome its own considerable momentum in order to make the turn. The same can be said of a walking man and a running man. Furthermore, a ball thrown at a relatively low speed will connect with a catcher's glove with little impact, resulting in the gloved hand moving only a small distance. The harder the ball is thrown, the greater its momentum and the greater the impact. In many cases, the on-screen speed of an animated object is insufficient to indicate how fast it is traveling, but the effects of its momentum when it hits something or changes direction communicates this information quite well, in addition to making the animated action look more real.

Momentum and weight also have bearing with some animated objects. For instance, when I have Megabit Mouse turn around slowly, his tail lazily drags behind. If I make him spin suddenly, his tail comes off the ground, whips along behind him and continues moving even after his body has stopped turning. His feet have provided the resistance to overcome his body's momentum, but his dangling tail keeps going until it reaches the limit of its movement and/or its momentum runs out. And when he turns around, he doesn't just "freeze" in that position; his body sways as the last of the turn's momentum is lost, and he shifts as his weight settles.

Keep Your Eye on the Ball

As important as these animation basics are, they won't do you any good if your audience doesn't see them.

A lot of would-be animators don't realize that it's not hard to inadvertently draw the viewer's attention away from the real focus of the scene. Having a character dancing through the background would draw the viewer's eye away from the characters in the foreground.

If you want your audience to watch those foreground characters, you'd better do something about that distracting dancing guy.

It's easy enough to say not to have action that distracts from the focus, but how do you control what the viewer is looking at, particularly in a scene where two or more things may be moving?

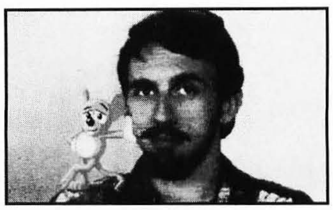
To illustrate this problem, let me drag up the storyboards for the *Art & Film Director* video. In some scenes Megabit Mouse was demonstrating the *Art Director* user interface. In many cases, he would do something, the on-screen arrow would cause something else to happen, he would react and then the arrow would cause something else to occur. It was necessary to have the viewer watch the arrow, then Megabit, go back to watching the arrow, back to Megabit and so on. The arrow's actions weren't complex as it just moved to a location, and the appropriate box, or whatever, would respond as if the ST mouse had been "clicked." So, if Megabit had just finished looking at something and the arrow was going to click on something else, the problem became one of drawing the viewer's attention away from the interesting-looking mouse to the dull arrow. Now, you might assume that just having the arrow move would be enough. Not so. By the time the viewer noticed the little arrow was moving, it had already done what it was going to do and the viewer would have missed it.

The solution I concocted was simple. Whenever I needed to draw attention to the mouse or the arrow, I would have the appropriate character or object do some little motion *before* actually getting on with its next task. For example, to draw the viewer's eye to Megabit, I might have him sigh or throw a funny look at the audience before getting on with this next action. The pointer was even easier, as I'd have it execute a little looping action before moving to its next task. Such actions last only a second, but they are enough movement to draw the viewer's attention to the desired object or character before the next action occurs. To help this effect, it's best to have any other moving objects on the screen either stop moving or fall into cycling, repetitive actions that are unlikely to distract the audience. In this way, the viewer should always be looking at the subject you want, not at something else.

To further keep the focus where you want it, you should keep the "path of action" where objects and characters will be moving relatively clear of visual distractions. In a forest scene, a clear path among the bushes will be a better location for moving objects than in front of a clutter of foliage. The same is true of an area more brightly lit than the surrounding scene. In such settings, moving items will be easier to spot; thus it's less likely the viewer will lose them in the clutter. This is particularly useful in complex settings where there are many details.

Oh No, Not Again . . . .

I was hoping to cover these techniques in a single article, but that doesn't look like it's going to happen. We'll finish up these topics next time, when we'll cover special animation effects and visual tricks, including how to minimize strobing and staccato movement effects in your animation and how to make objects look like they're moving fast. We'll also discuss explosions, lightning and other flashy effects. ■



Maurice Molyneaux *is a longtime user of Atari computers, whose particular fields of interest cover writing, graphics and animation. He enjoys good science fiction, loathes rap music and loves watching old Warner Bros. cartoons. Ideally, he would like to pursue a career creating animated films.*

U·F·FOTO
Listing 1:
GFA BASIC
2.0

U-F-OTO

```
' ********************************
' *           U-F-OTO            *
' *                              *
' *      A. BAGGETTA             *
' *      Copyright 1989          *
' *      by ST-LOG               *
' ********************************
'
Dim Photo$(16),Cdt%(16)
Cls
If Exist("TOWN.PI1") Then                       !Check for Picture File
  Print "Picture File Present..."
  Pause 50
Else
  Alert 3," | |Missing Winner File",1,"END",Ab
  Pause 100
  End
Endif
Cls
Setcolor 0,0,0,0
Setcolor 1,7,7,7
Deftext 2,0,0,2
'
@Build_sprite
@Title_screen
@Ld_screen
'
New_scren%=0
Count%=6
Get 0,0,319,199,Screen$
Game_top:
If New_scren%=1 Then
  Cls
  Put 0,0,Screen$
Endif
'
@Initialize
'
Graphmode 3
Deftext 1,0,0,6
For Jr%=1 To 4
  Text 85,100,"** SAUCER ATTACK **"
  '
  @Attack_signal
  '
Next Jr%
But%=0
Fc%=1
X%=0
Deftext 1,0,0,4
Text 263,192,Fc%
St:
Inc X%
'
@Change_sprite
'
Speed%=Int(Rnd*5)+1
Repeat
  If But%=1 Then
    Sound 1,10,5,5
    '
    @First_loop
    '
    Sound 1,0
  Endif
  If But%=0 Then
    '
    @Second_loop
    '
  Endif
  If Sunrise=60 Then       ! Get out of game if exceeded time
    Sprite X$
    Graphmode 2
    Deftext 1,0,0,4
    Deffill 0,2,8
    Pbox 200,3,300,8
    Text 200,8,"Time: 4:00 A.M."
    Graphmode 3
    Deftext 1,0,0,4
    For Jr%=1 To 6
      Text 85,100,"** DEADLINE EXCEEDED **"
      '
      @Time_up
      '
      Next Jr%
      '
      @Analysis_screen
      '
      Print At(14,22);"                "
      Print At(14,23);"Play Again? Y/N"
      Print At(14,24);"                "
      Goto Solong
    Endif
Until C%=1 And But%=0
Out:
'
' Flash the bulb
'
Setcolor 0,7,7,7
Pause 5
Setcolor 0,0,0,0
But%=1
If Ej%=1 Then
  Graphmode 3
  Deftext 1,0,0,4
  For Jr%=1 To 10
    Text 110,100,"** BAD EXPOSURE **"
    Pause 10
  Next Jr%
  Pic$=""
  Goto Ejump
Endif
'
' Get the picture of the saucer
'
Get A%,B%,A%+30,B%+25,Pic$
For Trav%=Col% Downto 1
  Sprite X$,Trav%,Row%
Next Trav%
'
@Saucer_sound_off
'
Sprite X$
Sprite A$
Sprite B$
Ejump:
Ka%=4
Kb%=6
Count%=Int(Rnd*(Kb%-Ka%+1))+Ka%
Ej%=0
Photo$(X%)=Pic$
If X%=16 Then
  Graphmode 2
  Deffill 0,2,8
  Pbox 260,188,285,195
  Deftext 1,0,0,4
  Text 263,192,"  "
  Text 260,192,"END"
  Graphmode 3
  Goto Fin
Endif
Pause 50
Col%=300
Row%=Int(Rnd*110)+1
Goto St
Fin:
'
' Photos delivered to the News lab
'
Graphmode 3
Deftext 1,0,0,6
For Jr%=1 To 10
  Text 35,100,"** PHOTOS DELIVERED ON TIME **"
  Pause 10
Next Jr%
'
@Analysis_screen
@Prt_values
@Prt_report
'
Solong:
Inkey$=""
Repeat
  Mouse M1,M2,M3
  P$=Inkey$
Until P$<>"" Or M3=1
If M3=1 Then
  Goto Game_top
Endif
If P$="Y" Or P$="y" Then
  Goto Game_top
Else
  Setcolor 0,7,7,7
  Setcolor 15,0,0,0
  Cls
  End
Endif
'
Procedure Change_sprite
  Chngsprt%=Int(Rnd*2)+1
  If Chngsprt%=1 Then
    X$=A$
  Endif
  If Chngsprt%=2 Then
    X$=B$
  Endif
Return
'
' ****************************
' Ld Degas Screen
' ****************************
Procedure Ld_screen
  Res=Xbios(4)
  Open "I",#1,"a:town.PI1"
  Ys=Inp(#1)
  Xs=Inp(#1)
  If Xs<>Res
    Alert 1," Wrong resolution ",1,"okay",Dummy
    End
  Endif
  For Ns=0 To 15
    Xs=Inp(#1)
    Ys=Inp(#1)
    Zs=Xs*256+Ys
    Cs=Xbios(7,Ns,Zs)
  Next Ns
  Seek #1,34
  Bget #1,Xbios(2),32000
  Close #1
Return
'
Procedure Film_count
  Graphmode 2
  '
  @Film_advance
  '
  Fc%=Fc%+1
  Deffill 0,2,8
  Pbox 260,188,285,195
```

Public Domain Software

#57 - Tease Me Adult Animation (Color Only)
#145 - Five Children's Programs (Color Only)
#352 - Lost Treasure (Lode Runner Clone) - Color
#393/394/533 - PrintMaster Graphics
#400 - 7 Disk Labeling Programs (w/100 Labels $6.95)
#443 - Intersect RAM Baby (RAM Disk/Print Spooler)
#456 - Bolo Breakout Game from Germany (1 Meg)
#470 - Two Virus Killer Utilities, Database and more
#475 - Werty's House of Horror (Adult Game, Color)
#500/600 - Publishing Partner Fonts
#511 - Dungeon Master Maps for Levels 1-7
#512 - Dungeon Master Hints/Character
#555 - The Assistant Chef - Electronic Cookbook
#557 - Children's Programs (Color Only)
#567/728 - Accessories
#575 - Sheet V2.0 - Shareware Spreadsheet
#588 - Pac Man, Hangman and 5 others (Color Only)
#599 - PageStream Fonts, Font Converter
#655 - ST Writer V3.0 w/Spell V2.8
#690 - Opus V2.10 GEM Spreadsheet (1 Meg/DBL)
#700 - ST Xformer V2.4 (Xformer Cable $19.95)
#721 - Drawing Programs

Introductory Offer - Above Disks Just
$2.99 Each
Call or Write for FREE Catalog
(800) 347-6760

BRE Software Dept. STL
352 W. Bedford, Suite 104
Fresno, CA 93711
Customer Service (209) 432-3072
Shipping $2.50 Ground / $4.00 2nd Day Air

VISA MasterCard

CIRCLE #102 ON READER SERVICE CARD.

U-F-OTO

```
  Deftext 1,0,0,4
  Text 263,192,"  "
  Text 263,192,Fc%
  Graphmode 3
  Pause 5
Return
'
Procedure Film_advance
  Wave 8,1,14,100,10
  Wave 0,0
  Count_saucer%=0
  Pause Int(Rnd*50)+1
Return
'
Procedure First_loop
  Showm
  Mouse A%,B%,C%
  Sun=Sun+0.05                  ! Increase deadline timer
  If Int(Sun)=>600 Then
    If Sunrise=60 Then
      Goto Bye2
    Endif
    Graphmode 2
    Sunrise=Sunrise+10
    Deftext 1,0,0,4
    Deffill 0,2,8
    Pbox 200,3,300,8
    Text 200,8,"Time: 3:"+Str$(Sunrise)+" A.M."
    Sun=0
    Graphmode 3
  Endif
  ' Put in lines to handle advance control by mouse
  If (A%>295 And A%<305) And (B%>150 And B%<160) And C%=2 Then
    '
    @Film_count
    '
    But%=0
    Hidem
  Endif
  Bye2:
Return
'
Procedure Second_loop
  Hidem
  Inc Sun                  ! Increase deadline timer
  If Sun=>600 Then
    If Sunrise=60 Then
      Goto Bye
    Endif
    Graphmode 2
    Sunrise=Sunrise+10
    Deftext 1,0,0,4
    Deffill 0,2,8
    Pbox 200,3,300,8
    Text 200,8,"Time: 3:"+Str$(Sunrise)+" A.M."
    Sun=0
    Graphmode 3
  Endif
  '
  @Saucer_sound_on
  '
  Sprite A$
  Sprite B$
  '
  ' Allow some saucers to disappear before going across the full screen
  '
  Reduce=Reduce+0.05
  Da%=50
  Db%=200
  Dis%=Int(Rnd*(Db%-Da%+1))+Da%
  If Col%<1 Or Col%=Dis% Then
    If Col%=Dis% Then
      '
      @Zing_noise
      '
    Endif
    '
    @Saucer_sound_off
    '
    If Col%=Dis% Then
      Graphmode 3
      Deftext 1,0,0,4
      For Jr%=1 To 8
        Text 100,100,"** ALIEN SHIELDS UP **"
        Pause 10
      Next Jr%
    Endif
    Sprite X$
    Pause Int(Rnd*50)+1
    '
    ' Count the number of saucers -- if  four to six flash a bad exposure
    ' and keep a record of it.
    '
    Inc Count_saucer%
    If Count_saucer%=Count% Then
      C%=1
      Inc Bad_exposure%
      Ej%=1
      '
      @Saucer_sound_off
      '
      Goto Bye
    Endif
    Sprite X$
    Col%=300
    '
    @Change_sprite
    '
    Row%=Int(Rnd*110)+1
    Speed%=Int(Rnd*5)+1
  Else
    Col%=Col%-Speed%
  Endif
  '
  ' Check the location of the saucer on the screen
  '
  Var%=Int(Rnd*10)+1
  If Var%>0 And Var%<6 And Row%<>20 Or Row%=110 Then
    Row%=Row%-1

    Endif
    If Var%>6 And Var%<10 And Row%<>110 Or Row%<2 Then
      Row%=Row%+1
    Endif
    Mouse A%,B%,C%
    Sprite X$,Col%,Row%
    '
    ' On occasion have a flash reflect back to the camera removing some
    ' points if a good ship is captured -- sprites overlap
    '
    Flsh=Int(Rnd*12)+1
    If Flsh=12 And C%=1 Then
      Sprite Z4$,Col%,Row%
      Sprite Z5$,Col%,Row%
      Color 1
      Box A%,B%,A%+30,B%+25
      Color 0
      Box A%,B%,A%+30,B%+25
      Sprite Z4$
      Sprite Z5$
    Else
      Color 1
      Box A%,B%,A%+30,B%+25
      Color 0
      Box A%,B%,A%+30,B%+25
    Endif
    Bye:
  Return
  '
  Procedure Analysis_screen
    Graphmode 2
    Deffill 15,2,8
    Defline 1,1,0,0
    Color 7
    Rbox 9,0,306,131
    Rbox 9,0,307,132
    Rbox 9,0,308,133
    Rbox 9,0,309,134
    Color 0
    Prbox 10,1,305,130
    Rbox 11,2,304,129
    Deftext 2,0,0,6
    Text 90,11,"Photo Lab Report"
    If Sunrise=60 Then
      Text 33,50,"Sorry...Deadline exceeded."
      Text 33,70,"We'll get another photographer."
      Pause 300
    Else
      Text 33,124,"Examining Data...Please wait."
      Color 1
```

PROGRAM·LISTINGS

```
          Restore Id_level_photo
          '
          ' Place the photographs on screen in the photo lab window
          '
          For Reed%=1 To 16
            Read X1%,X2%,X3%,X4%,X5%,X6%
            Put X1%,X2%,Photo$(Reed%)
            Box X3%,X4%,X5%,X6%
            Box X3%,X4%,X5%+1,X6%+1
          Next Reed%
          Restore Photo_data
          '
          ' Scan each picture to see if it is good or not
          '
          For Cd%=1 To 16
            Read Xdc%,Xdr%,Ydc%,Ydr%
            For X%=Xdc% To Xdr%
              For Y%=Ydc% To Ydr%
                If Point(X%,Y%)=2 Then  ! Check for white dots
                  Inc Cdt%(Cd%)
                Endif
              Next Y%
            Next X%
          Next Cd%
        Endif
      Return
      '
      Procedure Saucer_sound_on
        Wave 8,1,3,32000
      Return
      '
      Procedure Saucer_sound_off
        Wave 8,0
      Return
      '

      Procedure Build_sprite
        Restore Sprt
        Colr%=1
        For Mksprt%=1 To 7
          If Mksprt%=2 Then
            Colr%=14
          Endif
          If Mksprt%=3 Or Mksprt%=4 Or Mksprt%=5 Then
            Colr%=7
          Endif
          If Mksprt%=>6 Then
            Colr%=6
          Endif
          X$=Mki$(1)+Mki$(1)+Mki$(0)
          X$=X$+Mki$(0)+Mki$(Colr%)        !Color SPRITES
          For I%=1 To 16
            Read N%
            X$=X$+Mki$(0)+Mki$(N%)
          Next I%
          If Mksprt%=1 Then
            A$=X$
          Endif
          If Mksprt%=2 Then
            B$=X$
          Endif
          '
          ' Circle Sprites for the shields
          '
          If Mksprt%=3 Then
            Z1$=X$
          Endif
          If Mksprt%=4 Then
            Z2$=X$
          Endif
          If Mksprt%=5 Then
            Z3$=X$
          Endif
          '
          ' Flash Sprites for the backflash
          '
          If Mksprt%=6 Then
            Z4$=X$
          Endif
          If Mksprt%=7 Then
            Z5$=X$
          Endif
        Next Mksprt%
      Return
      '
      Procedure Zing_noise
```

```
  Sprite X$
  For Rz%=1 To 10
    Sprite Z3$
    Sprite Z1$,Col%,Row%
    Pause 1
    For I%=1 To 10
      Wave 1,1,1000,I%
    Next I%
    Sprite Z1$
    Sprite Z2$,Col%,Row%
    Pause 1
    For I%=10 To 1 Step -1
      Wave 1,1,1000,I%
    Next I%
    Sprite Z2$
    Sprite Z3$,Col%,Row%
    Pause 1
  Next Rz%
  Sprite Z3$
  Wave 1,0
Return
'
Procedure Prt_report
  '
  ' Print the report to the photo lab window
  '
  Text 153,117,Chr$(14)+Chr$(15)
  Text 30,104,"Total Value = "+Str$(Value%)
  Text 30,111,"Delay Reduction: "+Str$(Int(Reduce))
  Text 30,118,"Actual Value = "+Str$(Value%-Int(Reduce))
  Text 30,124,"Usable Photos: "+Str$(Usable%)
  Text 175,104,"Bad Exposures: "+Str$(Bad_exposure%)
  Rating=Int((Value%-Int(Reduce))/16)+10
  If Rating>100 Then
    Rating=100


  Endif
  Text 175,111,"Your Rating: "+Str$(Rating)
  Text 175,118,"Photo Status: "
  If Rating>50 Then
    Text 253,118,"Accepted"
    Text 175,124,"Payment: $"+Str$(Usable%*60)
  Else
    Text 253,118,"Rejected"
    Text 175,124,"Payment: $00.00"
  Endif
  Print At(14,22);"                    "
  Print At(14,23);"Play Again? Y/N"
  Print At(14,24);"                    "
Return
'
Procedure Attack_signal
  For At%=1 To 75
    For Aup%=1 To 12
      Sound 1,10,Aup%,4
    Next Aup%
  Next At%
  Sound 1,0
  Pause 30
Return
'
Procedure Time_up
  For Rtim=1 To 10
    Sound 1,10,11,7
    Pause 1
    Sound 1,0
    Pause 1
  Next Rtim
Return
'
Procedure Del
  For Delay=1 To 20
  Next Delay
Return
'
Procedure Initialize
  For Clean%=1 To 16      ! Clean up the arrays
    Cdt%(Clean%)=0
    Photo$(Clean%)=""
  Next Clean%
  New_scren%=1
  Setcolor 14,6,6,6
  Defline 3,0,0,0
  Sunrise=0
  Sun=0
  Ej%=0
  Deftext 1,0,0,4
  Text 200,8,"Time: 3:00 A.M."
  Rating=0
  Bad_exposure%=0
  Value%=0
  Usable%=0
  Col%=300
  Row%=Int(Rnd*110)+1
  Reduce=0
  Count_saucer%=0
  Hidem
  Defmouse 3
Return
'
Procedure Title_screen
  Color 1
  Repeat
    Inc Xstar%
    Cstar%=Int(Rnd*319)+1
    Rstar%=Int(Rnd*199)+1
    Text Cstar%,Rstar%,"."
  Until Xstar%=100
  Deftext 2,4,0,9
  Color 7
  For T_saucer%=1 To 7
    Color T_saucer%+3
    Box 21,21,295,175
    Pause 3
    Color T_saucer%+3
    Box 14,14,303,182
    Pause 3
    Color T_saucer%+3
    Box 7,7,310,190
    Wave 8,1,13,10000
    For Clm%=300 To 40 Step -2

Sprite A$,Clm%,90
If Clm%<=170 And Clm%>130 Then
  Text 180,100,"O"
Endif
If Clm%<=159 And Clm%>119 And T_saucer%=2 Then
  Sprite Z4$,169,95
  '
  @Del
  '
  Sprite Z4$
  Sprite Z5$,169,95
  '
  @Del
  '
  Sprite Z5$
  Text 169,100,"T"
Endif
If Clm%<=147 And Clm%>107 And T_saucer%=3 Then
  Text 157,100,"O"
Endif
If Clm%<=135 And Clm%>95 And T_saucer%=4 Then
  Text 145,100,"-"
Endif
If Clm%<=123 And Clm%>83 And T_saucer%=5 Then
  Sprite Z4$,133,85
  '
  @Del
  '
  Sprite Z4$
  Sprite Z5$,133,85
  '
  @Del
  '
  Sprite Z5$
  Text 133,100,"F"
Endif
If Clm%<=111 And Clm%>71 And T_saucer%=6 Then
  Text 121,100,"-"
Endif
If Clm%<=99 And Clm%>31 And T_saucer%=7 Then
  Sprite Z4$,109,95
  '
  @Del
  '
  Sprite Z4$
  Sprite Z5$,109,95
  '
  @Del
  '
  Sprite Z5$
```

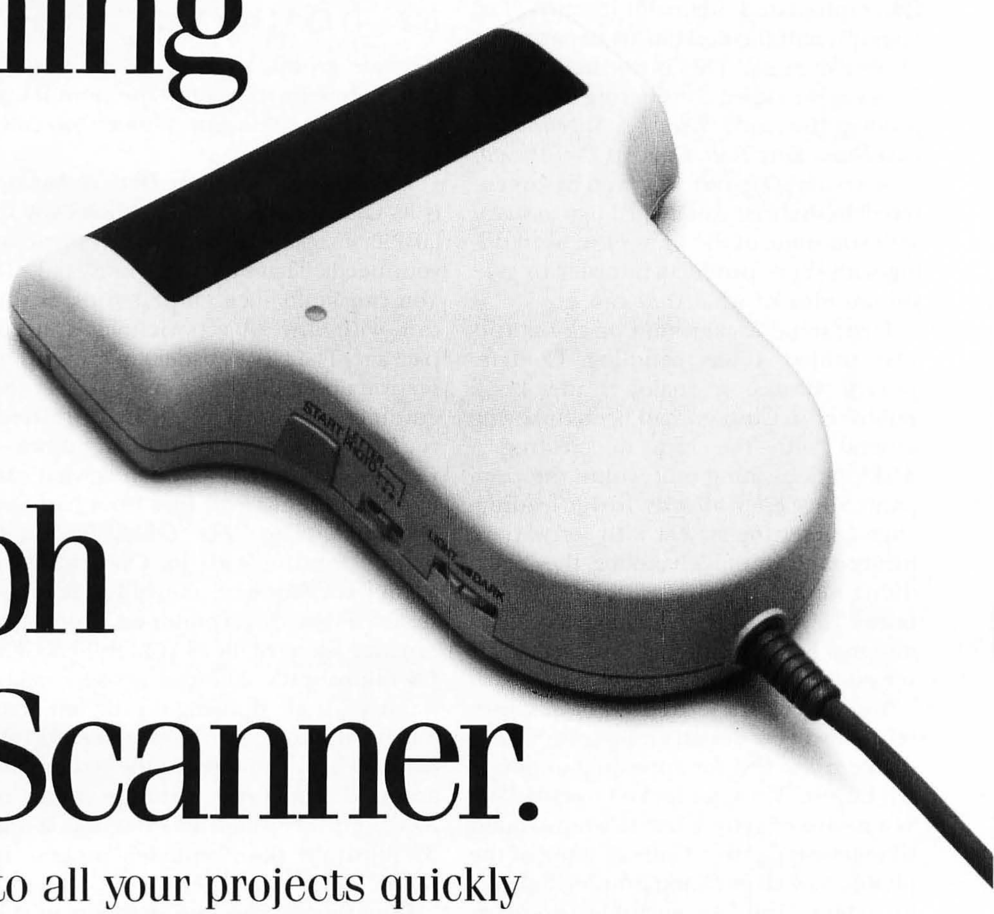
PROGRAM·LISTINGS

```
        Text 109,100,"U"
      Endif
    Next Clm%
    Wave 8,0
  Next T_saucer%
  Sprite A$
  Deftext 6,0,0,4
  Text 110,120,"By A. Baggetta"
  Pause 50
Return
'
Procedure Prt_values
  Deftext 0,0,0,4
  Restore Place_percent
  Deffill 2,2,8
  Prbox 13,95,302,128
  Color 0
  Rbox 14,96,301,127
  '
  ' Print the values under each picture
  '
  For Cd%=1 To 16
    Read Cp%,Rp%
    If Cdt%(Cd%)=>50 Then
      Cdt%(Cd%)=50
      Text Cp%-5,Rp%,Str$(Cdt%(Cd%)*2)
      Value%=Value%+Cdt%(Cd%)*2
      Inc Usable%
    Endif
    If Cdt%(Cd%)<50 Then
      Va$=Left$(Str$(Cdt%(Cd%)*2),2)
      Value%=Value%+Val(Va$)
      If Mid$(Va$,2,1)="." Then
        Text Cp%,Rp%,Left$(Va$,1)
      Else

        Text Cp%,Rp%,Left$(Va$,2)
      Endif
    Endif
  Next Cd%
Return
'
' Data
'
Place_percent:
Data 30,25,65,25,100,25,135,25,170,25,205,25,240,25,275,25
Data 30,93,65,93,100,93,135,93,170,93,205,93,240,93,275,93
'
Id_level_photo:
Data 20,30,20,30,50,55,55,30,55,30,85,55,90,30,90,30,120,55
Data 125,30,125,30,155,55,160,30,160,30,190,55,195,30,195,30,225,55
Data 230,30,230,30,260,55,265,30,265,30,295,55
'
Data 20,60,20,60,50,85,55,60,55,60,85,85,90,60,90,60,120,85
Data 125,60,125,60,155,85,160,60,160,60,190,85,195,60,195,60,225,85
Data 230,60,230,60,260,85,265,60,265,60,295,85
'
Photo_data:
Data 20,50,30,55,55,85,30,55,90,120,30,55,125,155,30,55
Data 160,190,30,55,195,225,30,55,230,260,30,55,265,290,30,55
Data 20,50,60,85,55,85,60,85,90,120,60,85,125,155,60,85
Data 160,190,60,85,195,225,60,85,230,260,60,85,265,295,60,85
'
Sprt:
Data &h0000,&h0000,&h0000,&h0180,&h07c0,&h0FF0,&h0000,&hFFFF
Data &h0000,&h1FF8,&h0FF0,&h03C0,&h0180,&h0000,&h0000,&h0000
Data &h0000,&h0000,&h0000,&h0180,&h07c0,&h0FF0,&h0000,&hFF9F
Data &h0000,&h1FF8,&h0FF0,&h03C0,&h0180,&h0000,&h0000,&h0000
'
' Circles
Data &h0000,&h0000,&h0000,&h0000,&h03C0,&h0420,&h0810,&h0810
Data &h0810,&h0810,&h0420,&h03C0,&h0000,&h0000,&h0000,&h0000
Data &h0000,&h0000,&h07E0,&h0810,&h1008,&h3004,&h3004,&h3004
Data &h3004,&h3004,&h3004,&h1008,&h0810,&h07E0,&h0000,&h0000
Data &h1FF8,&h2004,&h4002,&h8001,&h8001,&h8001,&h8001,&h8001
Data &h8001,&h8001,&h8001,&h8001,&h8001,&h4002,&h2004,&h1FF8
' Flashes
Data &h0100,&h0100,&h0100,&h0110,&h0120,&h0140,&h0180,&hFFFF
Data &h0300,&h0500,&h0900,&h1100,&h0100,&h0100,&h0100,&h0100
Data &h0001,&h4002,&h2004,&h1108,&h0910,&h0520,&h0340,&h1FF8
Data &h0180,&h0340,&h0520,&h0910,&h1108,&h4004,&h4002,&h8000
```

END

Increase your publishing power.



Migraph Hand Scanner.

Add scanned graphics to all your projects quickly and easily with the Migraph Hand Scanner.

The Hand Scanner provides a scanning window over 4″ wide with four scanning resolutions: 100, 200, and true 300 and 400 dots per inch. It also offers adjustable contrast, three dither settings for scanning halftones, and a special setting for line art.

Build your own library of images. Logos, photographs, books, illustrations . . . provide you with an unlimited supply of art.

Touch-Up,™ Migraph's high resolution image editor, lets you scan *and* edit your images. Powerful editing tools include crop, invert, resize, flip, cut and paste, multiple zoom levels, a full array of paint tools, and much more.

When your images are "pixel perfect", import them into your favorite ST publishing programs like Calamus, Fleet Street, PageStream, and Publisher ST. In addition, several save formats let you use your images on the PC and Mac.

The Migraph Hand Scanner and Touch-Up. Powerful tools for professional publishing.

See your dealer today for more details or call us direct. Dealer inquiries invited.

For all Atari ST and Mega systems with monochrome or color monitors, 1MB memory and 3.5″ DS disk drive.

MIGRAPH™ INC

200 S. 333rd St., Suite 220 Federal Way, WA 98003 (206) 838-4677 (800) 223-3729 *(Pacific Standard Time)*

CIRCLE # 103 ON READER'S SERVICE CARD

The Personal Publisher:

BY DONAVAN VICHA

Migraph's latest ads read, in part, "Let Migraph staff the desktop art department of your dreams." This is no small claim. It's no hype, either, considering that their product list reads *Touch Up, Supercharged Easy Draw, Easy Tools, DrawArt Professional, Scan Art* and *OSpooler*. Migraph has us covered! In this first column, I'd like to share with you some of the things I've been doing with these products in order to give you an idea of what they can do.

First, some background on the particular project I was handling. Contemporary Books, a major trade book publisher in Chicago, had been tinkering around with the idea of creating a desktop-publishing unit within the company. Since they already had a leading-edge typesetting system with software to incorporate graphics handling, they really didn't need DTP capabilities to handle books. Nevertheless, they contacted me to design a book on the old "Munsters" TV series.

You've seen these kinds of books; they rely heavily on design and photos rather than erudite text for appealing to potential buyers. Thus, we looked toward DTP as a means of getting text to wrap around silhouette-type treatments of some of the photos, as well as to add graphic borders to pages and to sprinkle doodads throughout the text.

I found the perfect page border ready-made in Migraph's *DrawArt Professional Package* (see illustrations). I loaded the HALLOWEEN.GEM file into *Easy Draw* (EZD) and began to break it down into its separate groups so as to delete most of the Halloweeny aspects—the pumpkins, ghosts, fence, and spiders (much too cutesy for the Munsters).

An added bonus of the *Draw* collection is its GEM pedigree, which allows you to take its images apart and modify them for your needs. That's the real beauty of EZD: You can build such graphic files, as you can with any object-oriented drawing package. The GEM header contains information about all of its pieces so that the graphic can be manipulated—resized, reproportioned and broken down—without loss of detail, which is what can happen to bitmap art files from painting programs. (The EZD ".GEM" header is mainly for use with its Outprint program.) For instance, the full-page snapshots (Figure 1, combined from two separate shots) of the HALLOWEEN.GEM file dramatically show that what looks like a fairly simple drawing on the left actually breaks down into a mass of parts (objects) when exploded (the command accessed under the Arrange menu of EZD). Further snapshots (Figures 2 and 3) illustrate the "explode" process in Zoom and 2x-size Views.

One thing I ran into early on in this project was the fact that Contemporary's art director insisted we would eventually send my DTP pages to a Linotronic service bureau for final, camera-ready printing. Thus, in my opinion, the whole project was doomed from the start because these services charge a minimum of $6 to $10 per page, plus as much as $2 per minute extra after a page has passed a certain time limit in the printing process. Add my modest per-page charge and the cost of doing the book would become prohibitive, considering the book's cover price had to be kept under $10. For instance, let's say the per-page production cost of this 192-page (projected length—usually an optimistic estimate at best) was $10, which in effect cuts me off from any remuneration for my efforts. That puts the cost of just getting camera-ready pages at close to $2,000. Add the actual printing and paper costs, and production of the book nears to between $12,000 and $22,000. You've gotta sell a lot of books at $8.95 to make back that cost!

I was aware that the Lino service would sink this book if I didn't find a way to keep document file size down. While the full-page web frame without the Halloween doodads looks like a fairly simple graphic, it took the size of a single *Pagestream* page with required text to around 167K! That's why I broke the web down further to a mere corner of the original, which I could duplicate and rotate for use in the opposite corner within *PageStream*. (I could then frame the whole thing with a slightly heavier box, which would print better—something unobtainable with the limited line weights provided by EZD.)

Another graphic treatment I worked out for the book was a drop-in cap with bats flying up from it (Figure 4). It started with a single bat (the mammal, not the

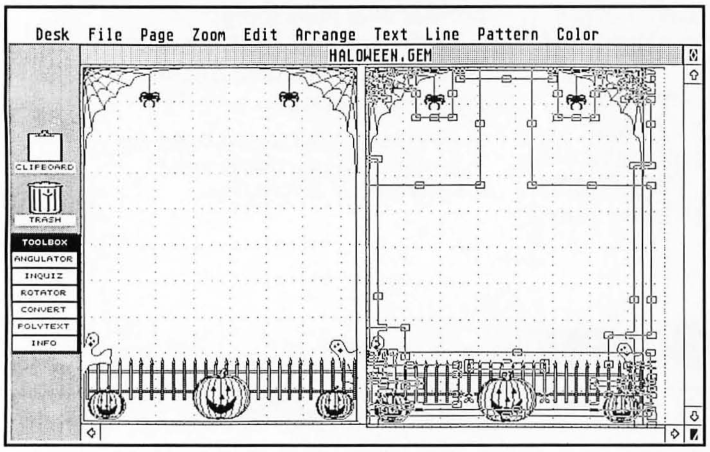


FIGURE 1

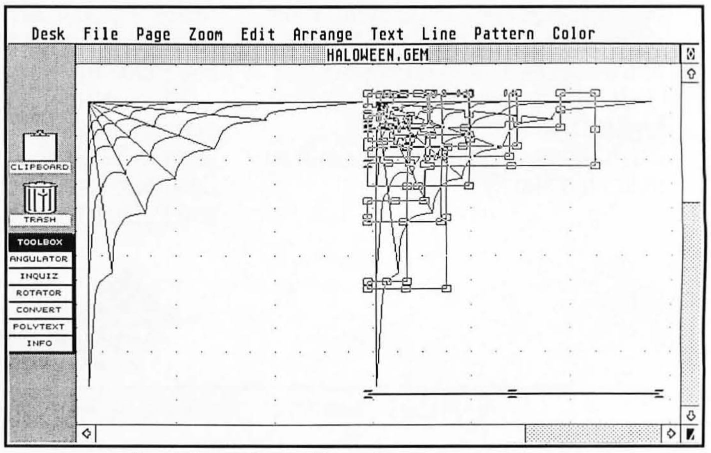


FIGURE 2

sports tool) from J. B. Brabson's clip art collection for XLent Software's *TypeSetter Elite.* I cleaned it up with the *Touch Up*'s Lightning mode's Fatbit editor, then duped it. From there, I used the skewing function to get a spreading-in-flight effect from a group of reproduced bats. In Text mode, I came up with a letter "W" in Schoolbook to match the text it would lead. Then I reduced the bats in Clip mode, placing them toward the left-front of the letter. This effect may be modified and used with a heading rather than drop caps; because there are close to 30 so-called chapters in the book, using the bats with a drop cap on each chapter would clearly be a case of overkill.

Touch Up is truly a boon to desktop publishers. To illustrate further *Touch Up*'s powers, let's look at a scan I did of Liz Taylor from a photocopy of a *National Enquirer* cover (Figures 5 and 6). To get her chin and neck using PictaScan, I had to start the scan where it cut off part of the top of her head, so the original image has a flat top and a lot of background black to it. Rather than use the square eraser in Lightning mode to remove the background, I used the spraypaint tool set at medium size and 82% saturation, turning the color and fill to white and black. This erases the background fine and leaves some texture at the edges of her hair. Stray pixels can be cleaned up using the process icon under the Drawing mode. This clean-up also took some detail away from her face.

I made a duplicate of the cleaned-up image so I had something to fall back on and compare with my working image. My first attempt to restore the top part of her head/hair was to readjust the spraybrush and spray it in; but it didn't look right. The lower-right side was especially lacking in detail, so I clipped part of the darker, lower-left hairline, flipped it to the right and fit it in where it looked good. I performed this using the transparent mode, since overwriting generally leaves gaps somewhere around the operation site.

To do the top of her head, I spraypainted away most of it, then used the lasso brush to capture the top part of the original. I included a part of the forehead where I'd lost some detail on my working image. I fit this part of her hair onto my working image, pasted it on, then cleared the lasso brush and lassoed a smaller part of the original in a shape to round out the flat top. When I got the proper size and shape, it blended right in.

To make the eyes more natural-looking—although part of the "charm" of this particular photo was the bug-eyed expression on Ms. Taylor's face—I went in first with the Fatbit editor, but pulled out when I'd botched the right eye. Using 2x mode, I then clipped out the offensive right eye, disposing of it with L-mode's eraser, and clipped a duplicate of the left eye, flipped it and carefully inserted it into the right socket. A little Fatbit clean-up and *voila*! *Touch Up*—what an apt name.

There's nothing exceptionally advanced in what I've done with these programs. Each one has such depth, you can get your creative juices flowing once you've learned the basic functions. Both work similarly in using the mouse buttons, with *Touch-Up* getting the nod for user-friendliness with its many icon-driven tools versus the menu approach of EZD. Then again, EZD has many more functions to work with as far as precision drawing goes, so it is more intimidating as well as essentially more powerful. Let's keep in mind that *Touch Up* is the front end of a set of scanners that Migraph will be releasing; its main market is for touching up digitized art, while EZD is aimed at graphic artists for creating original digitized art.

EZD has been around for quite a while and therefore has perhaps not attracted new ST/Mega users to its cause. It is quite unique, actually, not a true CAD program and not a flashy kilo-color paint program. But if you've been waiting for something more "state of the art" just for the sake of its newness, don't bother—it doesn't exist yet for the ST/Mega. (Although the publishers of *Calamus* have stirred a sensation with their forthcoming *Outline* program, which could be available by the time you read this.) With its extensions of the *Supercharger* and *Easy Tools*, EZD is an essential program for graphic artists and desktop publishers. For the Atari computers, EZD *is* state of the art.

As mentioned before, EZD is used to create graphics that can withstand resizing and distortion without any degrada-

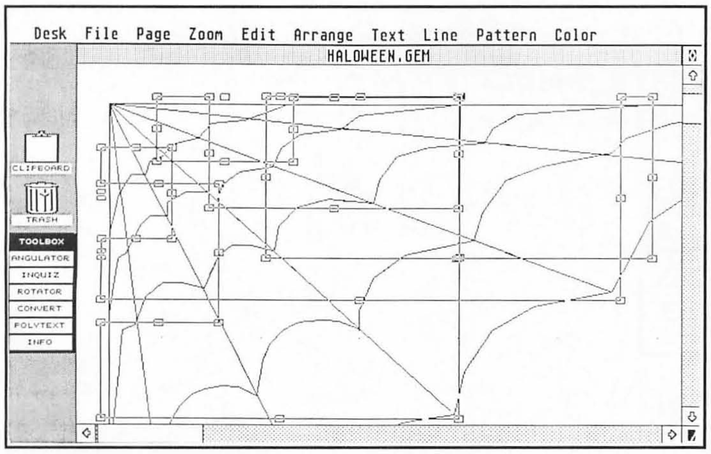


FIGURE 3

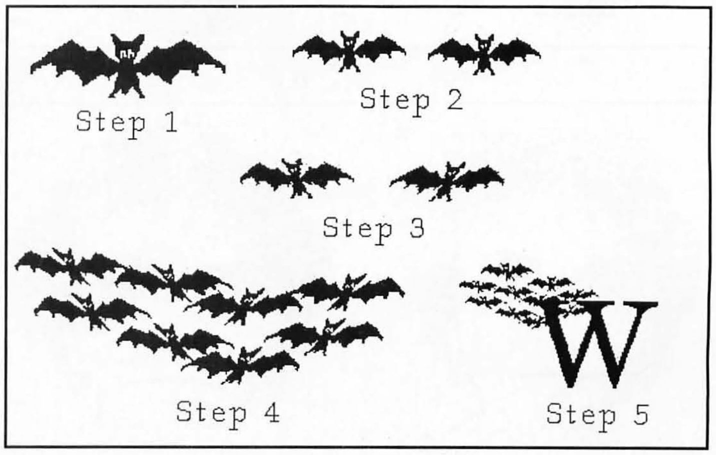


FIGURE 4

tion in the image. It does not have the tools commonly used by paint programs; you do not "freehand" draw, but put together pictures with various shapes, each overlaid on each other in such a way as to create a new combined image.

The example files that come with the package, as well as those found in the *DrawArt* package, truly show what this program is all about. For instance, there is a drawing of a house that can be broken down by clicking on the parts showing its facing side. Delete them, and what you see next is the interior of the house, which can also be deleted to show the other rooms and the other side of the structure. Your basic EZD "drawing" tools are boxes, ellipses (circles), arcs, lines and versatile polylines. *Supercharger* allows you to convert and import bitmap images as objects to be used for building images or for framing, or being framed by, other objects drawn with EZD. EZD can readily be used to draw various business graphics, architectural or engineering drawings and clip art for desktop-publishing purposes.

The *Easy Tools* component of EZD is an excellent extension of the program, which shows Migraph's commitment to keeping EZD on the cutting edge. It loads as an accessory (about 75K worth) and takes up a slot under the Desk menu. But it actually appears on the left-hand side of the EZD workscreen as a toolbox for accessing the following functions: Angulator, Inquisitor, Rotator, Converter and Polytext. When you click on one of these items, you get a button box for selecting further functions. For the Angulator, you produce a dialog box and a small ruler that can be used to measure the length and angle of figures or objects on-screen.

The Inquisitor dialog box allows you to check and change an image's coordinates and dimensions numerically, rather than by using the mouse. Depending on the kind of object being "quizzed," other attributes (such as the angle of rotation, wedge or arc) can be determined as well. Obviously, these first two tools provide greater precision for working with detailed or scaled drawings. The Inquisitor also allows you to customize your working (snap-to) grids to your precise needs.

The Rotator, as the name implies, allows you to define a point about which to rotate a polyline object, or rotate it about its center. You may also choose to make copies to be rotated about a center-point, thus giving you the ability to create spirals and toroids. This function alone is worth the price of the package. Keep in mind, however, that it cannot be used to rotate text or a bitmap object imported via the *Supercharger.*

The Converter allows you to create polylines, which work somewhat like the Bezier or B-spline curves in *Touch Up.* Choosing either Coarse or Fine, you can break an object's lines into segments with multiple "control" points, which can be edited using the Edit Polyline menu item. The example illustrated in this column (Figure 7) is a variation on one shown in the manual. I took a circle and reshaped it into a sunburst, with a control point pulled out from the surface of the circle to form a pair of rayed lines (an inverted V). The Converter is also a feature that by itself is worth the price of the package.

Finally, there is the Polytext function, which allows lines of polyline text to be rotated, resized and otherwise manipulated. Only one set of such fonts is included in the package, but they leave information in the manual for creating your own polyline fonts. This is a useful feature for creating captions that need to be rotated.

Thus, Migraph has kept their "old" program new. The company has now announced EZD 3.0, which incudes B-spline curves and the ability to lock bitmap images in place for tracing purposes. By tracing the bitmap or raster image, you create a vector or object-oriented image that can withstand resizing and other manipulations.

A variety of new printer drivers is also becoming available, including a driver for PostScript laser printers. These will be tied to Migraph's *OSpooler,* which, as the name implies, is a printer spooler for handling DTP-type printouts. It works as a desk accessory as well as in conjunction with the OUTPRINT.PRG components of the Migraph graphics programs. And it works with other GDOS-oriented programs, like *WordUp* and *Timeworks Desktop Publisher ST* (and the unreleased but promising *Wordflair*). It looks very flexible, and, of course, background printing of text and graphics pages is a welcome feature for any ST publisher. I just want to warn laser printer and 24-pin dot-matrix printer users that they will need the Migraph drivers, or, if they already have the drivers, may need to update them. This means laying out a little additional cash.

FIGURE 5

FIGURE 6

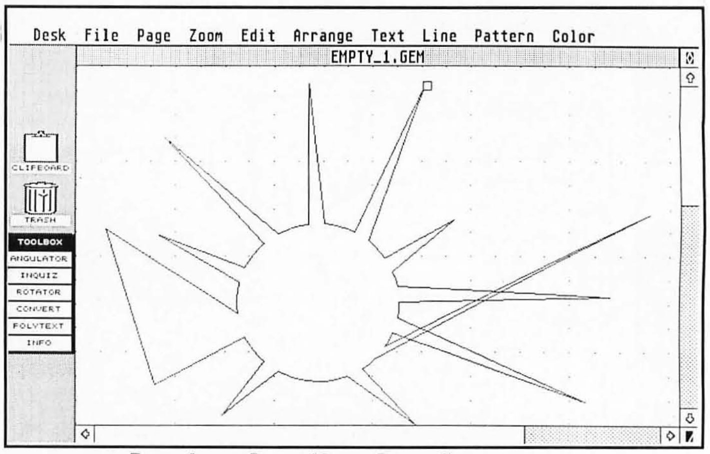


FIGURE 7

Clicking Off

While I didn't get to handle "The Munsters" book completely on my system, doing the design made me realize just what a complicated process such an assignment is. It was a situation aggravated by the vagaries of flaws in my software choices and my ignorance of a program's capabilities. These experiences do lead to frustration at times, but the knowledge gained is invaluable. For instance, I chose not to use *Desktop Publisher ST* because of its limited runaround abilities, and also because, with this program, I found my PostScript capabilities limited in the number of fonts available to me. Or so I thought. I recently had the opportunity to use a Mega 2 and NEC PostScript laser printer. I had installed *Desktop Publisher ST* on my meg-upgraded 520 ST for the PS laser and was only able to access two fonts. Since it was on my hard disk when I hooked up the Mega, I made no changes. Well, folks, little did I know that *Desktop Publisher*'s install program did not take my upgrade into account and short-changed me in the font department. I don't know how many PS fonts install on a 1040, but on the Mega 2 it fills all slots.

On another front, I was using a corrupted import module for GEM metafiles for *PageStream*, so I didn't know that with the proper import module I could have stripped down that HALLOWEEN.GEM file in *PageStream*, avoiding the need for EZD and using a less-than-satisfactory .IMG file in its stead. These instances do not bear out the old saying that "Ignorance is bliss." Rather, ignorance can cost you a contract!

If something big in Atari's brave new world of desktop publishing doesn't distract me, next time I'll continue what I started this month and discuss the steps involved in handling the production of a book. I'll also get into describing some ins and outs of *PageStream* (previewing the beta versions of *PageStream* leading to 1.6) in creating the design of *The Munsters*, which could be on sale even as you read this. ■

Donavan Vicha has been writing about desktop publishing on the Atari ST for three years. He uses a Mega ST2 system for his freelance editorial service, adapting 12 years of book-publishing experience to this growing field.

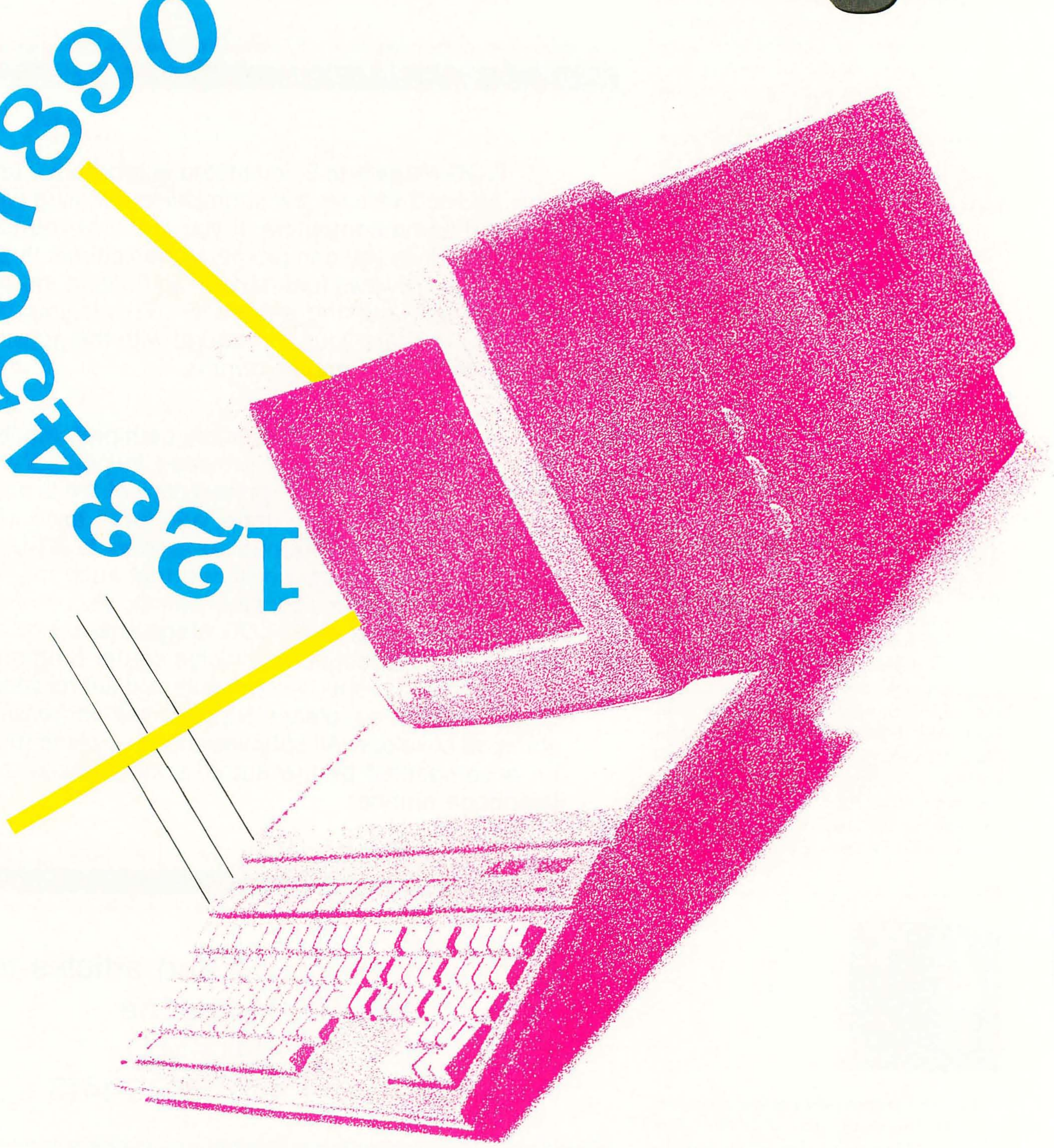
Attention Programmers!

ST-LOG Magazine is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing famiily of Atari computer owners.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. A typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ST-LOG Magazine**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ST-LOG Magazine**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send your programs and articles to:
ST-LOG Magazine
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

The Progra



mmer's Calculator

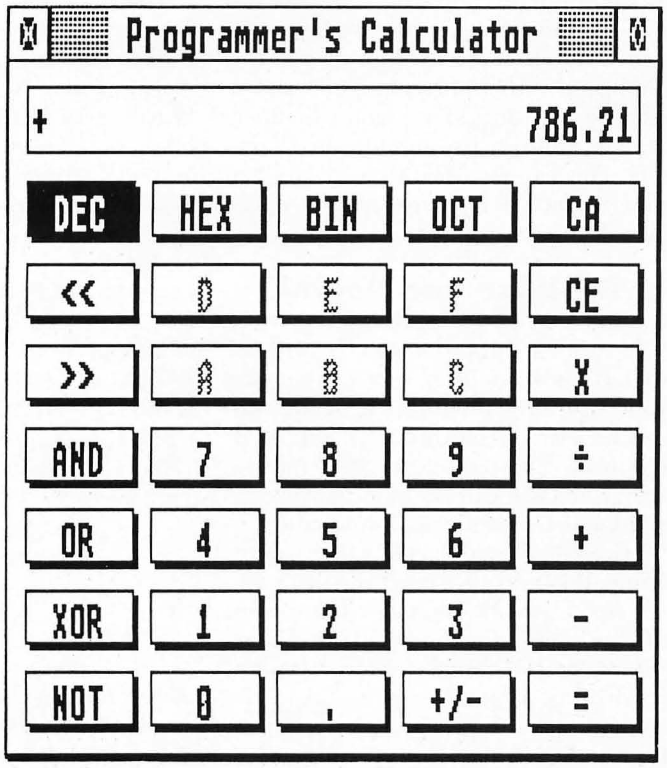
ROBERT M. BIRMINGHAM

When it comes to writing computer software, the old saying, "The better the tools, the easier the job" certainly rings true. When I'm programming, I often have to stop in the middle of whatever I'm doing and whip out the antiquated pencil and paper (remember those?), simply to solve some tiresome math problems. Not just your run-of-the-mill math problems, but countless conversions between number bases, ANDs, ORs, XORs and even (shudder) performing arithmetic operations with binary numbers. Having to do this day in and day out for several years is more than enough to push even the sanest person over the edge. Trust me, I know! Wouldn't it be nice if the computer would do all of this for you? I thought so. That's why I wrote the *Programmer's Calculator.*

The *Programmer's Calculator* is a GEM desk accessory that not only emulates a basic four-function calculator, but also takes the drudgery out of all the programmer's math you have to do. And perhaps most important, it saves paper!

To install the calculator, copy the file CALCULATOR.ACC to the same disk where you keep your other accessories; then reboot your computer. The *Programmer's Calculator* will now be accessible from the GEM desktop or from any program that supports drop-down menus.

NOTE: Because of this program's large size, it is available only on this month's disk version or from the databases of the ST-LOG ST Users' Group on DELPHI.



The Basics of Operation

To activate the *Programmer's Calculator,* select the "Calculator. . ." menu item from the desk drop-down menu. The calculator will appear in a window, which you can move, top it and close, just as you would do with most other GEM windows. To press the calculator's keys, position the mouse cursor over the desired key, then click the left mouse button.

Since you're probably already familiar with how real calculators work, I won't describe how the *Programmer's Calculator* works in any great detail. A brief explanation of a few of the more interesting features is probably enough.

As with some other calculators, the two keys labeled CA and CE represent the Clear All and Clear Entry keys, respectively. The CA key will completely reset the calculator, erasing any operation you were performing. The CA key is also used to clear any errors when they occur. The CE key lets you take back the last operation you performed. This can be handy if you entered " + " when you meant " – " or entered the wrong number in an equation.

Like all calculators, and even computers, there's a limit to the range of numbers the *Programmer's Calculator* can handle. For the decimal mode, the range is limited by the maximum of ten digits allowed for a number in this base (+/− 9,999,999,999). For the other number bases, the range is limited by the values that can fit into a 32-bit-long integer (−2,147,483,648 to +2,147,483,647.)

All the Bases Are Covered

One of the tasks often performed by the programmer is converting numbers from one base to another. The *Programmer's Calculator* supports the four most common number bases: decimal, hexadecimal, binary and octal. To convert a number from one base to another, select the base for the number you want to convert, enter your number, then select the number base you want to convert to. The converted number will appear in the display.

You may have noticed that when you switch to a different number base, some of the calculator's keys become enabled and some become disabled (when a key is disabled, you won't be able select it). This is because some keys apply to the current number base, while others don't. For example, the keys labeled "A" through "F" are enabled when you're in the hexadecimal number base, but are disabled for the decimal, binary and octal bases.

In addition to converting numbers, you may also perform arithmetic operations using any number base you like, just as you would with a normal calculator. For instance, you can add two binary numbers or even multiply two hexadecimal numbers. This can be a real time-saver, since most human beings are used to doing only decimal arithmetic.

Bit-Wise and Pound Foolish

To supplement the normal arithmetic operations of addition, subtraction, multiplication and division, there are also four bit-wise operations: AND, OR, XOR (eXclusive OR) and NOT. The bit-wise operations work in the same way as the arithmetic functions. So, if you wanted to OR two binary numbers together (for example, 1010 and 101), enter the first number, press the OR key, enter the second number, then press the equals key. The result will be 1111 (binary). Note that when performing a bit-wise operation in the decimal base, any fractional portion of the number in the display is discarded.

Shift to the Left, Shift to the Right!

Another common programmer's need is the ability to shift a number to the left or to the right, one bit at a time. For this, use the keys labeled "<<" (Shift Left) or ">>" (Shift Right). As with the bit-wise operations described above, if you're in the decimal mode, any fractional portion of the number in the display will be discarded before the shift is performed.

Clearing the Error

If you're in the decimal mode and you perform a calculation resulting in a number larger than the calculator can represent, an overflow error will occur. However, if you're in the hexadecimal, binary or octal modes, exceeding the maximum value the calculator can handle (a 32-bit integer) will cause the number to wrap around past zero. This really isn't an error, just something you should be aware of.

An error will also occur if you try to divide a number by zero. Unlike the overflow error, this error can occur regardless of the number base you're in. Dividing by zero is an undefined operation in mathematics. To clear either an overflow error or a division-by-zero error, press the CA key to reset the calculator.

Conclusion

The *Programmer's Calculator* was written entirely in Megamax C. The methods used to emulate a calculator might not be readily apparent, so the source code is heavily commented. This should help you if you're interested in studying the code. You might also find some of the individual routines interesting. Feel free to yank them out and use them in your own programs.

I hope you find the *Programmer's Calculator* useful. It can be a handy tool whether you program in C, Assembly or any other language. If you have any questions, comments or criticisms, feel free to contact me on CompuServe. My user ID is 73637,1120. Enjoy! ■

Robert Birmingham *is 26 years old and lives in Miami. When he's not programming he enjoys cycling, model rocketry and juggling.*

IAN'S QUEST

BY IAN CHADWICK

As I write this, I'm sitting outside on my deck, typing on my laptop computer, wishing it was a laptop ST. The Transportable is still a few months away, but it looks like a real product, not merely a promise. When it was first announced, I wasn't sure who would buy one: What applications on the ST make much sense for a laptop user? Then a musician friend pointed out his use: on-stage for MIDI control. Of course! A MIDI-laptop would be a welcome choice amid the growing clutter of the equipment required for the pro musician these days.

For me, the main computer functions are, in order: word processing, database management, spreadsheets, desktop publishing and telecommunications. After that falls everything else, including programming, games and graphics. Unless Atari improves the ST's mushy keyboard and makes the Transportable's keys industry standard-size, it's a bust for word processing, no matter what the size. Ditto for the next two applications. However, it might work well as a portable DTP station if I can import files from my PC (which I use for all typing-intensive activities). It would be even better if *Calamus* imported *WordPerfect* files directly, rather than forcing me to go through ASCII files as the intermediate stage.

Actually, a laptop ST would permit me to get back into programming on a larger scale. It would mean I could carry the computer with me and work in places presently restricted by the size of the ST, such as this deck. I wish someone would make a waterproof laptop I could use in my hot tub. . . .

I'm looking forward to the Portfolio, Atari's tiny PC-clone. To be fair, it's not really an Atari product (what is?); it's from Psion in the U.K. Atari simply bought the technology, an easier task than developing their own, I suppose. Anyway, the diminutive computer weighs in at about a pound (half a kilo for my Canadian friends). Powered by three "AA" batteries, it has 128K RAM, expandable to 640K. A *1-2-3*-compatible spreadsheet is also included.

Sounds good, but it has two drawbacks that make it less of a computer and more of an electronic notebook. First, the screen: It's only eight lines by 40 characters. That's useless for word processing. Second, the keyboard is too small and cramped for serious typing. I realize that it's a marvel of tiny technology to fit a computer into a box the size of a videocassette, but the design compromises push it outside certain practical uses.

This doesn't discourage me very much: I need an electronic notebook and had been considering the Sharp IQ calculator for that purpose. Compared with that, the Portfolio is truly several levels above. As long as it can communicate with my PC and has some sort of file transfer, load and save feature, plus reasonably good text-editing and database functions (the latter to store my phone book and contact list), then it will sell well and fill a niche. It appears the electronic notebook's time may have come at last, with the Portfolio as the first mature offering in that category.

• • •

One of the Forum discussions on DELPHI recently drifted into the Atari SLM804 laser printer. Against several of its strongest supporters, I had to defend my stand that the SLM804 is not an adequate professional design. True, it is fast—faster than several competitors, such as the Apple LaserWriter. But it has other drawbacks that weaken it. For example,

For me, the main computer functions are, in order: word processing, database management, spreadsheets, desktop publishing and telecommunications. After that falls everything else, including programming, games and graphics.

it doesn't allow for internally upgraded memory. This means that 1040 users are up the creek: The laser printer demands a Mega ST2 or Mega ST4. This major failing also prevents the SLM804 from being sold across system barriers: PC users, the largest computer base, can't use it. Of course, the printer's DMA connection, the reason behind the high print speed, also makes it unusable on other systems.

The Atari laser also lacks other features found on many laser printers: internal or cartridge-based fonts and emulation modes. The former can be rectified through software, but lacking the internal memory means that software fonts consume system RAM rather than printer RAM and reduce the available space for documents. As for emulation, this again is possible through software, but it's not as elegant or easy as a cartridge solution. Right now, the most popular laser printer is the Hewlett Packard LaserJet. I'd love to be able to port HP-compatible files over to my ST—such as *Ventura Publisher* print files—and print them on the SLM804. True, Ultrascript allows you to move Postscript files to the ST and then print them on the Atari laser, but again, it's a software solution and works outside existing DTP programs. It's much better to have a laser printer with expansion slots to allow a Postscript driver board inside.

For me, these drawbacks outweigh the advantages of the Atari laser, the chief one being the print speed. I never liked "closed architecture" systems, and I think they reduce salability in the long run. Certainly it doesn't help Atari to have its laser printer limited to the ST, a very small segment of the computer market. Consider that it even leaves Atari PC owners in the cold!

The Atari laser also is an odd duck; it isn't really compatible with anything else on the market. It isn't expandable or flexible, which restricts it to a small backwater while the rest of the computer world marches on.

...

Speaking of reasonable text editing (see the Portfolio notes, above), I just got a copy of *Tempus II* from Michtron. This is the upgrade of a text editor called *Tempus* from Creative Computer Design that was well received by programmers for its speed and features. The problem was that it didn't seem to get wide distribution. Now in the Michtron line, this excellent program will be available to everyone.

Tempus II is a GEM-like program that somehow seems to defy some of the accepted notions about GEM, particularly the cranky, slow scrolling windows we've all come to know and hate. *Tempus* literally screams, as far as scrolling goes. Written entirely in assembly language, *Tempus* is a fast, smooth program.

Tempus is a lot more than just fast. It has a wealth of good editing features that push it outside the realm of mere text editing into the domain of word processing. So fully featured is this version that it should easily become the editor of choice for the ST. It is certainly the most flexible: It has a wide range of keyboard shortcuts, and the user-definable installation options allow you to establish which keys execute what commands on a permanent basis.

Tempus permits sorting by text or by block, a rare option, but useful for creating lists. It has an internal calculator (RPN, which is an unfortunate choice for anyone not familiar with Forth or more accustomed to the standard scientific type) available through a pull-down menu. You can convert the entire 256 characters in the ST character set through a simple dialog box. The best use of this for programmers is to translate all lowercase input to uppercase, and vice versa. The conversion can affect the entire text or just a block. An unexpected but handy command allows you to quit *Tempus* and automatically launch another program immediately— for example, to jump right into a compiler.

Tempus does not have the printer-driver flexibility of most ST word processors. This is understandable because most GEM word processors produce graphic output to replicate the screen fonts, but it isn't necessary with *Tempus*. In other areas, especially the environmental control, it surpasses most word processors.

The only real flaw I found in *Tempus* is one shared by all other GEM ST word processors: It cannot perform a search-and-replace on a carriage return/line feed. For some this may be an ephemeral need,

Tempus II is a GEM-like program that somehow seems to defy some of the accepted notions about GEM, particularly the cranky, slow scrolling windows we've all come to know and hate. Tempus literally screams, as far as scrolling goes. Written entirely in assembly language, Tempus is a fast, smooth program.

but for me (and for porting ASCII files between systems), it's an absolute necessity. That's one main reason why I use *WordPerfect* or *ST Writer*. But I admit I appear to be among the minority of users who need this feature.

Of course, since it comes from Michtron, *Tempus II* is crippled by a mediocre manual and woefully inadequate index. Terms like "load file," "launch program" and "installation" are not even listed in the index, along with several hundred other functions and terms! The manual is confusing and opaque in places and is written entirely in the passive voice. Simply reading isn't enough to explain many features: You'll have to sit in front of the program and experiment in order to clarify several features. Maybe a good first learning project is to compile an index for the manual.

To be fair, I don't know if the abysmal manual is the fault of Creative Computer Designs or Michtron. Michtron may have simply bought or licensed the package as is and taken the manual intact. No matter: It is up to the publisher to ensure the competence and quality of the products to which it assigns its name. And if the manual needs rewriting, so be it. They should do it. But the number of really bad manuals that emerge from Michtron serves to convince me that they do not have the people or resources to identify or correct a bad product. For $79.95, the buyer deserves better.

I also think Michtron's well-deserved reputation for bad documentation does none of us any good and certainly doesn't make them appear a professional publisher.

Be that as it may, *Tempus II* is worth the effort. You can wade through the manual and figure out pretty much everything important in one session. It may take a few attempts to become accustomed to the program's commands and controls, but once you start using it, you may never want to return to another text editor again. And with the ability to launch directly into an external program, many programmers may never need to. There is a lot of power packed into a mere 70K. I recommend it.

Final Notes. . . .

A total non-sequitur, if I may: I recently acquired two ferrets as pets. I'm interested in hearing from any other ferret owners about their experiences with these delightful creatures, as well as suggestions for reading. ■

***Ian Chadwick** is a Canadian freelance writer with too many pets. He is also a volunteer worker at the Toronto Humane Society, an association similar to the ASPCA.*

DATEL COMPUTERS 800-782-9110 *ORDERS ONLY MON-SAT 8AM-6PM (PST)

CUST SERV / TECH 702-454-7700 TU-SA 10AM-6PM

CALL US FOR LOWEST PRICES ON OTHER SOFTWARE / HARDWARE

ST DEEP SCAN BURST NIBBLER - ONLY $49.99

Copy entire disk - even highly protected programs in under 60 seconds!!

*Works with one drive or two *Multiple copy option allows many backups from one original *Copy 1 or 2 sides up to 85 tracks and 11 sectors *Works on 520 or 1024 ST-color or mono *Special format parameters for non standard formats *Unique "INFO ANALYSER" displays vital disc parameters including sector distribution, dispersion, interleave, etc *Easy to use Icon driven program takes the mystery out of Disc back up.

ST SUPER TOOLKIT - ONLY 24.99 (mono monitor only)

**A powerful package of utilities for all ST models.*

*Read up to 85 tracks and 255 sectors! *Find/Replace function will scan for value and replace it automatically *Info mode displays vital disc information *3 different editor modes-Drive, Disk or File *Direct select function for Bootsector,Directory sector *Fully Menu/Icon driven HEX/ACSII edit modes - directly on the disk *Unique 2 disk compare function - scans both disks and reports on differences..perfect for the disk hacker! *Full printer support

3.5" EXTERNAL 1 MEG DRIVE - ONLY $179.99

*Slimline extra low profile unit - only 6" long *Top quality drive mechanism *Fully compatible - 1 meg unformatted capacity *Can be formatted to 85 tracks and 11 sectors *Complete with power supply and connecting cable - No more to buy

1 MEG INTERNAL DRIVE UPGRADE - ONLY $129.99

*Replace internal 500K drive with a full 1 Meg unit *Top quality drive Easily fitted *Full instructions *Direct plug in replacement *No special skills needed *Minor modification to enclosure is required *Remember,quality is most important

GENISCAN ST - ONLY$324.99

*An easy to handle Handy Scanner featuring 105 mm scanning width & 400dpi resolution enables you to reproduce graphics & text on your computer screen *Adjustable switches for brightness & contrast *Cut & paste editing of images *Save images as Degas, Neo, etc

VIDEO DIGITISER- ONLY $139.99

*On-board 32k of RAM allows real-time frame grabbing *Capture pictures into defined window*Multi-window display *Accepts input from camera or VCR *Cut and paste editing facility *Up to 10 screens can be stored at one time for animation, etc.

PRO SAMPLE STUDIO - $109.99

*A complete sound sampling package for the ST *100% machine code software for real-time functions *Saved files are compatible with other sampling systems *Sequencer for playback via keyboard or MIDI input.

EPROM BOARD - ONLY $24.99

*Complete Eprom Board & case *Accepts up to 4 X 27256 Eproms *Mapped into the cartridge area *Attractively styled case *High grade PCB * 64K or 128K blocks

TRACKMASTER - ONLY $ 59.99

*A must for the disk hacker *Plugs into disk drive port of the ST *Displays track number, disk side and read/write status as the head moves along the disk *Two-digit LED displays up to 85 tracks *Works on int. or ext. drives

ST TIMEKEEPER - ONLY $49.99

*Simple plug-in battery-backed calendar clock cartridge *Fits into cartridge port *On-board clock chip *Lithium battery backed lasts up to 5 years*Automatically adjusts for leap year etc *Exact time, date available on power up.

REPLACEMENT MOUSE - ONLY $44.99 FULLY COMPATIBLE

CALL US FOR THE BEST PRICES ON ALL YOUR SOFTWARE/HARDWARE NEEDS

DATEL COMPUTERS
3430 E. TROPICANA #67
LAS VEGAS, NV 89121

*ADD: $4.00 Shipping/ Handling in the Continental U.S. : $7.00- PR, VI, AK, HI, FPO, APO: $10.00- Canada / Mex: C.O.D. orders add $3.00 to above shipping charges

SPECIFY COMPUTER MODEL WITH ORDER VISA/MC / Checks / Money Orders / C.O.D. Accepted - Please call for return authorizaion number - Returns subject to 20% restocking fee.

CIRCLE #104 ON READER SERVICE CARD.

CAPITAL ST

by Bryan P. Schappel

Everyone dreams of owning it all, but no matter how hard we try, we just can't do it. You can't own everything. . .unless you're playing *Capital ST*.

Grab your checkbook, stuff your accountant into your briefcase and hit the road to success. In *Capital ST*, you move around the country, buying up as many properties as you can while steering clear of good ol' Mister IRS. Make sure you don't land on your competitors' property, or you'll have to spend some bucks.

ILLUSTRATION: BOB McMAHON

HIGH RESOLUTION

NOTE: *Due to this program's large size, it is available only on this month's disk or from the databases of the* ST-LOG *ST Users' Group on DELPHI.*

Running the Program

After you de-ARC the file CAPITAL.ARC, found on your disk (instructions are on the disk), you will have four files: CAPITAL.PRG (the program; run it from the desktop), CAP.RSC (the resource file), USAMAPHI.P13 (monochrome map of USA) and USAMAPLO.P11 (color map of the USA). All of these files must be kept in the same directory for *Capital ST* to run properly.

To run the game, just double-click on the CAPITAL.PRG icon on the desktop window. The program will load and run automatically. *Capital ST* runs only in low or high resolution.

About the Game

When the program first runs, you are presented with the intro screen; click on START to proceed. The CAPITAL PLAYER INFORMATION dialog then pops up, allowing you to select your play options. Clicking on one of the numbered buttons changes the number of players in the game; this may be set to between two and four players. Clicking on FAST or SLOW selects the game speed. As you will notice, there is a section of the dialog labeled PLAYER NAMES. This section contains an editable field for the name of each player. The names default to "Player N," where N is the player number. By typing in your actual names, the game becomes more personal. Click on ALL DONE to begin the game.

The Basics

Each player starts the game with 375G. (The "G" stands for "Grand." In *Capital ST*, all money is handled in denominations of $1,000.) In the lower-right corner of the screen you will notice tokens. There is one token for each player in the game; Player 1's token is a "1," Player 2's token is a "2," etc. These tokens will move around the path and indicate which property you are on.

Directly beneath the tokens you will notice a long white bar. This is where the name of the property that the token's on gets printed. You will also see some notices appear on this line; when a message appears on the line, the program pauses a little longer to let you read it.

In the lower-left corner of the screen you will notice a large white box labeled "Bank Accounts." This table shows how much cash the players have on hand. You can keep an eye on this to see who is going broke.

In the middle of the screen you will notice two large dice rolling. You may press the left mouse button to stop the dice from rolling and make a move. If you press the right mouse button, a dialog appears that will allow you to end the current game and either start over or exit to the desktop.

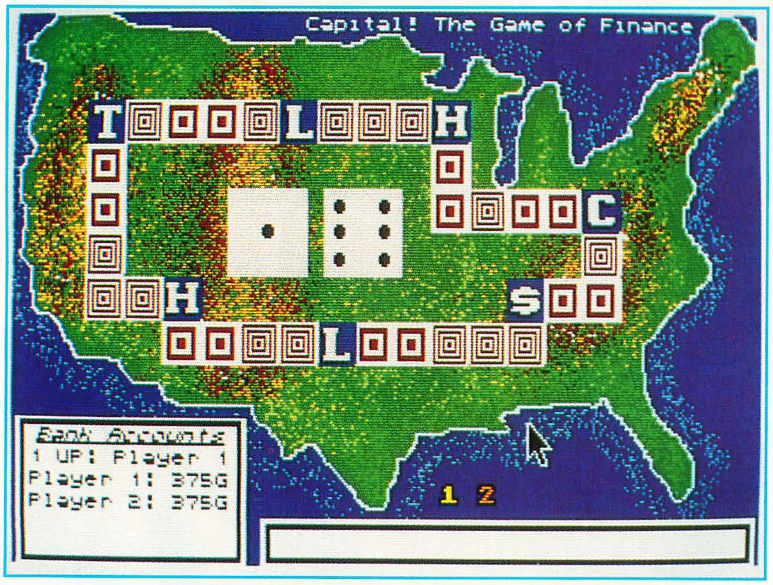
As mentioned earlier, *Capital ST* has two modes of play: *Fast* and *Slow*. When playing a fast game, if a player does not have enough money to pay taxes or pay another player's fee, that person is declared broke and sent to debtors' prison. In a slow game, the bankrupt player is forced to sell off enough of his properties to pay his debts.

The Map Screen

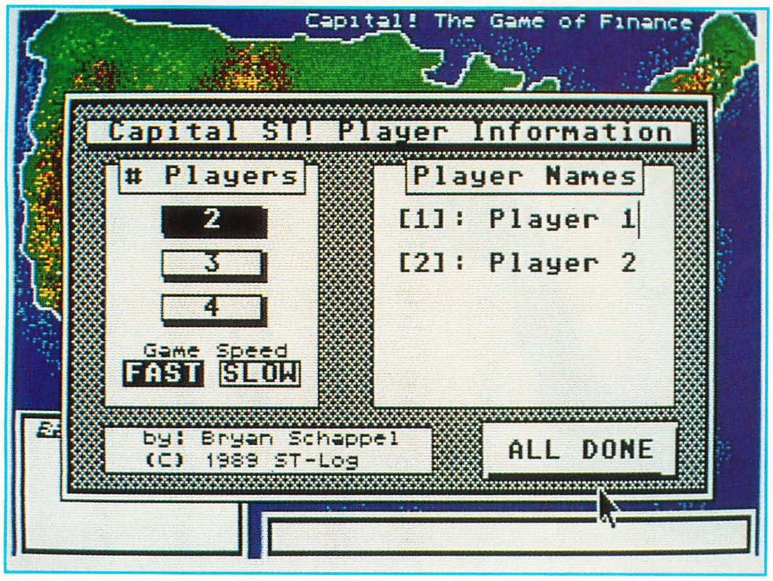
This screen contains a map of the United States. There is a path encircling the U.S. that is mostly made up of properties you may buy. There are two types of properties—Single and Double.

A Single property takes up one map square and you can buy the whole thing at once. Double properties are a little different. They are paired and take up two adjacent map squares. When you land on an unowned Double, you have the option to buy it (just like any other property). If you do buy the first half, you have the option of buying the second half, then and there, for twice the price. Owning both halves of a Double has advantages: When another player lands on either half of the double, he pays the sum of the fees.

You can tell which properties are Singles and which are Doubles by looking at the screen. Single properties are formed of thin concentric squares, and Double properties are formed of thick concentric squares. When a property gets bought, either an "X" is drawn through the property icon (in low resolution) or the word



Bank Accounts
1 UP: Player 1
Player 1: 375G
Player 2: 375G



Capital ST! Player Information
Players
2
3
4
Game Speed
FAST SLOW
Player Names
[1]: Player 1
[2]: Player 2
by: Bryan Schappel
(C) 1989 ST-Log
ALL DONE

"Sold" is printed on the property icon (in high resolution).

You will also notice other squares scattered on the path, which are not normal properties. Here is what those squares are and what they do.

The "L" square is "The Luck Square." There are two of these on the path. Each one can bring you good or bad luck. If you have good luck, you could win ten grand, 15 grand or a free improvement for one of your properties. If your luck is bad, you could lose ten grand, half of the value on a property or half of a property fee. These squares are always random, so you can't predict what will happen.

The "C" is "The Capital Gains Tax Square." This is like bad luck, only much worse. If you land here, you automatically lose 45G! (Ouch!)

The "$" is "The Stock Market." Every time you land on or pass this square, you get 12G in stock dividends. Not too bad.

The "H" is "Vacation Time At the Hogg Hilton." There are two of these squares on the path. Landing on either of these squares has no effect on your cash reserve. You do, however, have to take a holiday and may transact no business on this turn.

The "T" stands for "IRS Tax Audit." When you land on the Tax square, you will lose 12% of your current cash supply. This cannot cause you to go broke, as any player with seven grand or less pays no tax. (What a tax break!)

If you land on an unowned property, on Capital Gains Tax, or you are put in debt (in a slow game only), you are transferred to the *Capital ST* Transaction screen.

The Transaction Screen

It is through this screen that almost all business transactions take place. From this dialog you may buy, sell or improve a property. When you first enter the transaction screen, all the information about the property you landed on is displayed in the top half of the dialog. This information contains the property's name, owner, fee, full and market value, number of improvements and Double status (yes or no).

At the bottom of the dialog is a box labeled "Your Options." This box contains four buttons. If an option is not available, it will be shown as deselected (the text will be grayed out). The first time you enter the transaction screen, the only options you will have are BUY and EXIT. If you press BUY, you become the proud owner of the property you landed on. Pressing EXIT will take you back to the map screen, allowing the next player to move.

If you elected to buy the property, the SELL and IMPROVE buttons become active and you may perform these actions if you wish. If the property you landed on was a Double, the BUY button is renamed the DOUBLE button, and it allows you to purchase the other half of the property for twice the price.

Selling and Improving Properties

If you elect to sell or improve any of your properties, the buttons at the bottom of the screen are renamed to NEXT, BACK, SELECT and EXIT. The NEXT and BACK buttons allow you to browse through your property list; pressing SELECT, naturally, selects the property that is displayed; and pressing EXIT will abort the property-selection routine.

You are allowed to improve a property three times; each time an improvement is made, the value of the property is increased by five grand and the fee is doubled. Improving a property costs you ten grand. When you select the property to improve, make sure the improvement level is below three before you press the SELECT button. If you do not check this, it does not cost you any money, but you may not select another property to improve until your next turn.

When you opt to sell one of your properties, you do not get the full value of the property at the time of sale. You will receive a fair market price, which is determined when you enter the transaction screen. The market value will range between half and full value.

The only exception to this rule is during forced sales in a slow game. When you are forced to sell off your properties, you will only get half value for them—no exceptions! This is the penalty for improperly handling your cash supply. During a forced sale, the EXIT button becomes deselected.

You will notice that directly above the option box there is a small white box included to remind you of what you are doing. It is easy to be selecting a property and forget whether you are improving or selling.

Well, that's all you need to know about how to play *Capital ST*. Let's talk about the fun stuff.

Technical Notes

Capital ST was developed on a MEGA ST2 with the Laser C development system. It is my first program ever written in C. The player tokens were accomplished with the Line A sprite routines, and the dice and property squares are raster sprites displayed with the *vro_cpyfm* function. Great thanks to Clayton Walnum for all the support that made this game possible.

Getting the map screens into the ST was the hard part. The 8-bit version of *Capital* was programmed in ANTIC mode 4 (a character mode). This screen was then converted into a graphics 7.5 image that was dumped to my Panasonic printer. The map was then cut out of the paper and the resulting outline was taped to my ST monitor. The map was drawn freehand using *DEGAS Elite*. (That is why it is not 100% accurate. Don't teach geography class with it, okay?) The mountains were produced with the *DEGAS* block functions and then touched up considerably with the airbrush.

All the raster sprites were created with *Seurat* because of its incredible ZOOM mode. The zoom function plots a grid on the screen, allowing precision work within a predefined area. The sprites were then cut from the screen using the *Raster Sprite Editor* from ST-LOG #16.

I had a horrible time with "mouse button bleed through" between the AES and the VDI. Refer to my source code for the solution to this dilemma (see the ROLLDICE routine). The idea came from Charles F. Johnson's *Mouse-ka-mania* article, which can be found in ST-LOG #18.

All in all, this was an incredibly fun project and I will gladly write more programs for the ST (what a joy it is!). Until then, enjoy strutting your financial acumen! ■

***Bryan Schappel** lives with his wife in Madison, Wisconsin. In their free time they play* Dungeon Master *and watch Monty Python reruns and Warner Bros. cartoons.*

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing family of Atari computer owners.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send your programs and articles to:
ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

ST USER

by Arthur Leyenberger

Atari recently appointed a new User Group Coordinator, Chris Roberts. The post had previously been held by Sandy Austin, and then Cindy Claveran, and has been vacant for some time. In addition to being responsible for User Group coordination activities, Chris will oversee Atari's participation in all Atarifests and World of Atari shows.

For two years Chris was president of the Pasadena Area Local Atari Computer Enthusiasts (P.A.L.A.C.E.). He is an experienced BASIC programmer on both the 8-bit and ST computers, has run his own BBS and worked for the Federated Group. Chris is married, has three children and is a native Californian.

He has his work cut out for him. First, the User Group database is apparently out of date, so he asks that all User Groups reregister with Atari. To do so, send a card with your club name, official address, president's name and phone number and where and how often you meet. Also, a sample newsletter, if your club has one, would be helpful. This information should be sent to: Chris Roberts, User Group Coordinator, Atari Corp., 1196 Borregas Ave., Sunnyvale, CA 94086. In addition, Chris can be reached at (408) 745- 2052. If individual Atari users would like a listing of the registered User Groups in their state, they need only to write to Chris at the above address and include a self-addressed stamped envelope.

In order to foster better communications among User Groups, Atari and each other, Chris plans to revive the User Group Newsletter on a bimonthly basis. Further, Atari and GEnie are working to create a User Group area on that information service that will be available only to registered User Groups. The special area will allow User Groups to get the latest news and information from Atari, receive help with technical issues and stay in touch with each other.

It's encouraging that Atari has not only filled the User Group Coordinator position, but has done so with a person who has had hands-on experience with User Groups. One of the continuing problems that has plagued Atari users for many years is getting accurate information from Atari. Hopefully, Chris Roberts will be able to resolve this problem. I wish him much success.

CES Update

I've just returned from my 14th consecutive visit to the semiannual Consumer Electronics Show. As I have done twice yearly for the past seven years, I spent four days traversing the equivalent of 13 miles of aisles of some 1,300 audio, video, telephone, computer, video game and entertainment exhibits. In other words, I spent 32 hours roaming around two giant exhibit halls the size of about 17 combined football fields. Good thing I wore comfortable shoes!

Although not on display, several of the Atari folks were walking around showing the Portfolio, an under-$400 hand-held computer that weighs less than a pound and is about the size of a paperback novel.

Atari was there, but the focus (as in the past several shows) of their booth was games, games, games. Some twenty new game titles were announced for the 2600, 7800 and XE game systems. These new titles are scheduled to be available by the end of the year. Atari also announced new lower prices for the 2600 ($50) and the 7800 ($70). Further, they announced that light guns are now available for the 2600 and 7800 game machines.

To me, the silliest section of the Atari booth was the display of a full line of calculators. Why is Atari selling calculators? Atari is a lean company with limited resources. Selling calculators may not require all that much effort (the calculators are actually made by a third party that has paid for the right to use the Atari logo), but it sure dilutes the image of the company. Is Atari a computer company? Is it a game company? It's been difficult to figure this out for a long time now.

Still, there were just enough computers on hand to be confusing. In one outside corner of the booth, a lone ST, attached to a MIDI keyboard and running a MIDI program was being demonstrated. There were also a couple of IBM clones (shown previously) that have still not become available in the U.S.

Although not on display, several of the Atari folks were walking around showing the Portfolio, an under-$400 hand-held computer that weighs less than a pound and is about the size of a paperback novel. If you've read our previous coverage of this machine (when it was first shown at the Spring COMDEX), you'll recall that it is really a diminutive IBM clone since it contains DOS in ROM, has 128K bytes of RAM (expandable to 640K), an 8-line by 40-character display and a 63-key QWERTY keyboard.

The Portfolio can use either ROM cards for software or RAM cards for data storage. Two standard "AA" batteries power the Portfolio for up to 48 hours of continuous operation, and an interface jack is provided for exchanging data directly with a PC. Also, a text-editing program and an address/phone list ROM card are provided with the unit. According to Atari representatives, the Portfolio will be available by the time you read this.

Comparing the Atari portable game with the Nintendo unit shows that the Atari game is easier to see because of its use of color and backlighting. Further, the Atari screen is larger than that used on the Game Boy.

Another New Video Game

The main attraction at the Atari booth was the announcement of the Atari Portable Color Entertainment System. To quote Atari, "the world's first color portable hand-held video game system." In non-PR-babble, it's a hand-held game machine that uses a 3½-inch built-in color LCD screen, can display graphics with up to 16 simultaneous colors from a palette of 4,096 colors and has a resolution of 160 by 102 pixels. The portable video game uses a 16-MHz processor, which is faster than that used by other video game machines, such as the Nintendo and Sega.

The portable game weighs about a pound and is slightly larger than a videocassette. Although it is a completely self-contained unit, it can be linked with up to eight other units for multiplayer games. The game system has 64K RAM and runs on six "AA" batteries or can be powered by an included AC adapter.

The game has an eight-way "joypad," four fire buttons, two option buttons, a pause and an on/off switch. Other features include four-channel sound, volume and screen contrast controls and a headphone jack. One innovative aspect of the game is that the screen image can be rotated 180 degrees to accommodate both right- and left-handed players.

The Atari hand-held video game was developed by Epyx, which initially had planned to sell it under their own name. In fact, it was to have debuted at this show. The rights to the game were apparently sold to Atari shortly before the start of the show. Although Atari did not publicly acknowledge that Epyx designed and developed the hardware, they did announce that Epyx will be the first company to release games for the machine. These will be available on credit card-sized ROMS that slip into the unit and sell for under $35.

When the game is released, Epyx will have five games available: *Blue Lightning* (a first-person jetfighter game), *Time Quests and Treasure Chests* (an adventure/strategy game), *Gates of Zendocon* (arcadelike action game) *Impossible Mission* (action/adventure game) and *Monster Demolition* (action game). Each game card typically contains 256K bytes but is capable of holding as much as two megabytes of information. Epyx's *California Games* (an action game already familiar to many Atari computer owners) will be bundled with the system.

Atari said they are working closely with Epyx to develop more titles for the game and also hope to interest third-party developers in the system in order to produce more games. The retail price of the Atari Portable Color Entertainment System is $150 and it is scheduled to be available this fall, in time for the Christmas season.

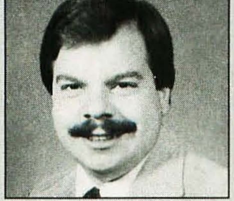
Competition

Nintendo also announced a hand-held video game machine at CES called the Game Boy. Already available in Japan, the Game Boy is of particular interest here since it will be competing with the new hand-held Atari model. The Game Boy is about half the size of the Atari unit and weighs about ten ounces. It operates on four 'AA' batteries, has stereo sound and a headphone jack. Unlike the Atari portable game machine, the Game Boy uses a monochrome non-backlit LCD screen. Two Game Boy units can be connected via a cable for two-person games, such as baseball.

The Nintendo Game Boy will retail for $90 and game cards will list for under $20. *Tetris* will be shipped with the unit, and other titles, such as *Super Mario Brothers*, will be available when the system is introduced this fall.

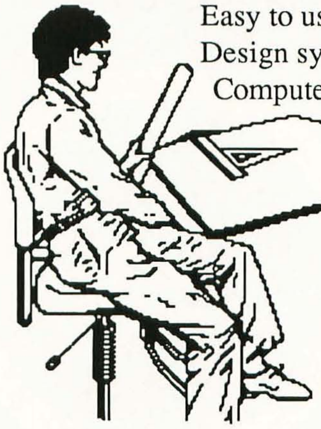
Comparing the Atari portable game with the Nintendo unit shows that the Atari game is easier to see because of its use of color and backlighting. Further, the Atari screen is larger than that used on the Game Boy. On the other hand, the smaller overall size of the Game Boy may make it more convenient to carry in a bookbag or a pocket.

Nintendo currently enjoys about three-quarters of the home video-game market. They will, no doubt, launch a major advertising campaign when the Game Boy becomes available. Given Atari's (non)advertising history and penny-pinching policies, it is anyone's guess what they will do to promote their portable game. As for me, I'm going to buy both. ■



Arthur Leyenberger is a freelance computer journalist who lives in, works in and loves New Jersey. He can be reached on CompuServe at 71266,46, or on DELPHI as ARTL.

ATHENA II



Easy to use Computer Aided Design system for the Atari ST Computer system.

Don't let the other companies fool you! You do not have to take a college course or wade through a three inch manual in order to use a CAD package! Within minutes you can be up and running with Athena II.
See what Athena II can do for you at your local Atari Dealer!!

Only **$99.95**

iliad Software Inc.

1470 South State, Orem, Utah 84058 (801) 226-3270

CIRCLE #105 ON READER SERVICE CARD.

ProCopy ST BACKUP UTILITY

You can't backup your software because copy protection locks you out. **ProCopy** is the key!

- Protects against the accidental loss of expensive software
- Works with all Atari STs
- Copies both S/S & D/S disks
- Use with 1 or 2 disk drives
- Not copy protected
- **FREE** shipping and handling
- **TOLL-FREE** telephone number
- Updates available to registered owners
- Orders shipped same day
- Disk analyze function included

Dealer Inquiries Welcome

VISA MasterCard and C.O.D. orders

Call (800) 843-1223 **$34.95**

Send check for $34.95 (overseas add $2.00 for air mail) to:

PROCO PRODUCTS

P.O. BOX 665, CHEPACHET, RHODE ISLAND 02814 USA
(401) 568-8459

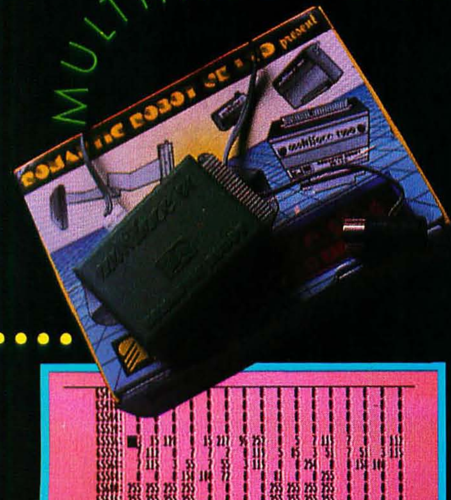
Available in Europe	THE MOVING FINGER CO. Building 2 Shamrock Quay Southampton, SO1-1QL England Tel. 0703-229041	TECH-SOFT COMPUTER WHOLESALE 324 Stirling Highway Claremont Western Australia 6010 Tel. 09-385-1885

CIRCLE #106 ON READER SERVICE CARD.

FROM OVER THE BIG WATER

By Marshal M. Rosenthal

MULTIFACE TOOLKIT



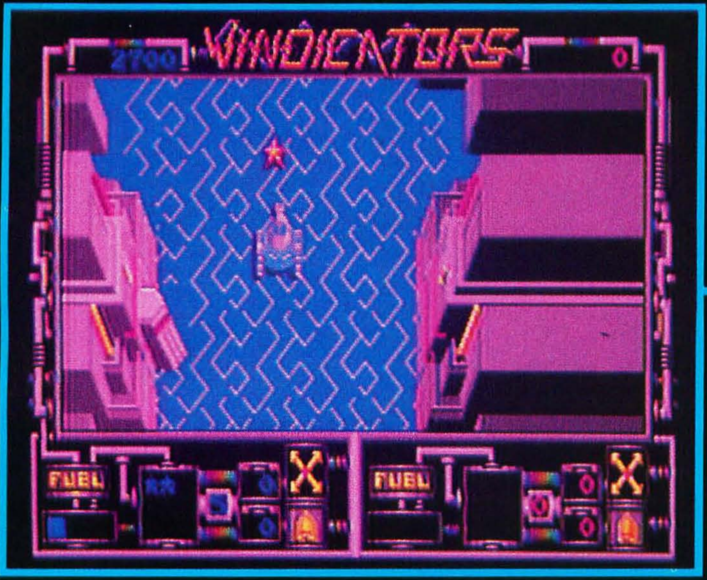
MULTIFACE

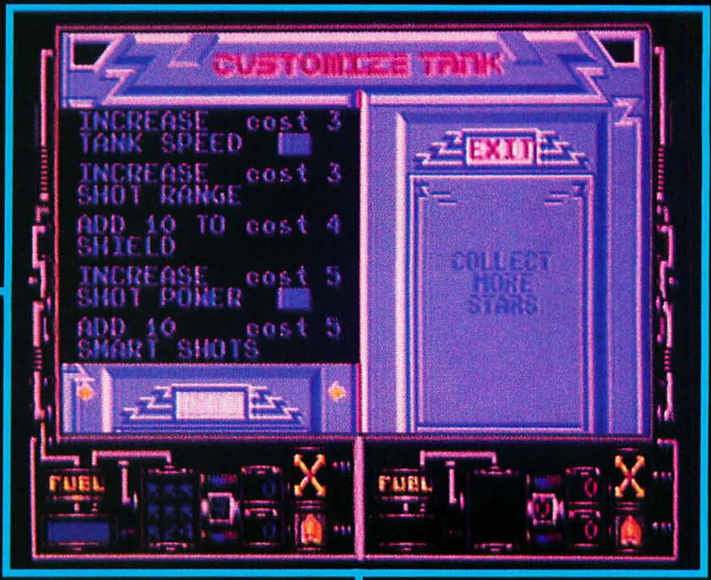
Welcome to England. I'm here to look into the ST scene, and the good news is that business is thriving. You can find ST software all over the place here, unlike New York, where it's scarcer than hens' teeth. My favorite place to buy software is Boots, a combination department/drug store that is like K-Mart and Woolworth's rolled into one. There just happens to be an Atari show going on in London, and we'll be looking into a few of the upcoming programs. Before we start, here are some tricks passed on to me for U.S. Gold's *Roadblaster*. If you type "LAVILLASTRANGIATO" while the car is on the starting line, pressing "F" refills the fuel tank, pressing "S" advances it to the next stage and pressing 1-4 adds special weapons. Now, on to those previews.

Altered Beast (Activision U.K.) will be growling its way to the ST in a bit. This arcade translation requires you to walk softly and carry a big fist—a fist that becomes massive as you change from a wimp into a Supertough Beast. Meanwhile, there are all sorts of ugly things trying to take you out while the landscape scrolls merrily from left to right. The fun part of whomping these jokers is watching the "glow" from the sheer power of your blow when you punch or kick them, and seeing them fall apart or disintegrate.

Horror Soft's *Personal Nightmare* may prove challenging, or perhaps a bit silly. I mean, you find Elvira on the advertising! Nothing to see yet, but it's being billed as a new kind of icon-driven game that purports to have incredible sound effects and superb, fully animated graphics, all brought together by a highly sophisticated user interface. (Who writes this stuff?) In all seriousness, though, *Personal Nightmare* could prove interesting, so keep it in mind. More on this when there's more to tell.

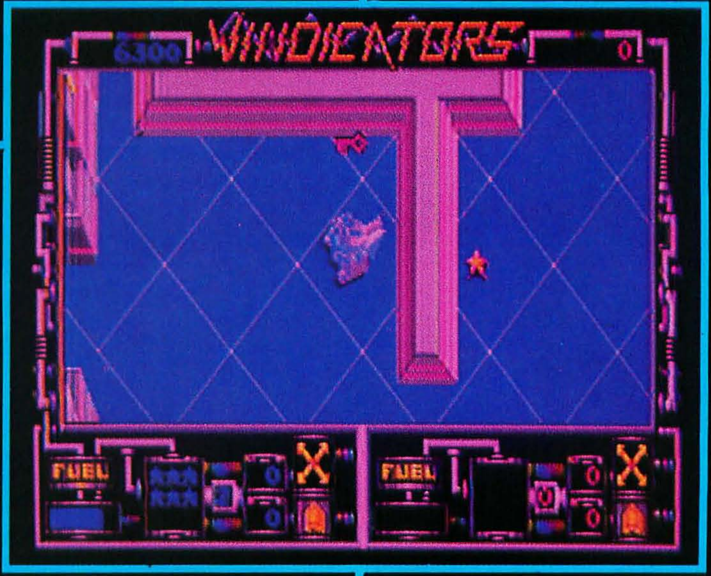
Xenon was a well-liked game, so how can *Xenon II: Megablast* fail? Again we find the

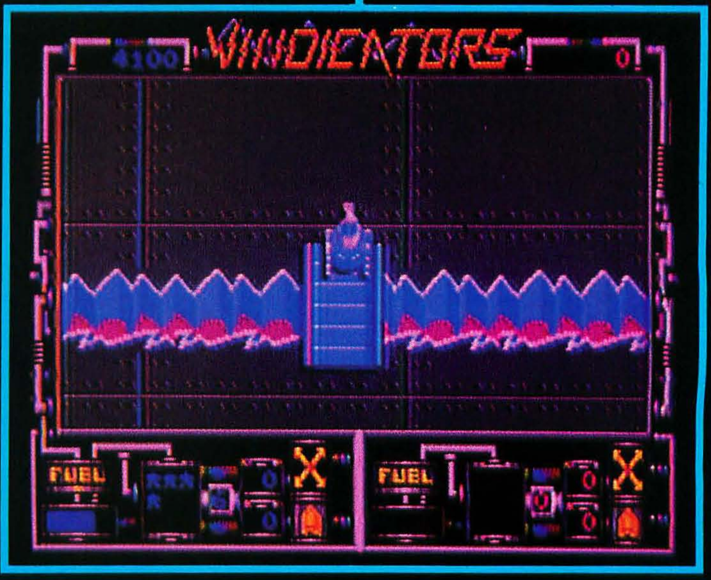




INCREASE cost 3
TANK SPEED
INCREASE cost 3
SHOT RANGE
ADD 10 TO cost 4
SHIELD
INCREASE cost 5
SHOT POWER
ADD 10 cost 5
SMART SHOTS

VINDICATORS

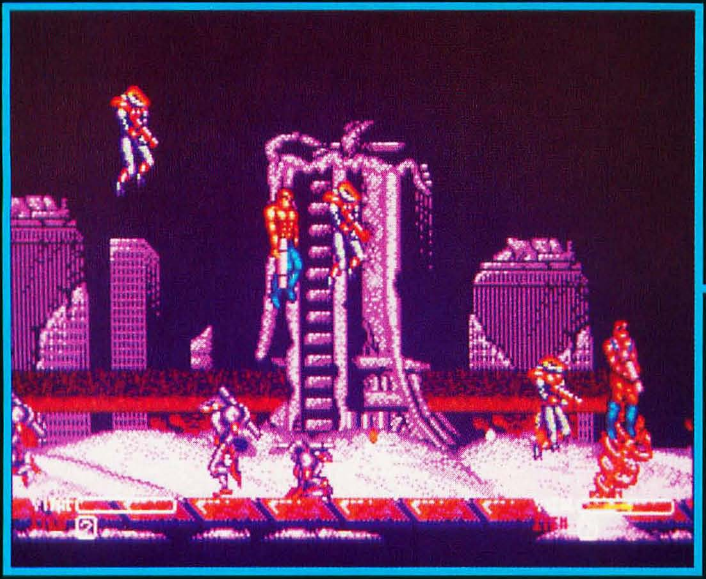
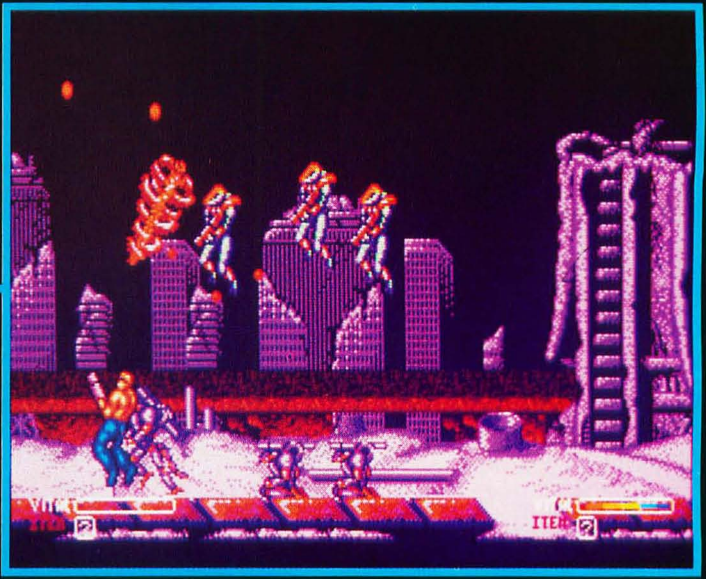




Xenites out to destroy Earth (why does every Tom, Dick and Thingo want our planet?) by shooting off Timebombs into our past. Your mission is to go into each level of the past—which means evolutionary levels up into the space age—and defuse those bombs. Of course, there are all kinds of obstacles, like reptiles, insects and the like. Huge sprites and three-layer parallax scrolling could make this one a big hit.

Domark is best known for James Bond games, so it's a nice change of pace to see *Vindicator* (developed along with Tengen games). Based on the Atari game co-op, you are required to drive a Super Tank through the merciless Starbase.controlled by the evil Tangent Empire,and blow the place up.

There are a few difficulties, like other tanks out to smash you to bits and rotating turrets tossing flak. Keeping on the move and taking offensive action (i.e. blasting everything that moves or doesn't move) is the best policy here as you scroll about, so make speed a priority for your tank. There will be opportunities along the way to run over Fuel and Keys (obvi-

FORGOTTEN WORLDS

Digitized graphics and excellent sound effects enhance what is basically a well-worn theme. Joystick control is a bit strange to get used to at first, but then becomes very intuitive.

ous what they're for), plus Stars. The Stars let you purchase added firepower and other extras in the "Shop" between Starbases.

Unfortunately, sampled music at the start can't make up for the incredibly poor quality of the sound effects. The visuals are a great treat though, with big explosions to accompany your blasts and lots of animation accompanying tank and turret movement.

Should being inside a metal shell seem cowardly, feel free to take on the entire eight worlds laid barren by the evil BIOS. *Forgotten Worlds* (U.S. Gold) requires that you, and a companion if desired, wipe out all of the "caretakers" that BIOS has left on each planet so that the people of the galaxy can get on with business as usual. It sounds like a good deal to you, since who would expect much smarts out of a character that looks like a punker who kissed a high-tension wire? Anyway, this means flying in all directions, shooting in all directions and generally having a great time going after everything that moves. Each world ends with a big baddie to whack. Hopefully, by then you'll have acquired something more powerful than a piddling laser gun.

Which brings us to the reward given for wiping out evil: Money, which can be used to buy that bigger gun or extra life. Of course, it just hangs there waiting to be taken while a zillion meanies are trying to demonstrate what you'd look like from the inside of a working microwave oven.

Digitized graphics and excellent sound effects enhance what is basically a well-worn theme. Joystick control is a bit strange to get used to at first, but then becomes very intuitive. There really isn't much time to think here.

Some products never make it to our shores, but Romantic Robot's *Multiface ST* may be taken off the shelves entirely. This device is a personal copier that "freezes" games/programs, then allows them to be saved to disk. There are many advantages to this, in addition to some obvious problems. European software houses take piracy seriously, as there's so much of it, and there's pressure to remove from circulation products that encourage piracy. Still, *Multiface* has some good points, so a closer look is in order.

First, the hardware consists of a cartridge that plugs into the ST, unlike a U.S. device called Switchback, which works through the printer port (you also run a cable between the RGB port to the monitor). Having all its needs in ROM means that there's no need to load any software. Pressing a button on the cartridge stops a program flat and then brings up a menu-driven command screen (even includes on-screen instructions) that saves the data out to disk. This could be a great way to save graphic screens for use in paint programs (of course, Romantic has a disclaimer stating that not all programs will work). This data can also be altered with the *Toolkit* program, included. Clever modifications with the *Toolkit* can make games easier to play since, after it has been "tweaked," the data can be loaded back into the ST, and the program continued from where it was saved. Compressing techniques work with a format that adds more space to a disk as well. To prevent improper use, *Multiface* requires that the cartridge be attached when running a saved program.

An added goody is an optional *Disk Or-*

ganizer program, which uses RAM for copying, meaning fewer disk swaps for single-drive users. Mass copying and deletions may be executed, plus disks can be compacted for faster loading.

The maual is quite readable (though a bit cheesy in assembly) and goes into extra functions, such as when to use the ST reset button if the "Magic" button fails, and how pressing F1 can also re-enable a malfunctioning joystick.

We'll continue with previews next month. Until then, 1) look both ways before crossing the street, and 2) remember (which I didn't) that English drivers drive on the *left* side of the road. ▪

Products mentioned:

• • • **ALTERED BEAST**
Activision U.K. Ltd.
Blake House
Manor Farm Road
Reading, Berks
England RG2 OJN

FORGOTTEN WORLDS
U.S. Gold
Unit 2&3 Holford Way
Holford, Birmingham
England B6 7AX

MULTIFACE ST
Romantic Robot
54 Deanscroft Avenue
London, England NW9 8EN

PERSONAL NIGHTMARE
Horror Soft
Unit 3, Addison Industrial Estate
Blaydon, Tyne & Wear
England NE21 4TE

VINDICATORS
Domark
Ferry House, Lacy Road
London, England SW15 1PR

XENON II: MEGABLAST
The Bitmap Brothers
London

***Marshal M. Rosenthal** has been a photographer and writer in the overseas market since the early days of the Atari 2600. His photographic/written features and pictorials can be found in major computer- and entertainment-related publications throughout England, France, Germany, Sweden, Mexico and the U.S.*

Attention Programmers!

ST-LOG Magazine is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing famiily of Atari computer owners.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. A typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ST-LOG Magazine**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ST-LOG Magazine**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send your programs and articles to:
ST-LOG Magazine
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

C·MANSHIP
Listing 1:C

```
do_menu ()
{
   int button;

   switch ( msg_buf[3] ) {

      case DESK:
         form_alert(1,"[0][  MicroCheck ST  |by Clayton Walnum  | | Copy\
right 1989   |    by ST-LOG  ][CONTINUE]");
         break;

      case FILEBAR:
         switch ( msg_buf[4] ) {

            case NEWACCNT:
               do_newacct ();
               break;

            case OPENMBR:
               button = get_acct ();
               if ( button )
                  open_acct ( filename );
               break;

            case CLOSEMBR:
               do_wind_close ();
               break;

            case NEWMNTH:
               do_new_mnth ();
               break;

            case QUIT:
               do_quit ();
               break;
         }
         break;

      case CHECKS:
         switch ( msg_buf[4] ) {

            case ENTER:
               do_enter ();
               break;

            case SEARCH:
               do_search ();
               break;

            case CHKCAN:
               do_check_canc ();
               break;

            case RECONCIL:
               do_reconcil ();
               break;

            case CHKAUTO:
               do_auto ();
               break;
         }
         break;

      case PRINT:
         switch ( msg_buf[4] ) {

            case PRNTWIND:
               print_wind ();
               break;

            case PRNTREG:
               print_reg ();
               break;
         }

      case UTILITY:
         switch ( msg_buf[4] ) {

            case NEWYEAR:
               do_new_year ();
               break;

            case NEWDATE:
               get_new_date ();
               break;
               case IMPORT:
                  do_import ();
                  break;

            }
            break;
      }
      menu_tnormal ( menu_addr, msg_buf[3], TRUE );
   }


   do_wind_close ()
   {
      int button;
      GRECT r;

      wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
      wind_update ( BEG_UPDATE );
      if ( search ) {
         search = FALSE;
         cur_top = edit_top;
         cur_count = num_trans;
         cur_chk_strc = checks;
         srch_trans = 0;
         set_menu_entries ();
         strcpy ( windname, acct_name );
         strcpy ( &windname[strlen(windname)], ": Edit mode" );
         wind_set ( w_h2, WF_NAME, windname, 0, 0 );
```

```
        draw_interior ( r );
    }
    else if ( canceling ) {
        if ( !saved )
            save_month ( monthfile );
        canceling = FALSE;
        strcpy ( windname, acct_name );
        strcpy ( &windname[strlen(windname)], ": Edit mode" );
        wind_set ( w_h2, WF_NAME, windname, 0, 0 );
        draw_interior ( r );
        set_menu_entries ();
    }
    else if ( loaded ) {
        button = form_alert ( 1, "[2][Do you want to close|this account?]\
[YES|NO]");
        if ( button == YES ) {
            do_save ();
            draw_rec ( r, 2, 8, WHITE );
            set_menu_entries ();
            wind_set ( w_h2, WF_NAME, noacct, 0, 0 );
        }
    }
    wind_update ( END_UPDATE );
}


handle_button ()
{
    wind_get ( w_h2, WF_WORKXYWH, &wrkx, &wrky, &wrkw, &wrkh );
    if ( mouse_y > wrky && mouse_y < wrky + cur_count * charh + 4
            && mouse_y < wrky + wrkh && mouse_x > wrkx
            && mouse_x < wrkx+wrkw && num_clicks == 1 )
        if ( !search && !canceling )
            edit ();
        else if ( canceling )
            canc_chk ();
}


do_arrow ()
{
    switch ( msg_buf[4] ) {

    case WA_UPPAGE:
        do_uppage ();
        break;

    case WA_DNPAGE:
        do_dnpage ();
        break;

    case WA_UPLINE:
        do_upline ();
        break;

    case WA_DNLINE:
        do_dnline ();
        break;

    case WA_LFPAGE:
    case WA_RTPAGE:
        do_hslide ();
        break;

    }
}


do_vslide ()
{
    GRECT r;
    int lines_avail;

    wind_get ( w_h2, WF_VSLIDE, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
    if ( r.g_x != msg_buf[4] ) {
        wind_update ( BEG_UPDATE );
        wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
        lines_avail = r.g_h / charh;
        cur_top = (long)msg_buf[4]*((long)cur_count-(long)lines_avail)/1000L;
        wind_set ( w_h2, WF_VSLIDE, msg_buf[4], 0, 0, 0 );
        draw_interior ( r );
        wind_update ( END_UPDATE );
    }
}


do_hslide ()
{
    GRECT r;

    wind_get ( w_h2, WF_HSLIDE, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
    if ( r.g_x != msg_buf[4] ) {
        wind_update ( BEG_UPDATE );
        wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
        left = !left;
        if ( left ) {
            wind_set ( w_h2, WF_INFO, infotext, 0, 0 );
            wind_set ( w_h2, WF_HSLIDE, 0, 0, 0, 0 );
        }
        else {
            wind_set ( w_h2, WF_INFO, &infotext[20], 0, 0 );
            wind_set ( w_h2, WF_HSLIDE, 166, 0, 0, 0 );
        }
        draw_interior ( r );
        wind_update ( END_UPDATE );
    }
}


do_uppage ()
{
     GRECT r;
     int lines_avail;
    wind_update ( BEG_UPDATE );
     wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
     lines_avail = r.g_h / charh;
     cur_top -= lines_avail;
     if ( cur_top < 0 )
        cur_top = 0;
    wind_update ( END_UPDATE );
     draw_interior ( r );
}


do_dnpage ()
{
    GRECT r;
    int lines_avail;

    wind_update ( BEG_UPDATE );
    wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
    lines_avail = r.g_h / charh;
    cur_top += lines_avail;
    if ( cur_top > cur_count - lines_avail )
       cur_top = cur_count - lines_avail;
    draw_interior ( r );
    wind_update ( END_UPDATE );
}
do_upline ()
{
   MFDB s, d;
   GRECT r;
   int pxy[8];

   if ( cur_top != 0 ) {
      wind_update ( BEG_UPDATE );
      cur_top -= 1;
      wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
      set_clip ( TRUE, r );
      graf_mouse ( M_OFF, 0L );
      s.fd_addr = 0L;
      d.fd_addr = 0L;
      pxy[0] = r.g_x;
      pxy[1] = r.g_y + 2;
      pxy[2] = r.g_x + r.g_w;
      pxy[3] = r.g_y + r.g_h - charh - 2;
      pxy[4] = r.g_x;
      pxy[5] = r.g_y + charh + 2;
      pxy[6] = r.g_x + r.g_w;
      pxy[7] = r.g_y + r.g_h - 2;
      vro_cpyfm ( handle, S_ONLY, pxy, &s, &d );
      prnt_chk_wnd ( cur_top, r.g_y + charh );
      set_clip ( FALSE, r );
      calc_vslid ( cur_count );
      wind_update ( END_UPDATE );
      graf_mouse ( M_ON, 0L );
   }
}


do_dnline ()
{
   MFDB s, d;
   GRECT r;
   int pxy[8];
   int lines_avail, index;

   wind_get ( w_h2, WF_WORKXYWH, &r.g_x, &r.g_y, &r.g_w, &r.g_h );
   lines_avail = r.g_h / charh;
   if ( (cur_top + lines_avail) < cur_count ) {
      wind_update ( BEG_UPDATE );
      cur_top += 1;
      index = cur_top + lines_avail - 1;
      set_clip ( TRUE, r );
      graf_mouse ( M_OFF, 0L );
      s.fd_addr = 0L;
      d.fd_addr = 0L;
      pxy[0] = r.g_x;
      pxy[1] = r.g_y + charh + 3 - 1*(res==MED);
      pxy[2] = r.g_x + r.g_w;
      pxy[3] = r.g_y + r.g_h - 1;
      pxy[4] = r.g_x;
      pxy[5] = r.g_y + 3 -1*(res==MED);
      pxy[6] = r.g_x + r.g_w;
      pxy[7] = r.g_y + r.g_h - charh - 1;
      vro_cpyfm ( handle, S_ONLY, pxy, &s, &d );
      prnt_chk_wnd ( index, r.g_y + lines_avail * charh );
      set_clip ( FALSE, r );
      calc_vslid ( cur_count );
      wind_update ( END_UPDATE );
      graf_mouse ( M_ON, 0L );
   }
}


do_quit ()
{
   int button;

   button = form_alert(1,"[2][Are you sure you|want to quit?][YES|NO]");
   if ( button == YES ) {
      search = FALSE;
      all_done = TRUE;
      if ( !saved )
         do_save ();
   }
}

save_month()
{}

do_new_year()
{}
```

```
do_new_mnth()
{}

do_newacct()
{}

get_acct()
{}

open_acct()
{}

get_new_date()
{}

do_save()
{}

do_import()
{}

do_reconcil()
{}

canc_chk()
{}

do_check_canc()
{}

print_reg()
{}

do_search()
{}

print_wind()
{}

do_auto()
{}

do_enter()
{}

edit()
{}
```

C-manship END

MAKE THE DELPHI CONNECTION

As a reader of ST-LOG, you are entitled to take advantage of a special DELPHI membership offer. For only $19.95 plus shipping and handling ($30 off the standard membership price), you will receive a lifetime subscription to DELPHI, a copy of the 500-page *DELPHI: The Official Guide* by Michael A. Banks and a credit equal to one free evening hour at standard connect rates. Almost anyone worldwide can access DELPHI (using Tymnet, Telenet or other networking services) via a local telephone call.

TO JOIN DELPHI

1. Dial 617-576-0862 with any terminal or PC and modem (at 2400 bps, dial 576-2981).

2. At the Username prompt, type JOINDELPHI.

3. At the Password prompt, enter STLOG.

For more information, call DELPHI Member Services at 1-800-544-4005, or at 617-491-3393 from within Massachusetts or from outside the U.S. DELPHI is a service of General Videotex Corporation of Cambridge, Massachusetts.

BETA-CAD

"The ultimate, easy to use CAD system"

⇨ ***Finally,*** a powerful new **CAD** package developed for the professional **CAD** user, yet easy enough for the novice. **BETA-CAD** is a full functioned professional drawing package which will produce quality drawings in less time *than any other package!*

Features Include:

☆ ***FAST, DIRECT ACCESS*** to **120** powerful commands! All the power of other packages without cumbersome "TREE MENUS" to slow you down. The entire screen is used for drawing & Command entry is a breeze!

☆ **Runs** completely from **RAM!** No annoying disk accessing!

☆ **Instant** online **HELP** for quick recall of all commands.

☆ **Auto dimensioning** places leader lines, arrowheads, and dimensions with a click of a button. **Beta-cad** also dimensions in **feet** and inches for architectural drawings. Check the competition on this!!

☆ **Dimensions** and **text** can be **"dragged"** any place on the screen and repositioned instantly! Try this on expensive CAD systems!

☆ **Commands** such as STRETCH, DUPLICATE, ROTATE, AUTO INTERSECT, BETA STORAGE, BETA MATH, OFFSET, MIRROR, FILLET, MEASURE, ADD DRAFT, ADD SHRINKAGE, SPLINES, ALTER GRAPHICS, and many others ***make this system fly!***

☆ **Print** your drawing to **any scale** on a dot matrix printer or **plot** your drawing on a professional plotter^M. **BETA-CAD** will even **plot your drawing** for you via our modem!^MM

☆ **Over 1700** entities and **500** drawing layers on the **520 ST**.

☆ Full **DRAWING and COMMAND TUTORIAL** make this system easy to learn ***even for the first time user!*** Ask our **9th grade** user.

☆ **Written** in FAST, POWERFUL, **GFA** basic with unique routines!

☆ **. . . And Beta-Cad** is now available for a **LIMITED TIME ONLY** at the **special introductory price of:**

$89.95 Reg $139.95

DON'T MISS OUT! OFFER ENDS 09/30/89 CALL TODAY!

"After reviewing several CAD packages, I immediately purchased BETA-CAD and the necessary Atari hardware to use this system – it's great!" — Greg Roy, Eng, Amherst Chain Inc.

Keep your money in your bank accounts — **BETA-CAD** truly is: ***"The serious CAD system, without the serious price!"***

NOW FOR THE 520, 1040, AND MEGA ST

CALL 1-800-695-9175

BETA-CAD • 31 MILLARD RD • NORTH ATTLEBORO MA 02760 (508) 695-9175

^MMCall for details *^MPlotter Driver optional* *Requires High Res Monitor*

CIRCLE #107 ON READER SERVICE CARD.

USE THE FORSLOOK:

GLIDING THROUGH STARGLIDER II

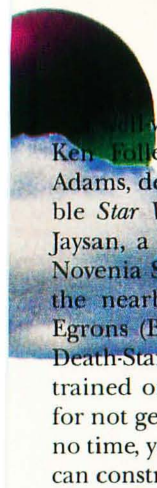
JAMES M. CIRILE

The year 1988 was alternately a dreadful and a splendiferous year for ST gamers. Things were lookin' a mite bleak for a while—we had to contend with the likes of *Test Drive* and its ilk while impatiently awaiting the messianic arrival of *Dungeon Master*. *DM* was everything we could have hoped for; not just a game, but a world, an experience. And it changed forever the rules and everyone's expectations.

How could software developers possibly compete with *DM*'s shadow lurking over their shoulders? Most of them couldn't, and didn't try. Except Argonaut Software (released in the U.S. by Rainbird). Enter *Sarglider II*, late in '88. And how fortunate we are to have received two incredible games in one year.

At first, *Starglider II* seems "merely" another vector-graphic 3-D-motion-simulating shoot-'em-up, rather like the underrated *Arctic Fox*. And it is. However, after a few games it becomes clear that *Starglider II* is much more: a multilayered quest game in which blasting aliens is actually secondary and largely unnecessary. Eventually, you're forced to ignore the spectacularly rendered enemy forces and concentrate solely on planet-hopping, searching for Neuron Bomb components from around the Solice Star System. As each component is found, it's as rewarding an experience as slaughtering your first purple worm or figuring out how to get the babel fish.

Well, maybe not quite as rewarding as getting the babel fish.



The Set Up

A well-written and amusing novella by Ken Follett, who's up on his Douglas Adams, details the cliché but still enjoyable *Star Wars*-derived scenario. You are Jaysan, a reluctant hero pilot from the Novenia Star System who has infiltrated the nearby Solice System because the Egrons (BAD GUYS) are constructing a Death-Star-like Space Station; its guns are trained on your home planet. So much for not getting involved. Within virtually no time, you must find the only man who can construct a weapon powerful enough to vaporize the Space Station (no mean feat), Professor Halsen Taymar, locate all the materials to build this Neutron Bomb (scattered on planets and moons throughout the system), hold the enemy forces at bay while it's constructed and ultimately deliver the nuke to the Egron Space Station as a going-away present.

Sound easy? It is, actually—it's just tricky as heck. The Egron Space Station and its accompanying projector bases (on the moons of the red-giant planet Millway) are constructed at a frightening rate, imbuing the adventure with a nerve-wracking, adrenaline-pumping frenetic atmosphere. There's also the additional urgency of your ship's iminent power drain. Shields, lasers and engines all sap precious power. Estimate your vessel's energy status incorrectly, and you may run out of juice in deep space, far from any refueling sites. You can never get too comfortable in this game; there's always some pressing need that's of more importance than the task you've just completed. Trying to figure out what to do, where to go and in what order is half the game's fun and strategy.

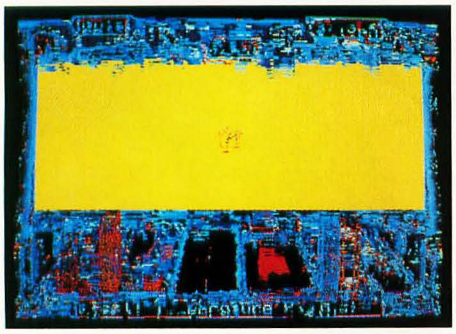


FIGURE 1

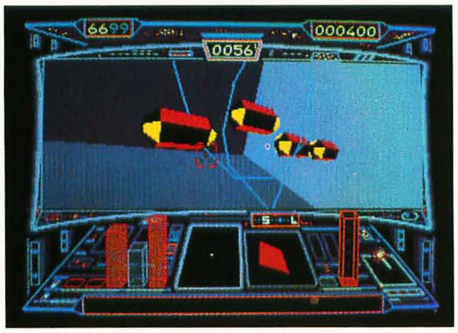
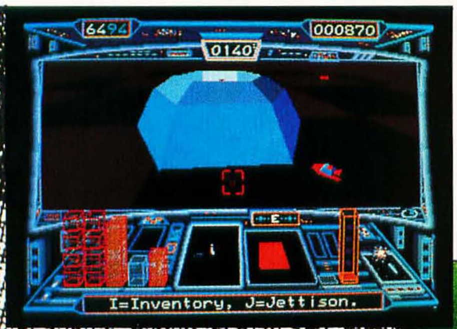


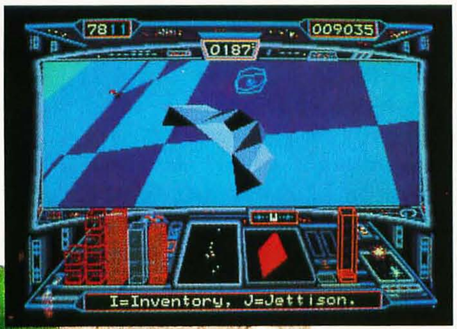
FIGURE 2

Any way you play it, *Starglider II's* amazing graphics, sound and even a built-in drawing program make this more than a winner—it's a classic.



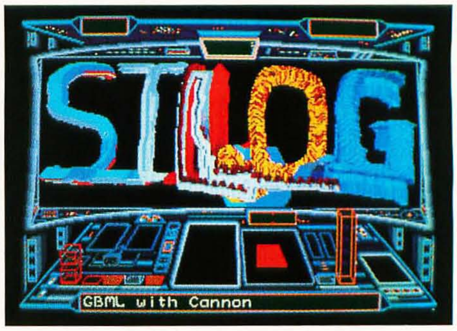
I=Inventory, J=Jettison.

FIGURE 3



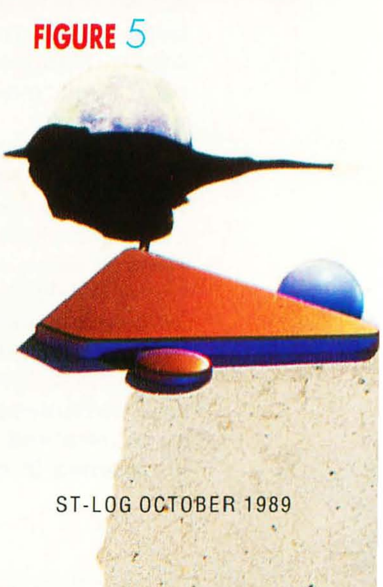
I=Inventory, J=Jettison.

FIGURE 4



GBML with Cannon

FIGURE 5



Topping everything off is the finest animation I've ever seen on any home system. The updating is so incredibly swift that you can believe this universe really exists. The animation is almost twice as fast as *Arctic Fox*'s, although the terrain isn't as varied. Up to 24 frame movements exist on each object or ship, not counting P.O.V. rotations, zooms or perspective changes. That means you can fly under, around and through things, and see them depicted accurately in even the most extreme angles as they move. You are *there.* Rainbird has given us an entire solar system with seemingly real physical space to explore.

Warning: Do Not Read Unless You Want the Game Spelled Out for You.

You begin on Apogee, one of the Solice System's five planets, surrounded by hordes of vitriolic Egrons. It may be quite some time before you survive longer than a minute or two, since flying your craft, the Icarus, takes some getting used to and the BAD GUYS are particularly nasty here. Your best bet is to disobey the documentation and ditch the mouse. Icarus is quite impossible to control by mouse. Save it for point-and-click games, like *Dungeon Master* and *ChronoQuest*. Not just a few games have been rendered unplayable (*Tanglewood*, for one) because their designers opted for mouse-control on what should've been a joystick game. Fortunately, *Starglider II* gives you the choice. My nine-year-old Atari VCS stick, veteran of entirely too much 2600, Colecovision and C-64 abuse, worked perfectly. Set your cross-hair autocenter to "X" and avoid the other options, and you'll be flying fine.

While Apogee is a good training ground, your best bet is to vamoose, pronto, before a stray shot takes out one of the all-important power lines. If you're a fledgling pilot, dive directly into the underground tunnel network. Here you'll find another graphic miracle—and the game's only glitch—blast doors disappear momentarily as you pass them. No big deal.

Map this network carefully! There are dead ends, deserted depots and miles of circles to get lost in. (Incidentally, tunnel-flying is a bit easier when you toggle to Icarus' outside view—just behind, and looking past, your ship. Press "Enter." Another wonder.) When you find the depot, write down its coordinates and what they tell you to retrieve, then haul your carcass into space and get searching.

If you've done all this already, a trip to the depot isn't necessary until you've rounded up three of the components needed for The Bomb's construction. Head immediately for Castron, one of Apogee's moons.

Before we get to Castron, one word about the solar system: decoys. There are many wondrous places to visit in the Solice System, both above ground and below, and for fun's sake it's pleasant to buzz them and enjoy the marvelous animation of each planet's indigenous creatures. However, to finish the game, you never have to visit and should probably avoid the following places: Vista, Q-Beta, Esprit, Wackfunk Depot, Enos, Westmere Depot, Millway and, of course, the sun.

Wackfunk Depot provides staggeringly insignificant ship repairs in return for goods; Esprit will supposedly give you a case of Nuclear Fuel in exchange for your pilot couch. Alas, the only way to surrender your seat is by running out of energy while in the depot, thereby destroying your ship. Not a good thing to do. You'll just have to get the Nuclear Fuel someplace else. And though one does have to tight-orbit Millway to find Egron ducks and whales (required by other depots to trade for Bomb components), flying too close will immolate your ship. Similarly, Solice is useful only for the LSD-inspired graphics it provides as you die a colorful, melting death (Figure 1). Supposedly, you can recharge Icarus' systems via a close flyby, but I've yet to pull this maneuver off. Recharging is also (supposedly) possible by sitting on one of Dante's volcano's lips, but all that's yielded me has been a lava-ly demise. Stick to asteroids and power lines for juicing up. And avoid unnecessary jaunts to worthless planets.

Castron is a useful place, and you'll have to go there a few times before you finish the game. Castron has it all: an extensive array of efficient power lines (buzz them immediately upon planetfall), mineral rock scattered about the surface (tractor-beam it aboard) and, hidden deep in the tunnels, the indisputably handy bouncing bombs (Figure 2). You can pretty much dispense with what they're trading at the depots. The wine and Castrobars can be obtained for much less trouble by raiding the Pirate Ships (more on this momentarily). Get four bouncers and skedaddle. Don't worry about using them up, because each time you return to Castron they'll be restocked. And don't forget to map the tunnels!

It's about time to deal with the moons of Millway, your main source of trade and blistering warfare. Its seven moons are uniformly inhospitable and, worse, have no available power sources. That means each trip to a moon requires a stop at the asteroid field between Apogee and Millway. Asteroid Methane Fragmentation Refueling (wherein you catch an asteroid in your tractor beam and siphon its rotational energy) is thankfully easy, but be careful: Linger too long, and pirates will come to party. They'll steal whatever goods you're carrying if you don't expeditiously engage them. Also, be careful not to stop anywhere but at the edges of the asteroid field. Heaven help you if you're anywhere near the middle. . . .

Millway's moons all have tunnel networks, each depot carrying four "Fire & Flee" missiles. In addition, the depots provide the following:

MOON	GIVE	RECEIVE
1) Broadway:		(Prof. is here)
2) Apex:	Pyramid	Mineral Rock
3) Esprit:	Pilot Seat	Nuclear Fuel
4) Questa:	Whale	Flat Diamond
5) Westmere:	Asteroid	Egron Mini-Rocket
6) Synapse:	Egron Duck	Cluster of Nodules
7) Wackfunk:	Any goods	Menial repairs

All of these components can be found on various planets, also.

By now, one, probably two projector bases have been erected on Millway's moons. Time to try out one of those bouncing baby bombs! After a pitstop at the asteroid field, bolt over to the nearest projectored moon (Broadway, most likely) and park just above and behind the projector base. Don't waste gas lasers (or missiles, if you've gotten them yet) on any part of the base, including the Sentinel; just drop a bouncer, sit back and savor the stupendous graphics feast (and the additional 8,000 points you've added to your score).

If you're not on Broadway, go there now. Fly and map the tunnels; this is important, since all Millway's moons share the same tunnel layout. At the depot you'll be informed that the dear, insane Professor isn't in (yikes!). He's out cruising the planet in his beautifully rendered Emma 2 jet-car enjoying the fabulous graphics! Capturing him requires the trickiest flying you'll ever have to do. At first, catching the zigzagging vehicle seems impossible. It blazes along at three-quarters of your top speed in an entirely erratic, drunken fashion. Try chasing him, and ground-based cannons will make brie out of Icarus before you can say Caspar Weinberger. You can eliminate a few with

F&F missiles, but. . . if you'll notice, the Prof. has a penchant for a particular strip of moon between the two tunnel entrances! Park near one and wait for him. You'll be shielded from the shells, and he'll eventually whiz right past you! (See Figure 3.) Floor it after him, tractor him in and save the game quickly so you won't have to do this again.

Nabbing the Prof. will also give you one of his inventions: the Time-Warp Cuboid launcher. It clones whatever it hits, projecting it forward one second in time; this has the effect of making objects collide with and destroy themselves. Only trouble is, it doesn't work. Well, even geniuses have their failings. Stick to lasers and missiles for effective results.

Back to Apogee with your booty. You'll probably need another sojourn at your friendly neighborhood asteroid field. On Apogee, charge to full with the power-line stunt if you need to, then submerge. The Prof. will begin working on The Bomb, leaving you to cope with the new projector bases that've no doubt been built.

Before we mulch another projector base, an easy jaunt: Blast off for Millway, stopping just on the asteroid field's brink (on the Apogee side). Beam-in an asteroid and await the pirates. After a scalding space battle, you should have tractored aboard their dropped cargo: Castrobars and Vistan Wine. Zip back to Apogee and bless the depot with three more items from their wish-list. Mineral rock can also be obtained from the Pirates.

Now we can drop by a Millway moon for some projector mashing. After this bit of fun, you can pretty much choose your priorities. Use the chart to find the remaining components and their moons, and obtain the items in whatever order you see fit. As I mentioned earlier, whales and ducks can be found in the Millwayan atmosphere. They're needed to obtain the Flat Diamond and Cluster of Nodules. Pyramids can be found on most planets (including Apogee), as can Egron Mini Rockets. Gather up another load of stuff and convey it to the Prof.

Right around here you'll receive a message on your Mini-Screen: "Bomb Completed." How is this possible without all the key elements? On Apogee, you'll discover the answer. If you attempt to collect The Bomb without fulfilling the depot's list, The Bomb itself will punish you severely. Leave it there along with the favors you bring, and set out for the last item: the case of Nuclear Fuel. Where in tarnation is it?

Aldos. And it's a *long* way off. First, it might pay to venture to Castron for some bouncers and the power lines. On the way to Aldos, it's probably a good idea to remind the enemy just how dazzling the animation is when a bomb annihilates one of their projectors.

Aldos is heavily fortified, but by now this should be a cake-walk for an experienced pilot like yourself. Scoop up an Egron Mini-Rocket if you haven't already, and reconnoiter the terrain. Notice those odd striations every now and again? Hover near one for a bit and watch what happens. If you're lucky, you'll discover the purpose they serve. Try to tractor-beam the fuel up. No go, but the picture should be clearer by now. A well-placed F&F missile will extricate the mine guarding the fuel, or you may be able to swipe it before the Egrons tap into it, locking it into the grid (Figure 4). Hopefully, you've enough power to get back to the asteroid field and refuel. However, don't return to Apogee just yet. Unfortunately, The Bomb is very unstable; even slight changes in atmospheric pressure can detonate it (such as entering or exiting a tunnel network; odd that reentry and breakaway don't affect it). Get The Bomb later. You've got some wanton destruction to take care of first.

Zoom to Q-Beta's "moon" (really a 3-D projection), where you'll see the Space Station surrounded by its seven Sentinels. To finish the game, you need to destroy all the Sentinels, all the projector bases and pick up The Bomb and wing back to Q-Beta to deliver it, Federal Distress, to the baddies. You won't be allowed to launch the bomb if even one of the bases is operational. So for now, dispatch four F&F missiles to knock out some Sentinels. These, happily, will not be rebuilt.

Off to Millway's moons. Hopefully, you've got at least three bouncing bombs remaining; if not, you'll need to visit Castron at least two more times. One last trip to a Millway moon depot is in order—replenish those missiles. To save time, do this before you start moon-hopping and wiping out projectors. Once you've dismantled as many projectors as possible, time becomes of the essence. You must vault over to Castron (more bouncers), pick up The Bomb (and recharge) on Apogee, destroy four more Millway projectors and get back to Q-Beta before more than one projector is rebuilt. Have a bowl of Wheaties and get crackin'.

Now they let you collect The Bomb. Things go much more smoothly when you listen to people, no? Be careful exiting Apogee's tunnels—fly too fast and you're grits. From this point on, any potential source of collision must be avoided. That Bomb's just too dang sensitive, swinging away under the Icarus in its "forslook" containment harness. (Click on the outside view for a good peek at this.) Whatever you do, don't brave the asteroid field! A quick round of projector-mulching later, you should be on your way toward your destiny. Use the forslook!

The End?

Quickly now, wipe out the remaining Sentinels with the F&F's, target the Space Station and feed the baby his nukeburger! *KAAA-BLOOOEY!* Congrats, young Luke, er, Jaysan, you've wiped the Egron scourge from the galaxy. . . for now. Horror sets in as you realize that this hasn't really stopped the Egrons at all—they're already rebuilding their projectors—and a new, more powerful Space Station is under way.

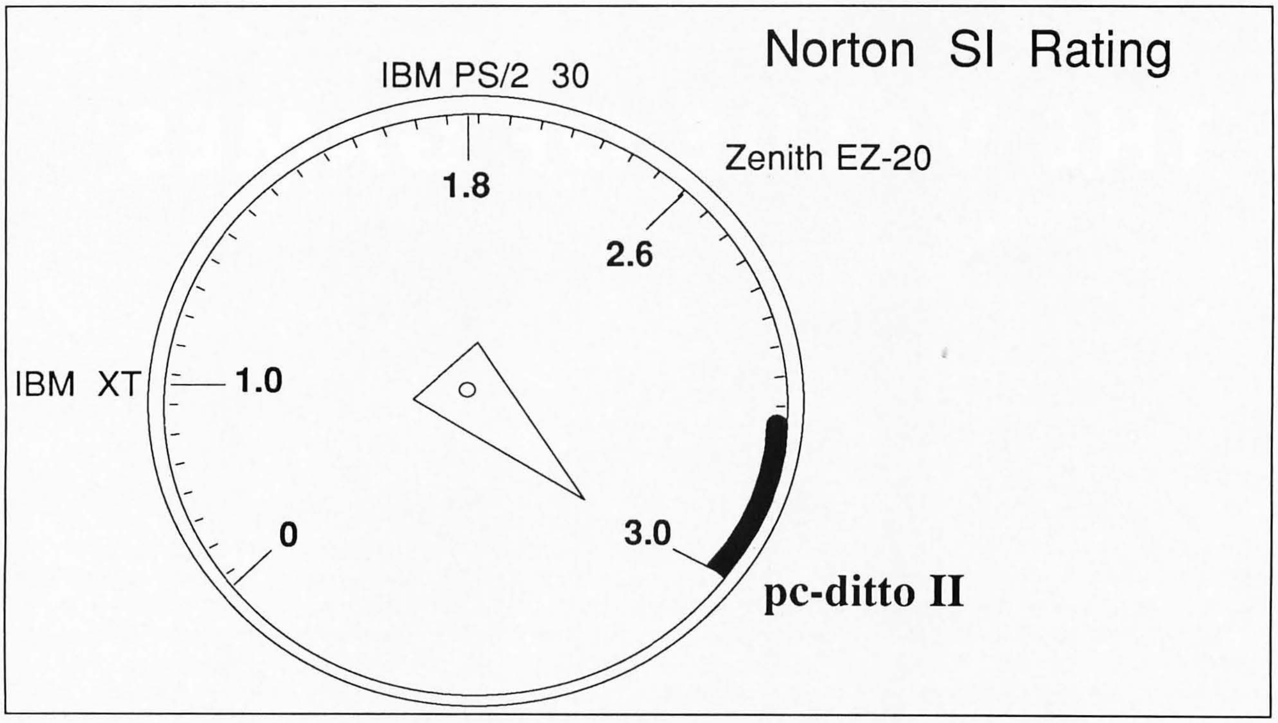
And for those of you who like the bad guys to win, wait around and watch the Space Station. You'll soon receive these messages on your Mini-Screen: "Space Station Arming Main Guns," "Space Station Firing Main Guns" (it unleashes hundreds of blue energy spheres at your home planet, Novenia) and "Satellites Report: Novenia Destroyed." Bah. Much more fun to fry the heathen enemy scum!

One great thing about this game is that there are myriad strategies for winning. If you like, you can stall the Space Station's construction by bombing a projector every few minutes, and just play for the score, wiping out vast quantities of Egron forces. Any way you play it, *Starglider II*'s amazing graphics, sound and even a built-in drawing program (Figure 5) make this program more than a winner—it's a classic. Kudos to all the designers: Jez San, Richard Clucas, Alister Perrott, Paul Gomm, Chris Humphries, Pete Warnes, Tim Watson and Adrian Friday. And to all the other software designers on this planet: Best of luck topping this one, guys. ▪

***Jim Cirile** is a New York-based screenwriter and film director whose first feature,* Banned, *will be released about the time you read this. When not shackled to his 1040, Jim also plays bass in the rock band* Export, *draws, paints and complains quite a bit.*

pc-ditto II

By Avant-Garde Systems



IBM XT COMPATIBLE !

IBM AT PERFORMANCE !

ATARI ST PRICE !

$299.95

Now! Run the most popular IBM software on your Atari ST...

FAST !

See pc-ditto II at your local Atari dealer or write for free information.

Yes. Please send me more information !

Avant-Garde Systems, 381 Pablo Point Drive
Jacksonville, Florida 32225

Name____________________

Address____________________

City________ **St**____ **Zip**______

The Compukid Connection

THE VALUE OF GAMES

BY D.A. BRUMLEVE

Program and program instructions
BY FRANK M. HUNDLEY

Children love to play games, and computer games are no exception. Parents often feel that game-playing is a poor use for their expensive home computers—and sometimes they are right. But the public-domain and commercial software markets offer a wide variety of games, some of which make ample use of the editing and problem-solving features of the computer, giving kids a distinct advantage over their non-computer counterparts. While every category of ST game offers several with the potential to provide hours of entertainment, some kinds of games have more long-term value for children than others.

One type of game that immediately comes to mind as "valuable" is the skill-teaching game. These provide practice in a specific academic skill and can be helpful to children who need that special practice. Computer games of this kind can provide instant feedback. The "game" aspect of the package may serve as a motivator for reluctant students, but it will not mask the academic purpose of the program. The educational value of these programs is obvious to adults and children alike, but there are several other kinds of games in which learning is not a goal of the program, but rather a wonderful by-product.

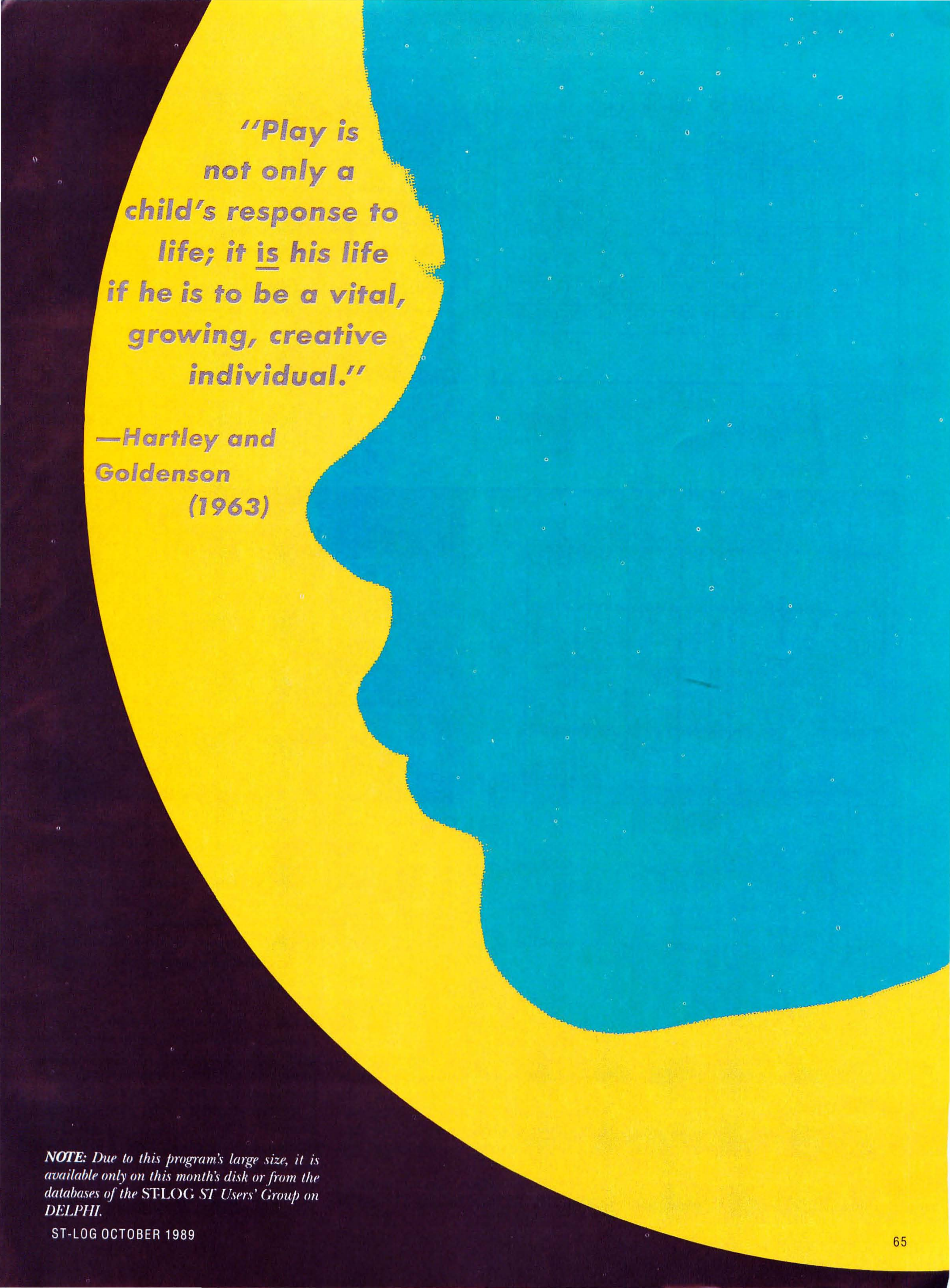
Games like *Pac-Man* do have relatively little value beyond entertainment. Strategies must be discovered, but, once learned, the gameplay is monotonous and one's prowess on succeeding levels is determined largely by the ability to move the joystick quickly; thinking is barely necessary. Kids love these mindless games, but a few quarters spent at your local arcade will often give them enough time with a game to become proficient at—and bored with—it. There are arcade games, however, which do have real value. Arcade games, like *Time Bandits* (MichTron) and *DGDB* (public domain), require the development of more sophisticated strategies. And when these games are played by two players, cooperation is much more likely to lead to success than is competition. Some games have an arcade component that grabs the child's interest, while other aspects of the same game will challenge thinking skills. *Sundog* (FTL) and *Defender of the Crown* (Cinemaware), for example, include arcadelike activities as well as challenging strategy.

Word games, crossword puzzles and jumbled stories can reinforce spelling and word construction rules learned at school. Some of these games are easy enough for beginning readers, while others can challenge a college-educated adult. There are many such games for the ST in the public domain, and commercial games, such as *Scrabble* (Parker Brothers), are included in this group.

With some games, children may not benefit from the learning value without a little nudge from their parents. Strategy games, simulations and adventure games are especially challenging to young thinkers; if they do not succeed quickly, children may abandon these programs and the learning potential they offer. Small interventions from parents can greatly increase a child's enjoyment of and benefit from a particular game. Also, because typical adventure games are designed for adults, it is a good idea to make a child's first game a family activity. By playing the game with the child, parents can observe the strategies the child is using and bring them to the child's attention. Parents or an older sibling can help the child construct a map of a game's storyscape (see Figure 1), complete with hints as to what can be found at each location; a map can prove an invaluable aid in solving a game, and the process of creating one has some direct academic value. Such maps can be shared with friends, who may be able to expand them and add to the hints provided.

Some games take better advantage than others of the unique features a computer affords. A few provide "construction kits" with which the user can design screens for use in the game. Even when the game itself is a mindless one, use of a construction kit can be a creative and thought-provoking task.

Traditional games, such as board games and card games, are often converted for



"Play is not only a child's response to life; it is his life if he is to be a vital, growing, creative individual."

—Hartley and Goldenson (1963)

NOTE: *Due to this program's large size, it is available only on this month's disk or from the databases of the* ST-LOG *ST Users' Group on DELPHI.*

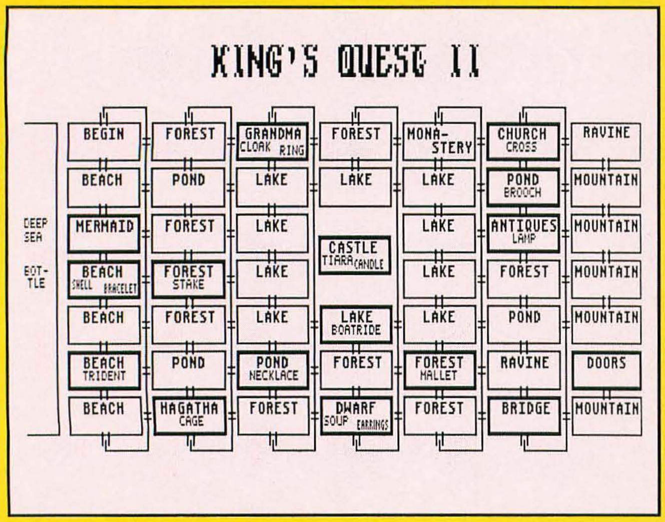


FIGURE 1 **My seven-year-old son and I developed this storyscape map while on our way to solving *King's Quest II* (Sierra).**

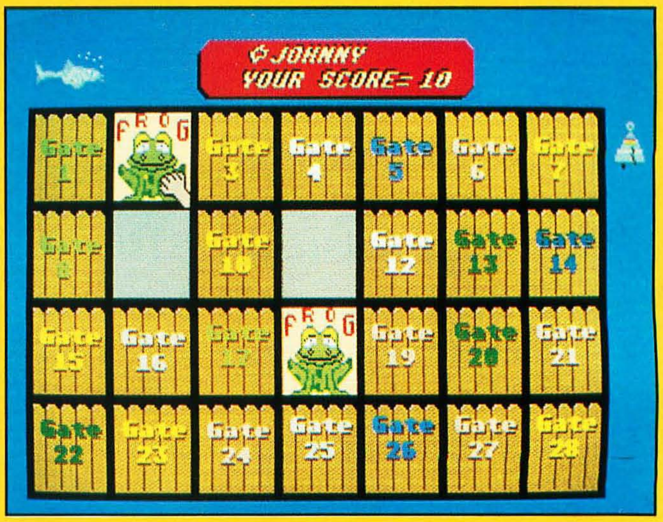


FIGURE 2 **A successful match on level one!**

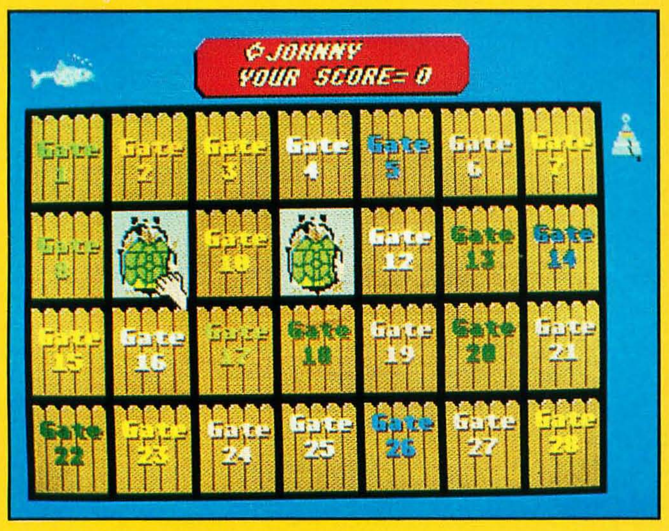


FIGURE 3 **The turtles are a match, but look out! A shark is coming, and you must ring the bell on the right edge of the screen or the match points will be lost. At this point in the shark's approach, its jaws are a faint outline around the turtle, and the player has enough time to ring the bell—if he moves quickly.**

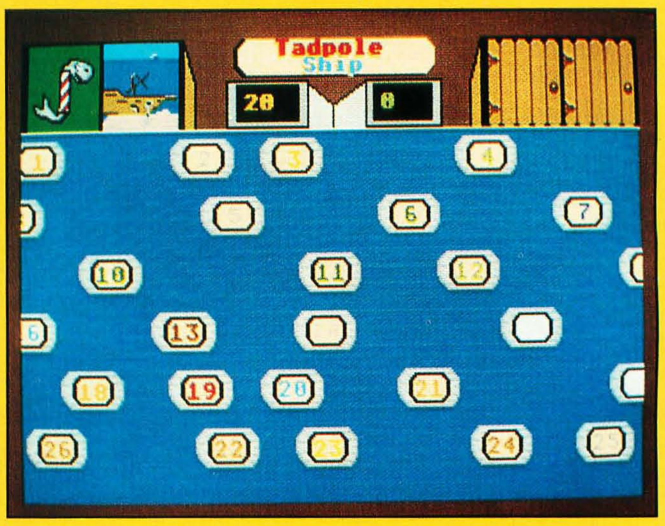


FIGURE 4 **On level three, the computer has presented a tadpole for matching, but the door (raft, in this case) that the player chose had a ship under it. In level three, the doors move until the player clicks the mouse to stop them. Each succeeding level in *Seaside* offers increasing difficulty. Even adults will find a challenge in levels three and four.**

use on computers. There are versions of backgammon, Monopoly, chess and solitaire in the public domain. There are commercial versions of bridge, crazy eights and mah-jongg. Games such as these, whether in tabletop or software form, can be fascinating and stimulating. The computer versions typically offer the added advantage of instantaneous editing: saving a game at a specified point, returning to an earlier point in the play, testing possible moves before deciding on the final one, customizing the play pieces, etc. Conversions that exploit the special features of computer editing may well be worth the added investment over the traditional tabletop product. An only child will especially appreciate the ability of the computer to provide an opponent.

One conversion that takes advantage of the special abilities of the ST can be found in this month's program: *Tiffany's Seaside* is a charming memory-matching game featuring various kinds of sea life hidden by numbered doors. *Seaside* contains many features that increase its value and interest, and it can be played by children of all ages (and also adults). The challenge on the upper levels is quite taxing, while a two-year-old who may not even understand the object of the game can enjoy simply knocking on the gates to see what's behind them. Targets for the mouse are large and easy to hit.

Seaside's programmer, Frank M. Hundley, has included many features that make the game more enjoyable to play than a simple box of matching cards. He provides auditory, visual and kinesthetic feedback to aid memory. The numbers on the doors help the child identify and remember matches, while the sounds made by the animals,as well as the charming animations, the delightful touches of humor and the threat of a random attack by a shark focus a child's attention. A fish at the left or right of the top portion of the screen indicates who's turn it is, and the fish claps when the child succeeds.

The program, written in GFA BASIC, is on your ST-LOG disk in the file named SEASIDE.ARC. The program and associated files will occupy most of a single-sided disk. Follow the disk instructions to decompress the file, then copy all the files to a freshly formatted disk. *Tiffany's Seaside* will run from a hard drive and from a folder. If you do use a folder, be certain that all related files are in the same folder with SEASIDE.PRG. The program requires low resolution.

Game Instructions

Tiffany's Seaside is a children's memory-matching game with four progressively challenging levels. Like the old "Concentration" card game, the goal is to pair up matching objects hidden behind doors. The objects in this case are animated sea creatures. To play *Seaside*, you need a mouse and color monitor set to the low-resolution mode.

When you start the game, press the mouse button to get past the introduction screen. The players will then be asked to enter their names on the keyboard. Any mistakes can be corrected by pressing the backspace key. This will give you a chance to start over.

The next screen is an important one. You will be given a choice of playing level one only, level three only, levels one and two, levels three and four or all levels at once. To make your choice, put the mouse pointer in the appropriate box and press the left mouse button. Click the PLAY button the same way to move on.

The last screen before actual play gives you three more choices. You can click on PLAY to start immediately, or if you feel you need some more information for the level you are about to play, click on INSTRUCTIONS. Clicking QUIT will terminate the game and return you to the desktop.

Level One

When you enter level one (see Figure 2), you'll see a series of numbered doors. At the top, middle of the screen is a scoreboard, which also keeps track of the turns. Player one starts by knocking on a door. Put the fist-shaped mouse pointer on the door and press the button to knock. The door will open and reveal a creature. Now you knock on one more door. If you find the matching creature, you score ten points and retain your turn. If not, the mouse changes hands and it is the second player's turn.

All creatures are worth ten points, except the Shark, which is worth 20 points, and the Treasure Chest, which can be worth zero, ten, 20 or 30 points. When you match the Treasure Chest, watch it open on the screen to reveal your booty.

Occasionally, after you make a successful match, a Shark may try to rob you of your points (see Figure 3). When you see one of these Shark attacks, the only way to save your points is to ring the Shark-Alert bell on the right side of the screen. Put the pointer on the bell and press the button. If you sound the alert in time, the Shark disappears. If not, the bell won't ring and you lose the points from that match—as well as your turn.

Level Two

The matching continues on level two, but now the creature that you have to locate will be picked for you. At the beginning of each turn, a creature will appear from behind a special door on the top of the screen. You knock on one door, only this time to find a creature that matches. A different creature will be picked for each turn.

You still have to keep an eye out for the Shark and a new addition, the Ghost. After any turn, a Ghost sprite may appear at the door that was just opened. It will travel along the doorways until it eventually stops and disappears in front of another door. This routine signals that the creatures behind the two respective doors have traded places. The creature that was behind the door where the Ghost started its journey is now behind the door where the Ghost stopped, and vice versa.

Level Three

The third level (see Figure 4) plays just like the second level, except the Ghost and the Shark attacks are absent. The increase in challenge comes from the doors, which are now all moving. To pick a door on level three, press and release the left mouse button once. This will temporarily stop the movement of the doors and bring the hand pointer into view. At this point you can go ahead and knock on a door. If you find that the door you want is not on the screen, you can press the right mouse button to start the doors moving again. You can also wait a few seconds and the doors will start moving again on their own. After a turn, the doors resume moving automatically.

Level Four

On the fourth and final level, the doors still move, but, as in the first level, you knock on two doors to make a match. At the start of the game, you see two closed doors on the top of each player's side of the screen. These will open and show your picks as you make two attempts to match a creature. With the doors always moving, finding the location of both creatures can be quite difficult, but since it is the last level, the Ghost is back to add to the challenge even further. After a turn, he may be seen traveling between the two doors at the top of the screen. This means the creatures behind the last two doors chosen have exchanged positions.

Conclusion

Tiffany's Seaside has been used by more than 200 preschool through second-grade students at Creative Discovery School in Champaign, Illinois, and at Leal Elementary School in Urbana, Illinois. It is among the favorites most often cited by these students, and we know your child (and you!) will also enjoy it tremendously. ■

***D.A. Brumleve**, M.A., is involved with children and computers in a variety of ways. The mother of five children, ages 2 to 10, she serves as the adult facilitator of the Children's ST Users' Group in Urbana, Illinois. An avid programmer, she has developed a beginner's course in GFA BASIC and is the author of* PreSchool KidProgs *(MichTron) and numerous freely distributed programs for young ST users. Her daughter, Catherine, is a student at Creative Discovery School.*

***Frank M. Hundley** has given up repairing and installing electronic hardware, and now finds more satisfaction in programming. Most of his programs were written to introduce his niece, Tiffany, to the world of computers.*

C·CHEST
Listing 1:C

```
/***********************************************************************/
/*                                                                     */
/*  Program Name:  CHARTEST.C                                          */
/*  Purpose:  Exercises a series of functions that perform character   */
/*            string operations.                                       */
/*  Written By:  Karl Wiegers                                          */
/*  Date Written:  April 2, 1989                                       */
/*  External Functions Called:  COPIES, LEFT, RIGHT, STRREV, SUBSTR,   */
/*                              WORDS, WORD, OVERLAY                   */
/*                                                                     */
/***********************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *copies(), *left(), *right(), *strrev(), *substr(), *word();
extern char *overlay();
extern short words();
extern char *malloc();   /* don't forget this for Laser C */

main()
{
   char *string, *string2;
   short length;

/***********************************************************************/
/*                                                                     */
/*  First test COPIES function, which returns a pointer to some number */
/*  of copies of an input character string.                            */
/*  syntax:  COPIES ( *string_to_copy, number_of_copies )              */
/*                                                                     */
/***********************************************************************/

   string = "Now is the time.";
   printf ( "Demonstration of COPIES function:\n\n" );
   printf ( "Twenty x's = %s\n", copies ( "x", 20 ) );
   printf ( "Three 'Now is the time.'s = %s\n", copies (string, 3 ) );

   printf ( "\nPress RETURN to continue.\n\n\n" );
   getchar ();

/***********************************************************************/
/*                                                                     */
/*  Now test STRREV function, which returns a pointer to a character   */
/*  string which consists of the characters from a supplied input      */
/*  string, but in reverse order                                       */
/*  Syntax:  STRREV ( *string_to_be_reversed)                          */
/*                                                                     */
/***********************************************************************/

   printf ( "Demonstration of STRREV function:\n\n" );
   printf ( "Original string = %s\n", string );
   printf ( "After reversal  = %s\n", strrev ( string ) );

   printf ( "\nPress RETURN to continue\n\n\n" );
   getchar ();

/***********************************************************************/
/*                                                                     */
/*  Now test LEFT function, which returns a pointer to the leftmost    */
/*  'n' characters of a supplied input string.                         */
/*  Syntax:  LEFT ( *input_string, number_of_characters)               */
/*                                                                     */
/***********************************************************************/

   string = "Now is the time for all good men.";
   printf ( "Demonstration of LEFT function:\n\n" );
   printf ( "Original string = %s\n", string );

   printf ( "leftmost character = %s\n", left ( string, 1 ) );
   printf ( "leftmost 10 characters = %s\n", left ( string, 10 ) );
   length = strlen ( string );
   printf ( "leftmost %d characters = %s\n", length, left ( string, length ) );
   printf ( "leftmost 50 characters = **%s**\n", left ( string, 50 ) );

   printf ( "\nPress RETURN to continue\n\n\n" );
   getchar ();

/***********************************************************************/
/*                                                                     */
/*  Now test RIGHT function, which returns a pointer to the rightmost  */
/*  'n' characters of a supplied input string.                         */
/*  Syntax:  RIGHT ( *input_string, number_of_characters )             */
/*                                                                     */
```

C CHEST

```
/************************************************************************/

   printf ( "Demonstration of RIGHT function:\n\n" );

   printf ( "rightmost character = %s\n", right ( string, 1 ) );
   printf ( "rightmost 10 characters = %s\n", right ( string, 10 ) );
   printf ( "rightmost %d characters = %s\n", length, right ( string, length ));
   printf ( "rightmost 50 characters = %s\n", right ( string, 50 ) );

   printf ( "\nPress RETURN to continue\n\n\n" );
   getchar ();

/************************************************************************/
/*                                                                      */
/*  Now test SUBSTR function, which returns a pointer to substring of   */
/*  characters of a supplied input string, beginning at a specific      */
/*  character position and extending for a specified number of          */
/*  characters.  A pad character is also supplied in case the          */
/*  substring requested extends beyond the end of the input string.     */
/*  Syntax:  SUBSTR ( *input_string, starting_position,                 */
/*                    number_of_characters, *pad_character )            */
/*                                                                      */
/************************************************************************/

   printf ( "Demonstration of SUBSTR function:\n\n" );

   printf ( "substr(12,4) = %s\n", substr ( string, 12, 4, " " ) );
   printf ( "substr(1,13) = %s\n", substr ( string, 1, 13, " " ) );
   printf ( "substr(1,33) = %s\n", substr ( string, 1, 32, " " ) );
   printf ( "substr(21,20), blank pad = **%s**\n",
            substr ( string, 21, 20, " " ) );
   printf ( "substr(21,20), pad is 6 = %s\n", substr ( string, 21, 20, "6" ) );

   printf ( "\nPress RETURN to continue.\n\n\n" );
   getchar ();

/************************************************************************/
/*                                                                      */
/*  Now test WORDS function, which returns the number of blank-         */
/*  delimited words in a supplied input string.                         */
/*  Syntax:  WORDS ( *input_string)                                     */
/*                                                                      */
/************************************************************************/

   printf ( "Demonstration of WORDS function:\n\n" );

   printf ( "Should be 8:  %s = %d words\n", string, words ( string ) );
   strcpy ( string, "" );
   printf ( "Null string:  %s = %d words\n", string, words ( string ) );
   strcpy ( string, "     " );
   printf ( "Just some blanks:  %s = %d words\n", string, words ( string ) );
   strcpy ( string, "     Some leading blanks." );
   printf ( "Some leading blanks:  %s = %d words\n", string, words ( string ) );
   strcpy ( string, "Some trailing blanks.     " );
   printf ( "Some trailing blanks:  %s = %d words\n",
           string, words ( string ) );
   strcpy ( string, "Nowis the time for allgood men." );
   printf ( "Should be 6:  %s = %d words\n", string, words ( string ) );
   strcpy ( string, "Nowisthetimeforallgoodmen." );
   printf ( "Should be 1:  %s = %d words\n", string, words ( string ) );

   printf ( "\nPress RETURN to continue\n\n\n" );
   getchar ();

/************************************************************************/
/*                                                                      */
/*  Now test WORD function, which returns a pointer to a specific       */
/*  word in a blank-delimited supplied input string.                    */
/*  Syntax:  WORD ( *input_string, word_number_to_return)               */
/*                                                                      */
/************************************************************************/

   printf ( "Demonstration of WORD function:\n\n" );
   string = "Now is the time for all good men.";

   printf ( "Word 1 = %s\n", word ( string, 1 ) );
   printf ( "Word 3 = %s\n", word ( string, 3 ) );
   printf ( "Word 8 = %s\n", word ( string, 8 ) );

   printf ( "\nPress RETURN to continue\n\n" );
   getchar ();

/************************************************************************/
/*                                                                      */
/*  Now test OVERLAY function, which returns a pointer to a string in   */
/*  which one string has been overlaid upon another, beginning in a     */
/*  particular character position and extending for a specified length */
/*  Syntax:  OVERLAY ( *string_to_overlay, *string_to_be_overlaid_on,   */
/*                     overlay_character_position, overlay_length,      */
/*                     *pad_character )                                 */
/*                                                                      */
/************************************************************************/

   string = "Now is the time for all good men.";
   string2 = (char *) malloc ( strlen ( string ) + 20 );
   printf ( "Original = %s\n", string );
   strcpy ( string2, overlay("week", string, 12, 0, " " ) );
   printf ( "Example 1 = %s\n", string2 );
   strcpy ( string2, overlay("weekly", string, 25, 4, " " ) );
   printf ( "Example 2 = %s\n", string2 );
   strcpy ( string2, overlay("week", string, 12, 9, "*" ) );
   printf ( "Example 2 = %s\n", string2 );

   printf ( "\n\nPress RETURN to exit\n" ) ;
   getchar ();
}
```

C·CHEST
Listing 2:C

```
/***********************************************************************/
/*                                                                     */
/*  Function Name:  COPIES                                             */
/*  Purpose:  Return a pointer to a character string which contains    */
/*            a specified number of copies of a supplied input         */
/*            string, all concatenated into a single long string.      */
/*            NULL is returned if memory allocation fails.             */
/*  Written By:  Karl Wiegers                                          */
/*  Date Written:  March, 1989                                         */
/*  Usage Syntax:  copies ( input_string, number_of_copies_desired )   */
/*  External Functions Called:  malloc, printf, strcat, strlen         */
/*                                                                     */
/***********************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *malloc ();

char *copies ( instring, num_copies )
char *instring;
short num_copies;

{
    static char *string;    /* pointer to string of copies to return  */

/*---------------------------------------------------------------------*/
/*    attempt to allocate space to contain string of copies; don't     */
/*    forget a space for the trailing null character                   */
/*---------------------------------------------------------------------*/

    if ( ( string = malloc ( num_copies * strlen ( instring ) + 1 ) ) == NULL )
    {
         printf ( "Unable to allocate memory for COPIES\n" );
         return ( NULL );
    }
/*---------------------------------------------------------------------*/
/*  initialize output string to null character; append contents of     */
/*  of input string to the output string the desired number of times  */
/*---------------------------------------------------------------------*/

    *string = '\0';
    while ( num_copies-- )
         strcat ( string, instring );

    return ( string );
}
/***********************************************************************/
/*                                                                     */
/*  Function Name:  LEFT                                               */
/*  Purpose:  Return a pointer to a character string which contains    */
/*            the leftmost 'number' characters of a supplied input     */
/*            string, terminated by a null character.  NULL is         */
/*            returned if memory allocation fails.                     */
/*  Written By:  Karl Wiegers                                          */
/*  Date Written:  March, 1989                                         */
/*  Usage Syntax:  left ( input_string, number_of_left_chars)          */
/*  External Functions Called:  malloc, printf, strcat, strlen,        */
/*                              sprintf, copies                        */
/*                                                                     */
/*********************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *copies ();
extern char *malloc ();

char *left ( instring, number )

char *instring;
short number;

{
     static char *temp_string; /* pointer to string of leftmost chars */
     register short length;    /* number of characters in 'instring'  */

/*-------------------------------------------------------------------*/
/*    attempt to allocate space to contain string of leftmost charac- */
/*    ters; don't forget a space for the trailing null character     */
/*-------------------------------------------------------------------*/

     length = strlen ( instring );
     if ( ( temp_string = (char *) malloc ( length + 1 ) ) == NULL )
     {
          printf ( "Unable to allocate memory for LEFT function.\n" );
          return ( NULL );
     }

/*-------------------------------------------------------------------*/
/*  do internal print of leftmost 'number' characters to the address */
/*  pointed to by 'temp_string'; if 'number' was longer than the     */
/*  length of the input string, pad 'temp_string' with blanks to the */
/*  desired length.                                                  */
/*-------------------------------------------------------------------*/

     sprintf ( temp_string, "%-.*s", number, instring );
     if ( number > length )
     strcat ( temp_string, copies ( " ", number - length ) );

     return ( temp_string );
}
/*********************************************************************/
/*                                                                   */
/*  Function Name:  RIGHT                                            */
/*  Purpose:  Return a pointer to a string which contains the right- */
/*            most 'number' characters of a supplied input string.   */
/*            NULL is returned if memory allocation fails.           */
/*  Written By:  Karl Wiegers                                        */
/*  Date Written:  March, 1989                                       */
/*  Usage Syntax:  right ( input_string, number_of_characters)       */
/*  External Functions Called:  malloc, printf, strcat, strlen,      */
/*                              sprintf, strcpy, alloca, copies,     */
/*                              strrev                               */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *copies ();
extern char *strrev ();
extern char *malloc ();

char *right ( instring, number )

char *instring;
short number;

{
     register short length;      /* number of chars in 'instring'      */
     register char *copy_string; /* temporary work space               */
     static char *temp_string;   /* pointer to string of rightmost chars */

/*-------------------------------------------------------------------*/
/*  attempt to allocate space to contain the string returned; don't  */
/*  forget a space for the trailing null character                   */
/*-------------------------------------------------------------------*/

     length = strlen ( instring );
     if ( ( temp_string = (char *) malloc ( length + 1 ) ) == NULL )
     {
          printf ( "Unable to allocate memory for RIGHT function\n" );
          return ( NULL );
     }

/*--------------------------------------------------------------------*/
/* Use different algorithms, depending on whether the number of char- */
/* acters desired from the right side of the input string is greater  */
/* than, less than, or equal to the number of chars in the string.    */
/*                                                                    */
/* If length is less than number of characters desired, do internal   */
/* write of entire input string, reverse it, append the required      */
/* number of blanks to pad to desired length, then reverse it again   */
/* and append a null character.                                       */
/*--------------------------------------------------------------------*/

     if ( length < number )
     {
          sprintf ( temp_string, "%.*s", number, instring );
          strrev ( temp_string );
          strcat ( temp_string, copies ( " ", number-length ) );
          strrev ( temp_string );
          strcat ( temp_string, "\0" );
     }

/*--------------------------------------------------------------------*/
/* If equal, simply copy the input string to the output string.       */
/*--------------------------------------------------------------------*/


     else if ( length == number )
          strcpy ( temp_string, instring );

/*--------------------------------------------------------------------*/
/* If number of characters desired is less than the length of the     */
/* input string, allocate temporary work space on the stack, copy the */
/* input string into this temporary string, reverse it, do internal   */
/* write of desired number of characters, reverse it again, and       */
/* append a null character.                                           */
/*--------------------------------------------------------------------*/

     else if ( length > number )

     {
          copy_string = (char *) alloca ( length + 1 );
          strcpy (copy_string, instring );
          strrev (copy_string );
          sprintf ( temp_string, "%-.*s", number, copy_string );
          strrev (temp_string );
          strcat (temp_string, "\0" );
     }

     return ( temp_string );
}
/**********************************************************************/
/*                                                                    */
/*  Function Name:  STRREV                                            */
/*  Purpose:  Reverse the sequence of characters in a supplied input  */
/*            string.  A pointer to the altered input string is       */
/*            returned.  NULL is returned if memory allocation fails. */
/*  Written By:  Karl Wiegers                                         */
/*  Date Written:  March, 1989                                        */
/*  Usage Syntax:  strrev ( input_string)                             */
/*  External Functions Called:  malloc, printf, strcpy, strlen        */
/*                                                                    */
/**********************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *malloc ();

char *strrev ( instring )
char *instring;

{
     register short i;        /* index variable                            */
     register short j;        /* index variable                            */
     register short length;   /* number of characters in 'instring'        */
     char *tempstr;           /* temporary work space                      */

/*--------------------------------------------------------------------*/
/*    attempt to allocate temporary space in which to construct the   */
/*    reversed string, including the trailing null character          */
/*--------------------------------------------------------------------*/
```

```
     length = strlen ( instring );
     if ( ( tempstr = (char *) malloc ( length + 1 ) ) == NULL )
     {
          printf ( "Unable to allocate memory for STRREV\n" );
          return ( NULL );
     }

/*-------------------------------------------------------------------*/
/*  start at the end of the input string and work backward; copy each */
/*  character into the next position of the temporary string; append  */
/*  a null character at the end of the temporary string, then copy    */
/*  the temporary string to the input string and return the pointer   */
/*-------------------------------------------------------------------*/

     for ( i = 0, j = length; j > 0; i++ )
          tempstr[i] = instring[--j];
     tempstr[++i] = '\0';
     strcpy ( instring, tempstr );

     return ( instring );
}
/*******************************************************************/
/*                                                                   */
/*  Function Name:  WORDS                                            */
/*  Purpose:  Return a short integer whose value is the number of   */
/*            blank-delimited words contained in a supplied input    */
/*            string.                                                */
/*  Written By:  Karl Wiegers                                        */
/*  Date Written:  March, 1989                                       */
/*  Usage Syntax:  words ( input_string )                            */
/*  External Functions Called:  strlen                               */
/*                                                                   */
/*******************************************************************/

#include <strings.h>

/*-------------------------------------------------------------------*/
/*  define logical YES to integer 1 and NO to 0; this code makes sure */
/*  we don't try to redefine YES and NO if they already have been set */
/*-------------------------------------------------------------------*/


#ifndef  YES
#define  YES   1
#endif

#ifndef  NO
#define  NO    0
#endif

short words ( instring )
char *instring;

{
     register short num_words = 0;  /* counts # of words in 'instring' */
     register short inword    = NO; /* flag for whether current char   */
                                    /* is in a word (YES) or not (NO)  */

/*-------------------------------------------------------------------*/
/* If the length of the input string is zero, it contains no words.   */
/* Otherwise, the basic idea is to scan each character in the string. */
/* The variable 'inword' is set to YES if the current character is    */
/* not blank, NO if it is blank.  Each time we reach a blank char,    */
/* the 'num_words' counter is incremented, except that multiple       */
/* blanks are treated as a single blank for word separation purposes. */
/* If we're in a word when the terminating null character is reaced,  */
/* increment 'num_words' one more time.                               */
/*-------------------------------------------------------------------*/

     if ( strlen ( instring ) > 0 )
     {
          while ( *instring != '\0' )
          {
               if ( ( *instring == ' ' ) && ( inword == YES ) )
               {
                    num_words++;
                    inword = NO;
               }
               else if ( ( *instring != ' ' ) && ( inword == NO ) )
                    inword = YES;
               instring++;
          }
     }
     if ( inword == YES )
          num_words++;
     return ( num_words );
}
/***********************************************************************/
/*                                                                     */
/*  Function Name:  WORD                                               */
/*  Purpose:  Return a pointer to a specific word in a string of       */
/*            blank-delimited words.  NULL is returned if the          */
/*            attempt to allocate memory fails, if the requested       */
/*            word position is less than 1, or if the requested        */
/*            word position is greater than the number of words        */
/*            in the string.                                           */
/*  Written by:  Karl Wiegers                                          */
/*  Date Written:  March 1989                                          */
/*  Syntax:  word ( input_string, desired_word_position )              */
/*  External Functions Called:  malloc, printf, words,strlen           */
/*                                                                     */
/***********************************************************************/

#include <stdio.h>
#include <strings.h>

#ifndef      YES
#define      YES      1
#define      NO     0
#endif

extern short words ();

extern char *malloc ();

char *word (instring, word_pos)

char  *instring;
short word_pos;

{
     register short word_ctr = 0;  /* counts # of words scanned so far */
     register short inword   = NO; /* flag for whether current char is */
                                   /* in a word (YES) or not (NO)      */
     char  *temp_ptr;              /* temporary work space             */
     static char *word_ptr;        /* points to desired word, returned */

/*-----------------------------------------------------------------*/
/* attempt to allocate space to contain the word being sought in the */
/* input string, including room for the trailing null character    */
/*-----------------------------------------------------------------*/

     free ( word_ptr );
     if ( ( word_ptr = (char *) malloc ( strlen ( instring ) + 1 ) ) == NULL)
     {
          printf ( "Unable to allocate memory for WORD function\n" );
          return ( NULL );
     }

/*--------------------------------------------------------------------*/
/* see if argument value for word position to return is okay;        */
/* if not, return NULL pointer                                        */
/*--------------------------------------------------------------------*/

     if ( word_pos < 1 || ( word_pos > words ( instring )  ) )
          return ( NULL );

/*--------------------------------------------------------------------*/
/* Initialize current character pointer to the beginning of the       */
/* input string.  Then search character by character down the         */
/* string until the terminating null character is hit.  The main     */
/* logic is to see if the current character is a blank or not, and   */
/* whether we are currently in a word (non-blank characters) or       */
/* not.  When we reach the end of a word, increment the word          */
/* counter.  When we reach the beginning of the desired word, scan    */
/* it until the next blank is found, then exit from this big loop     */
/* and append a terminating null character.  At this point, the       */
/* word_ptr is pointing to the beginning of the desired word, so      */
/* return that pointer.                                               */
/*--------------------------------------------------------------------*/

     temp_ptr = word_ptr;

     while ( *instring != '\0' )
     {
          if ( *instring != ' ' )
          {
               if ( inword == YES )
               {
                    if ( word_ctr == word_pos )
```

```
                        *temp_ptr++ = *instring;
                }
                else if ( inword == NO )
                {
                    inword = YES;
                    word_ctr++;
                    if ( word_ctr == word_pos )
                        *temp_ptr++ = *instring;
                }
            }
            else if ( *instring == ' ' )
            {
                if ( inword == YES )
                {
                    if (word_ctr == word_pos )
                        break;
                    else
                        inword = NO;
                }
            }
        instring++;
        }
        *temp_ptr = '\0';
        return ( word_ptr );
    }
/**********************************************************************/
/*                                                                    */
/*  Function Name:  SUBSTR                                            */
/*  Purpose:  Return a pointer to a character string which contains  */
/*            a substring of a supplied input string, beginning at a  */
/*            particular character position and extending for a      */
/*            specified number of characters, with a supplied pad    */
/*            character used to fill trailing blanks.  NULL is       */
/*            returned if memory allocation fails or if starting     */
/*            position does not fall within the string.              */
/*  Written By:  Karl Wiegers                                         */
/*  Date Written:  March, 1989                                        */
/*  Usage Syntax:  substr ( input_string, start_position,             */
/*                       number_of_characters, pad_character )        */
/*  External Functions Called:  strlen, malloc, strcpy, printf,       */
/*                              copies                                */
/*                                                                    */
/**********************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *copies ();
extern char *malloc ();

char *substr ( instring, startpos, length, pad )

char *instring;
short startpos;
short length;
char *pad;

{
    static char *temp_string;      /* pointer to substring to return  */
    register short i = 0;          /* index variable                  */
    register short stop = 0;       /* # of chars to copy from instring */
    register short inlength = 0;   /* # of chars in 'instring'        */

/*--------------------------------------------------------------------*/
/*  make sure starting position is inside the string; also, attempt  */
/*  to allocate space to contain the substring to be returned,       */
/*  including a space for the trailing null character                */
/*--------------------------------------------------------------------*/

    inlength = strlen ( instring );
    if  ( ( startpos > inlength ) || ( startpos < 1 ) )
        return ( NULL );

    free ( temp_string );
    if  ( ( temp_string = (char *) malloc ( length + 1 ) ) == NULL )
    {
        printf ( "Unable to allocate memory for SUBSTR\n" );
        return ( NULL );
    }

/*--------------------------------------------------------------------*/
/*  initialize substring to pad character; if requested length goes  */
/*  past the end of the string, then number of characters to copy    */
/*  ('stop') is number between starting position and the end of the */
/*  string; otherwise, it's the actual number of characters requested */
/*--------------------------------------------------------------------*/
```

```
    strcpy ( temp_string, copies ( pad, length ) );
    if  ( length >  ( inlength - startpos + 1 ) )
         stop = inlength - startpos + 1;
    else
         stop = length;

/*-----------------------------------------------------------------*/
/*  now copy the correct number of characters from the input string   */
/*  into the output substring and return the pointer                  */
/*-----------------------------------------------------------------*/

    for  ( i = 0; i < stop; i++ )
         temp_string[i] = instring[i+startpos-1];

    return ( temp_string );
}
/*******************************************************************/
/*                                                                    */
/*  Function Name:  OVERLAY                                           */
/*  Purpose:  Overlay one character string of specified length,       */
/*            padded to that length with a supplied pad character if  */
/*            necessary, into a second string, beginning at a speci-  */
/*            fied character position.  Returns a pointer to the      */
/*            composite (overlaid) string.  Returns NULL if memory    */
/*            allocation fails.                                       */
/*  Written By:  Karl Wiegers                                         */
/*  Date Written:  March, 1989                                        */
/*  Usage Syntax: overlay ( string_to_overlay, string_to_be_overlaid, */
/*                         character_position_to_overlay,             */
/*                         length_of_overlaying_string, pad_char )    */
/*  External Functions Called:  strlen, malloc, printf, strcpy,       */
/*                              strcat, copies, left, right           */
/*                                                                    */
/*******************************************************************/

#include <stdio.h>
#include <strings.h>

extern char *copies ();
extern char *left ();
extern char *right ();
extern char *malloc ();

char *overlay (new, target, charno, length, pad )
char *new, *target, *pad;
short charno, length;

{
     static char *tstr;       /* points to overlaid string, is returned  */
     char *newnew;            /* holds 'new' string, padded if necessary */
     register short newlen;   /* length of string to be overlaid         */
     register short targlen;  /* length of string being overla           */

     newlen = strlen ( new );
     targlen = strlen ( target );

/*-----------------------------------------------------------------*/
/*  attempt to allocate space to hold padded string being overlaid   */
/*  and final overlaid string that is being constructed              */
/*-----------------------------------------------------------------*/

     if ( (tstr = (char *) malloc ( newlen + targlen + 1 ) ) == NULL )
     {
          printf ( "Unable to allocate memory for OVERLAY\n" );
          return ( NULL );
     }

     if ( ( newnew = (char *) malloc ( strlen ( new ) + 1 ) ) == NULL )
     {
          printf ( "Unable to allocate memory for OVERLAY\n" );
          return ( NULL );
     }

/*-----------------------------------------------------------------*/
/*  first handle different values for argument 'length'; if 0, then  */
/*  set it to the length of the 'new' string, and copy 'new' into    */
/*  temporary buffer 'newnew'                                        */
/*-----------------------------------------------------------------*/
```

```
        if ( length == 0 )
        {
             length = newlen;
             strcpy ( newnew, new );
        }

   /*---------------------------------------------------------------*/
   /*  if requested length is greater than length of 'new', copy 'new'  */
   /*  to 'newnew' and pad it to requested length with pad character    */
   /*---------------------------------------------------------------*/

        else if ( length > newlen )
        {
             strcpy ( newnew, new );
             strcat ( newnew, copies ( pad,  ( length - newlen ) ) );
        }

   /*---------------------------------------------------------------*/
   /*  if requeste length is shorter than length of 'new', just keep the */
   /*  desired number of characters from the left side of 'new'         */
   /*---------------------------------------------------------------*/

        else if ( length < newlen )
             strcpy ( newnew, left ( new, length ) );

   /*---------------------------------------------------------------*/
   /*  otherwise, 'length' matches the actual length of 'new', so just  */
   /*  copy the 'new' string into 'newnew' work space                   */
   /*---------------------------------------------------------------*/

        else
             strcpy ( newnew, new );


   /*---------------------------------------------------------------*/
   /* now build the output string 'tstr'; if the overlay character pos  */
   /* is beyond the end of the string, copy the entire target into the  */
   /* string 'tstr' and pad the end with 'pad' characters up to the     */
   /* character position where the 'new' string will be appended        */
   /*---------------------------------------------------------------*/

        if ( charno > targlen )
        {
             strcpy ( tstr, target );
             strcat ( tstr, copies ( pad, charno - targlen - 1 ) );
        }

   /*---------------------------------------------------------------*/
   /* if the 'new' string is being overlaid at char #1 of the target,   */
   /* we won't be using the left part of 'target', so initialize 'tstr' */
   /* to nulls                                                          */
   /*---------------------------------------------------------------*/

        else if ( charno == 1 )
             strcpy ( tstr, "" );

   /*---------------------------------------------------------------*/
   /*  otherise, copy into 'tstr' the leftmost characters from 'target' */
   /*  up to but not including the char where the overlay is to begin   */
   /*---------------------------------------------------------------*/

        else
             strcpy ( tstr, left ( target, charno - 1 ) );

   /*---------------------------------------------------------------*/
   /* now concatenate 'newnew' onto 'tstr' and add on the right part of */
   /* the original 'target', if it hasn't been overwritten and if we    */
   /* aren't appending 'newnew' onto the very end of 'target'           */
   /*---------------------------------------------------------------*/

        strcat ( tstr, newnew );
        if  ( targlen - charno - length > 0 )
             strcat ( tstr, right ( target, targlen - charno - length + 1 ) );

        free ( newnew );
        return ( tstr );
   }
```

END

HyperFont

**Michtron
576 South Telegraph
Pontiac, MI 48053
(313) 334-5700
$49.95, Color or monochrome**

Reviewed by Ian Chadwick

***HyperFont* is the first attempt to escape from** the wearisome world of bit-mapped fonts by offering a designer and editor to create vector fonts from which GEM fonts can be generated at the required sizes, at any resolution from 50 by 50 to 999 by 999 DPI, from six to 999 points. While no programs (currently) use *Hyperfont's* vector output (programs like *DynaCADD* and *Calamus* use their own vector fonts and commercial Compugraphic fonts), any GDOS application can use the bit-mapped fonts created by *HyperFont*.

The immediate advantage of *HyperFont* is that, once created, a single vector font can be used to generate endless numbers of pixel fonts from the same template by simply specifying new resolutions and sizes. These generations are far truer to the original than those fonts built by algorithms, which manipulate existing pixel fonts into the desired point size.

Working like a mini-CADD program, *Hyperfont*'s tools are far more suited to font design than the pixel-based font editors, with real circles and curves, rather than approximations of them. Unfortunately, the tools are not sufficient for the job intended, and the 56-page manual (including five blank pages) is wholly inadequate—a Michtron trademark.

First and foremost, there is no Undo function. If you make a mistake—move a line by accident, bend a curve the wrong way, rotate something incorrectly—you have to restore it manually, if possible. Pressing Undo only sends you to character zero. Although the manual fails to tell you, pressing any key or key combination displays the character associated with that key; it doesn't perform the function you might expect of it. So Delete, Help, Clr/Home and so on don't act as intended—or as needed!

Since there are already two character-selection features—one on the icon pad, the other in the Edit menu—this keyboard selection method is overkill. However, although the current character is highlighted in the selection manu, there is no way to tell which characters have been done, or even how many have been done. And there are no character numbers present with the character display for reference.

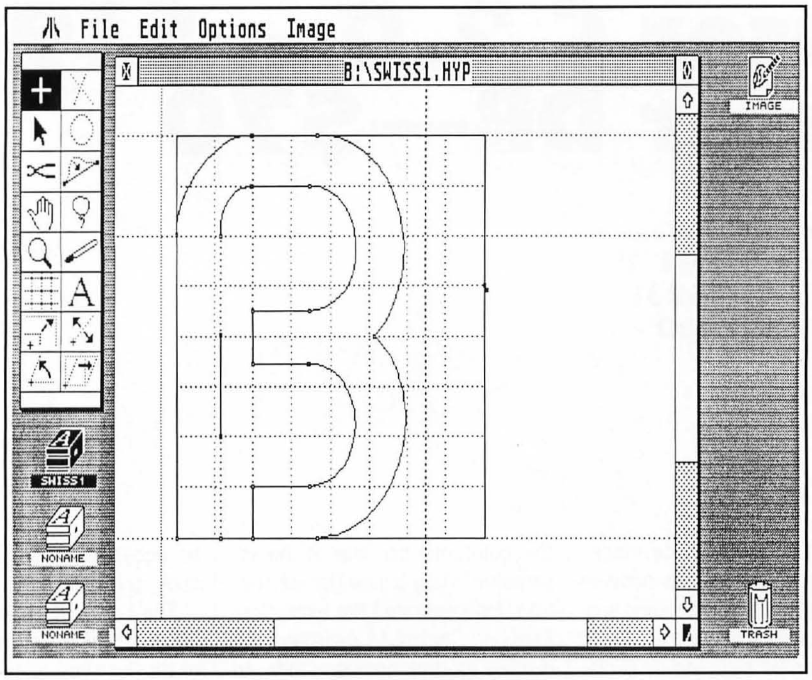
There is no indication of the size of the font in terms of relative points on the drawing area. There is no display of X and Y coordinates to guide you, so absolute positioning of lines is impossible. There is a square of solid lines, which one assumes represents character borders. You can, however, draw outside it with impunity, although you can't generate a font beyond the border. You cannot change the size of this boundary square, although it only occupies about 20% of the full drawing area.

Although there is a grid display, there is no snap-to-grid to aid drawing. This is awkward, since unconnected lines produce bizarre effects in the "show character" display box. It's not mentioned in the manual (like so much else), but you quickly learn that if endpoints of lines are unconnected, problems result. The maximum grid size is 20 by 20, far too coarse for detailed work.

There is a "pliers" tool, described in the manual as:

"The (dis) connecting tool: You can connect adjacent lines with this tool. This way the shape will run smoothly from one line to another, and dragging one line will move the other. Again, it's much easier to try things than to describe them. It's a good idea to study the demofonts for the effects of connecting and disconnecting lines."

This gives you a good idea of the problems associated with the manual and the capacity of the instructions. That's all you get for this tool, a suggestion to experiment with the demofonts (sic.). No



explanation of *how* the tool operates. I was unable to make or do anything constructive.

Another example comes from the same page, in the description of the pointer:

"Double-click on any endpoint with this tool to display a pop-up menu with several options, which will be described later."

Needless to say, these options aren't described *anywhere*, later or earlier. The pointer is one of the most important tools; it performs several tasks, including bending lines into Bezier curves. What's a Bezier curve? The manual doesn't tell you that, either.

The pointer is also used to stretch and shrink a line, but because it wants to create curves all the time, it requires use of the "straight-line" tool to correct it.

The manual also neglects to mention that the "magnify" tool also produces a pop-up menu when you double-click (options of which, again, remain undocumented).

The manual briefly mentions "six moveable lines," vaguely described as the "half line," "top, descender and bottom line," the "kerning information line" and the two "dotted vertical lines," which "specify the current character width in proportional fonts." That's seven lines by my count. They don't use standard typographic terminology, but you can guess at the intent. It appears from the screen that one line is missing. Since there are no labeled pictures, it's hard to figure which one, but I assume it's the "half line," better known as the "x-line" or "x-height line." No relevant information is given as to how to use the lines, what they're for or why. Worse, each line must be set individually in each character (like the grid), an impossible task without some form of snap or X-Y information.

As for the important concepts of kerning and proportional fonts, forget it. Like so much else, you're left in the dark.

The "demofonts" provided are all incomplete in some way; none provides a full 256-character set. SWISS1 is the best, with all the ASCII letters, numbers and punctuation except the ("@") sign available. It's also the most primitive and least interesting of the lot. The other two sets are in considerably worse states of incompletion, missing many ASCII characters. The supply of demo material is far from generous and nowhere near instructive.

The CUT command in the Edit menu doesn't cut, it copies. And the manual doesn't tell you that if you hold Shift while selecting lines with the pointer tool, you can select more than one line.

The "sizer" tool is another bizarre item. The two lines of explanation fail to demystify its effect. Ninety-nine percent of the time it erases my carefully constructed character, leaving me with a single dot. The effect, of course, can't be undone.

Despite the manual's assurance that the lasso tool can be used to "capture character parts," it only captures whole lines within its boundaries, not parts thereof.

Rotate and Slant are graphic: They don't provide any textual indication of the angle, so there's no precision involved. Since you also can't rotate or slant an entire character set at once, it means guesstimating the degree for each character and hoping you're correct. Good luck.

Rotate, Slant and Mirror, like everything else, are permanent and can't be undone.

You can load a GEM .IMG file and use a part of it as a background to trace a character over. This is useful to a certain degree when using, say, scanned images of characters as templates for your own. But if you change characters, the background image is lost and must be recopied back to the character each time—a real pain, since there is no way to reposition the background once you've copied it over to the character editor. There are no sample images for you to experiment with, either. It would have been better had they also allowed users to load *Neochrome* and *DEGAS* format files, since these two are probably the most popular graphic file formats around.

Every time I tried to generate a GEM file, I got a disk I/O error message (on my good hard disk), even when using the supplied "demofonts." Afterward, the "Now generating character:" box stayed on the screen, against my character (no proper redraw). It turned out that the program uses a default directory that didn't exist on my drive, called /FONTS. This cannot be changed in the dialog box, so I had to quit, create the folder, then reload *HyperFont*. So much for user-friendliness!

There are no tutorials or design tips. There is no technical information, discussion of error messages or description of the file formats. There isn't an adequate index. Neither are there suggestions for creating or editing fonts or even a bibliography of recommended texts. If you aren't intimately familiar with GDOS and the ASSIGN.SYS file, you'll find the description of how to install your GEM fonts into it far too vague and obfuscatory to be of real value.

The net result is a great idea minimally realized. It has some nice touches, a few good approaches and is a generally well-designed desktop. It is, however hamstrung by inadequate tools, no Undo feature, a bargain-basement manual and sloppy samples. Back to the drawing board, Michtron: Until these problems are corrected, it's not worth the time or the money.■

***Ian Chadwick** is a Canadian freelance writer and editor whose hobbies include writing about, reading about the exploring the use and abuse of the English language. He is also an amateur paleontologist, naturalist and carpenter.*

Laser C 2.0— $195 Laser DB—$70

Megamax, Inc.
P.O. Box 31294
Dallas, TX 75231
(214) 699-7400

Reviewed by Darek Mihocka

When I sent my check to Megamax, Inc., for the Laser C update, I was a bit skeptical about what I would receive. After all, Megamax was asking $50 for the upgrade, which was more than double the upgrade cost for the first version of Laser C, and so far, all of their software has had some pretty annoying bugs. I've already lost track of how many phone calls I've made to Megamax over the last year to report bugs. But after installing the new disks on my hard disk and playing around with it, I was satisfied that it was money well spent. Here's why.

Laser DB

The shining star of the Laser C 2.0 package is the new debugger, Laser DB. For the first time, Megamax offers source-level debugging of C code and symbolic debugging of assembler code and code not compiled by Laser C.

When I first saw Laser DB, I wondered why they hadn't just gone ahead and called it Codeview Junior. Being a regular user of Microsoft C and Codeview, I was delighted to finally see a product on the ST that is almost as good. I say almost because a few features are missing from Laser DB.

To run Laser DB, you can either click on the Debug menu entry in the Laser C 2.0 shell or run it from the desktop. In either case, a dialog pops up, prompting for the name of the target program to debug, tne command line (if any) and the debugging modes: source, assembly or both.

Figure 1 shows a typical Laser DB screen in source-code mode. It is composed of four windows: Watchpoint, Code, Expressions and the Register/Stack window. Each window can be selected by clicking on it with the mouse or by pressing an appropriate key. The first three windows can also be resized by clicking the top of the window with the mouse and dragging. The Register/Stack window can be hidden if you wish to debug only at the source level. Clicking on it changes the display between a register dump and a stack dump. Figure 2 shows the assembly-level display with a register dump.

The power of Laser DB doesn't become apparent until a piece of compiled code starts to mysteriously crash and put cherry bombs on the screen. Usually, panic sets in. I used to load up my code with *printf()* calls, but this is not always helpful. Having a memory-resident debugger, such as the excellent *Templemon* program loaded at all times helps by showing the exact machine-language instruction that generated the exception. But such utilities can't display symbols or help you locate the line of source code that is at fault.

With Laser DB, any time the 68000 generates an unwanted exception, such as a bus trap error or illegal instruction error, the 6800 registers, as well as the offending address and other related information, are displayed in the register window. the 68000 registers are displayed in the register window, as well as the offending address and other related information. At this point, one can look at the assembler listing to see the instruction that generated the exception. Pressing Control-M switches the display to the source mode to show you the exact line of source code that needs to be fixed.

Laser DB uses a separate screen for its display, thus allowing it to debug any GEM-based program. A Flip command switches between the two screens. Laser DB also grabs control of all the important system vectors, including the Alt-Help vector, making it difficult (but not impossible) to generate screen dumps. In fact, Alt-Help is used as the break key.

If that was all the Laser DB did, it would hardly be worth its price, considering the availability of public-domain debuggers. But, like Codeview, it offers more.

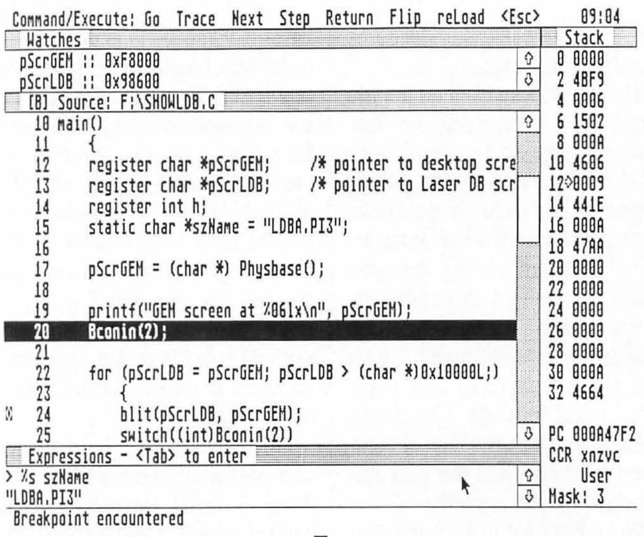
The watchpoint window can display up to ten watchpoints. A watchpoint is an expression that is evaluated each time Laser DB is in control. For example, Figure 1 shows part of the program I used to generate the Laser DB screen shots. The Watch window is displaying the value of two pointers as the program searches through memory looking for Laser DB's screen. In the Expression window on the bottom, I asked it to show me the filename to which the screen shot is being saved. Both the Watchpoint and Expression windows support C-like expressions, allowing evaluations of both simple integers and complex pointer expressions. There are some predefined symbols that are used to access the 68000 registers, code and data segments.

The Laser DB manual is small, about 80 pages, but describes all of the features quite well. Single keystrokes access most of the functions, and there is a menu bar;that can be accessed with the mouse. The menu bar is not a standard GEM menu bar; it is similar to the type of menus seen in programs like *Lotus 1-2-3*. Pressing the first letter of a menu entry selects that menu entry, and the escape key always takes you to the top menu.

Search functions are also provided. The search can be for a particular string in the source code, or a label in the object code. This feature can be used when debugging a large program.

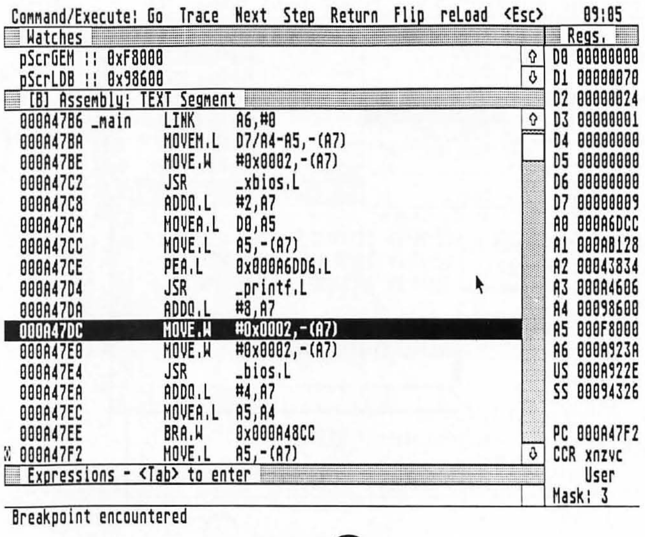
Laser DB also has a Journal window, which displays the last 100 lines of code executed. This can be useful when trying to determine the cause of erratic program behavior, although it does not log the actual bus activity. A Calls window displays a traceback of the stack, showing all function calls and their parameters.

A standard feature of any debugger is the ability to set breakpoints. Laser DB is no different. To set a breakpoint, simply scroll the Code window to a particular line of source code *and* click on that line with the mouse or press the asterisk on the keyboard. There are fancier variations on that theme, such as optional breakpoints. These types of breakpoints interrupt the program when they are reached only if some ex-



```
Command/Execute: Go  Trace  Next  Step  Return  Flip  reLoad  <Esc>     09:04
 Watches                                                              Stack
 pScrGEM :: 0xF8000                                                 0 0000
 pScrLDB :: 0x98600                                                 2 4BF9
 [B] Source: F:\SHOWLDB.C                                           4 0006
 main()                                                             6 1502
     {                                                              8 000A
     register char *pScrGEM;     /* pointer to desktop scre        10 4606
     register char *pScrLDB;     /* pointer to Laser DB scr        12⇨0009
     register int h;                                               14 441E
     static char *szName = "LDBA.PI3";                             16 000A
                                                                   18 47AA
     pScrGEM = (char *) Physbase();                                20 0000
                                                                   22 0000
     printf("GEM screen at %061x\n", pScrGEM);                     24 0000
     Bconin(2);                                                    26 0000
                                                                   28 0000
     for (pScrLDB = pScrGEM; pScrLDB > (char *)0x10000L;)          30 000A
         {                                                         32 4664
         blit(pScrLDB, pScrGEM);
         switch((int)Bconin(2))                                    PC 000A47F2
 Expressions - <Tab> to enter                                      CCR xnzvc
> %s szName                                                           User
"LDBA.PI3"                                                         Mask: 3
Breakpoint encountered
```

FIGURE 1



```
Command/Execute: Go  Trace  Next  Step  Return  Flip  reLoad  <Esc>     09:05
 Watches                                                           Regs.
 pScrGEM :: 0xF8000                                              D0 00000000
 pScrLDB :: 0x98600                                              D1 00000070
 [B] Assembly: TEXT Segment                                      D2 00000024
 000A47B6 _main    LINK     A6,#0                                D3 00000001
 000A47BA          MOVEM.L  D7/A4-A5,-(A7)                       D4 00000000
 000A47BE          MOVE.W   #0x0002,-(A7)                        D5 00000000
 000A47C2          JSR      _xbios.L                             D6 00000000
 000A47C8          ADDQ.L   #2,A7                                D7 00000009
 000A47CA          MOVEA.L  D0,A5                                A0 000A6DCC
 000A47CC          MOVE.L   A5,-(A7)                             A1 000AB128
 000A47CE          PEA.L    0x000A6DD6.L                         A2 00043834
 000A47D4          JSR      _printf.L                            A3 000A4606
 000A47DA          ADDQ.L   #8,A7                                A4 00098600
 000A47DC          MOVE.W   #0x0002,-(A7)                        A5 000F8000
 000A47E0          MOVE.W   #0x0002,-(A7)                        A6 000A923A
 000A47E4          JSR      _bios.L                              US 000A922E
 000A47EA          ADDQ.L   #4,A7                                SS 00094326
 000A47EC          MOVEA.L  A5,A4
 000A47EE          BRA.W    0x000A48CC                           PC 000A47F2
 000A47F2          MOVE.L   A5,-(A7)                             CCR xnzvc
 Expressions - <Tab> to enter                                       User
                                                                 Mask: 3
Breakpoint encountered
```

FIGURE 2

pression is true. For example, only break if the return value of a BIOS call is nonzero.

Finally, there are several ways to debug the target program. You can select the Go command, which runs your program until it either crashes or reaches a breakpoint, or until you press the break keys (Alt-Help). The Trace command does the same thing, except it continually single-steps through the program, updating the Watch and Register/Stack windows after each instruction. The Step command single-steps under user control and can be used to step through either the source code or assembly code. You can choose to step through or step over subroutine calls.

Compared with Megamax's previous debugger, Laser DB is a fantastic improvement. I was pleasantly surprised to find no major bugs in Laser DB either. The only major flaw is its inability to log sessions to disk or to remotely debug through the serial port. Remote debugging would be required in situations where memory is so tight that Laser DB would not fit along with the target program. In that case, only a small portion of the debugger remains in memory, while the rest runs on a separate machine. Atari's developer's kit provides such a debugger.

Debugging a target program not compiled by Laser C is possible, but only at the assembler level. If a symbol table is included in the executable file, the symbols can be used; otherwise, you debug in hex! Fortunately, most other packages can generate symbol tables.

Laser C 2.0

Laser C 2.0 is twice as good now, right? No, not really. As it turns out, Laser C 2.0 offers only a handful of new features. The desktop is similar to the original in that it offers a built-in editor capable of editing up to four text files simultaneously; a disk cache for speeding up all disk operations, an STDIO output window (which doubles as a command-line shell); and a set of drop-down menu entries for calling up the compiler, linker, debugger and other programs. For a detailed look at the original Laser C, see my review in the November '88 ST-LOG.

Laser C 2.0 will run on a 512K machine, but its powers are severely limited by the lack of memory: One cannot take advantage of having memory-resident tools or edit large source-code files. I highly recommend using a Mega ST with Laser C, but if you're like me, a 1040ST with a fast hard disk will do fine.

The editor has been enhanced with a few new options, such as selecting the entire buffer. This is handy for clearing out the STDIO window in a hurry. Figure 3 shows the Laser C 2.0 desktop with four source files being edited, plus the STDIO window.

The editor uses a dynamic memory-allocation scheme that is closely tied in with the disk cache. This is both a blessing and a curse. As I mentioned in the first review, the disk cache can't be disabled. This is fine if you have plenty of memory, as this allows you to store all your tools, source code, object code and executable code in memory at once.

It results in extremely fast compiles, but with a price. As memory starts to fill up, especially when editing large source-code files, the cache must start making decisions about which files will be kept in memory and which will be flushed to disk. On a 1040ST, this can happen quickly, and performance drops off fast. A compile-and-link operation that takes 20 seconds to complete with an empty cache takes 50 seconds the second time around. This is more than twice as slow.

In all fairness to Laser C, the problems don't appear to be totally its fault. When I tried running Laser C with TOS 1.4, the performance increased to almost empty cache levels. Unfortunately, 99.9% of ST users do not yet have TOS 1.4, so a cache-disable option would still be desirable.

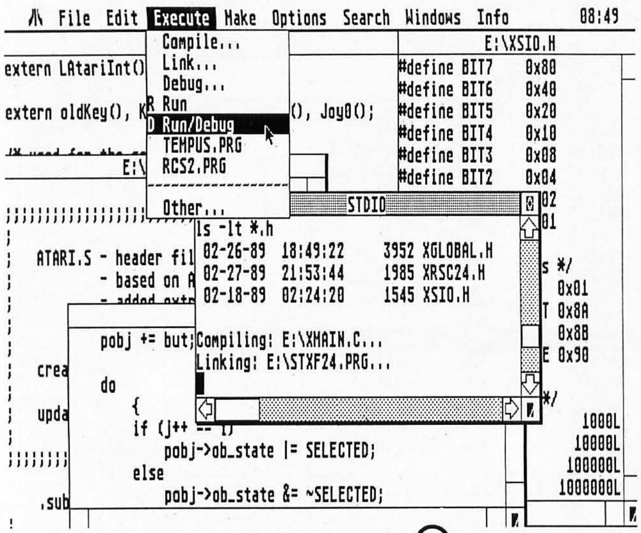
The editor now has the ability to print the buffers, and a printer setup feature is provided.

The compiler and linker have changed very little, except to support Laser DB. With the Z flag enabled, the compiler and linker will add extra debugging information to the executable file. This extra information can easily double the size of the executable module and should only be used during development. More on the debugger later.

The compiler now fixes some of the code-generation errors in the inline assembler. The new linker also fixed a bug I found earlier. I once accidentally generated an object file that contained 65K of code. When I tried to link it with other modules, the linker bombed! As it turns out, the old linker couldn't handle object files greater than 64K.

Unfortunately, the compiler still does not support function prototyping, a feature found on most DOS and UNIX compilers and an essential part of the ANSI C standard. Hopefully, future versions of Laser C will support this, since it is an easy and painless way of avoiding program crashes caused by careless coding.

The resource-construction program is now up to Version 3.0, although I fail to see much of a difference between it and previous versions. The one obvious addition is the ability to read and write the new .DFN file format used with Atari's Resource Construction Set 2.0, as well as the older and more common .DEF files used with other resource editors. In fact, Megamax's RCP is the only one I know of that can easily convert from .DEF files to .DFN and back. Fig-



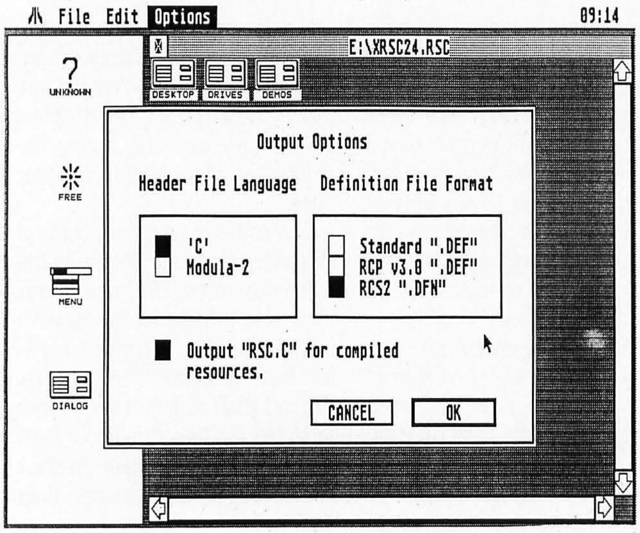
```
E:\XSIO.H
#define BIT7     0x80
#define BIT6     0x40
#define BIT5     0x20
#define BIT4     0x10
#define BIT3     0x08
#define BIT2     0x04
```

```
ls -lt *.h
 02-26-89  18:49:22      3952 XGLOBAL.H
 02-27-89  21:53:44      1985 XRSC24.H
 02-18-89  02:24:20      1545 XSIO.H

Compiling: E:\XMAIN.C...
Linking: E:\STXF24.PRG...
```

```
    pobj += but;
    do
        {
        if (j++ == 1)
            pobj->ob_state |= SELECTED;
        else
            pobj->ob_state &= ~SELECTED;
```

FIGURE 3



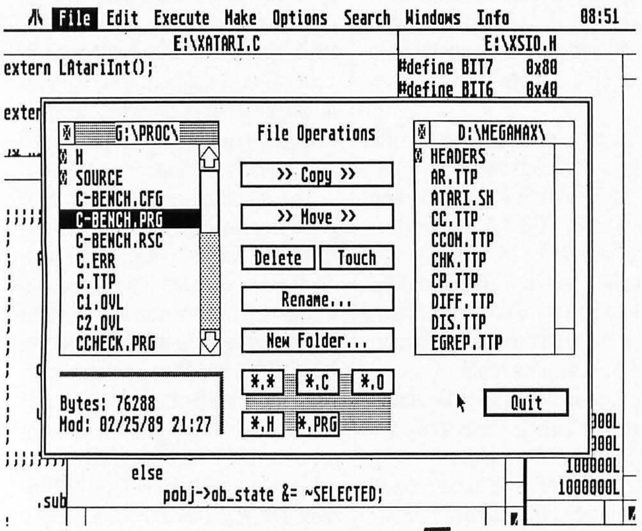
Output Options

Header File Language: ■ 'C' □ Modula-2

Definition File Format: □ Standard ".DEF" □ RCP v3.0 ".DEF" ■ RCS2 ".DFN"

■ Output "RSC.C" for compiled resources.

FIGURE 4



```
G:\PROC\              D:\MEGAMAX\
 H                     HEADERS
 SOURCE                AR.TTP
 C-BENCH.CFG           ATARI.SH
 C-BENCH.PRG           CC.TTP
 C-BENCH.RSC           CCOM.TTP
 C.ERR                 CHK.TTP
 C.TTP                 CP.TTP
 C1.OVL                DIFF.TTP
 C2.OVL                DIS.TTP
 CCHECK.PRG            EGREP.TTP

Bytes: 76288
Mod: 02/25/89 21:27
```

FIGURE 5

ure 4 shows the Laser Resource Construction Program.

The RCP is also claimed to have the ability to generate C code from resource files. So far, only Atari's RCS 2.0 has this ability. Unfortunately, the "code" it generates is nothing more than a static array of integers, rather than the neatly commented structures and arrays I was expecting. Laser C 2.0 provides a *mrscr__load()* library function to replace the usual *rscr__load()* function. This works nicely and makes the task of compiling resources into the executable code almost invisible.

One other nice feature of Laser C 2.0, which was around before, is the Disk Ops dialog, shown in Figure 5. It provides a quick method of performing disk operations without having to exit to the ST desktop. *Deskcart* and the *Universal Item Selector II* provide similar features, but if you don't already have these programs, you don't need them.

In a nutshell, if you already use Laser C 1.0, don't expect Version 2.0 to add much. But do send in for the update so you can use Laser DB.

As Good as You-Know-Who?

One does not have to look far to see the direction in which C compilers are going. Several packages for the ST, including Turbo C, Prospero C and Laser C are emulating the integrated environment made famous in the DOS world by Borland's Turbo Pascal and Turbo C. Laser C was the first for the ST, and with features like the Laser DB debugger, it will continue to lead the pack.

Compared to the DOS packages, like Turbo C and Quick C, the Laser C 2.0 package with Laser DB still falls short in a few areas. The flow from editing to compiling to debugging is not as smooth as in the DOS packages. Laser C still has to load each module separately, either from disk or from the disk cache, and Laser DB is, for the most part, a separate entity. For example, in Microsoft's Quick C, one can single-step through a program and edit the source code as soon as a change is required. With Laser C, you have to exit Laser DB, make the changes, recompile, then run Laser DB again. Considering the relatively small size of Laser DB (about 100K) compared to the Laser DB shell (160K) and the amount of code that must be common to both, I don't see why the debugging features of Laser DB couldn't be integrated into the shell. Of course, a stand-alone version of Laser DB is still necessary for those situations where memory is tight.

I recommend Laser C 2.0 to anyone interested in developing software on the ST. However, I'm not jumping up and down with joy and delight as I did when I first reviewed Laser C, simply because the new enhancements do not warrant the 2.0 name. Until Laser C starts to support the ANSI C standard, this latest version is a 1.2 at best, and only because of the improvements in the resource editor.

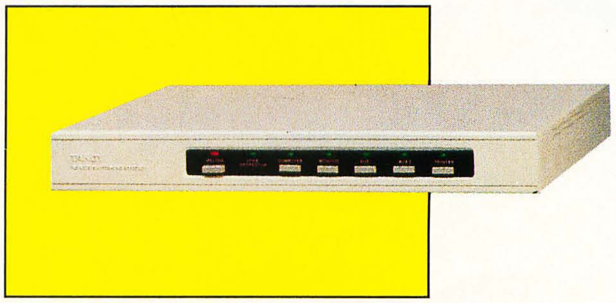
Laser DB is another story, though. Compared to the other debuggers available on the ST, it is clearly the best. Also, its ability to debug not only Laser C code but also code generated by other C packages and assemblers should make it a welcome addition to any developer's library.

The package of Laser C and Laser DB is very complete. It includes an editor, compiler, linker, resource editor, make utility, disassembler, various utilities cloned from UNIX, a command-line interface, a source-level debugger and GEM/VDI and BIOS/GEMDOS documentation. About all that is sorely missing is a good code optimizer, although Laser C's is still one of the fastest.■

***Darek Mihocka** is a computer engineering student at the University of Waterloo. His latest ST project is the improved* ST Xformer, *and he is currently working on image-processing and character-recognition software.*

Tandy Power Switching System

Radio Shack
Cat. no. 26-203
$99.95



Reviewed by Pamela Rice Hahn

I bought my ST for a variety of reasons. While I have never regretted the investment, I did become temporarily frustrated by being required to reach behind disk drives, under and around the monitor and behind the computer itself to turn everything on. A computer is supposed to automate your life, right? There had to be an easier way.

For me, Tandy's Power Switching System (PSS) was the answer. Now I have push-button efficiency whenever I'm ready to power up. Granted, there are less expensive alternatives; however, they don't have the features I found in this unit.

Setup is simple. The computer and appropriate peripherals are plugged into the back of the PSS; the PSS cord is then plugged into your three-wire grounded AC outlet. The outlets at the back of the PSS are labeled according to the corresponding (and equally labeled) controlling switch on the front of the unit. The manual suggests you either leave all unit switches in the on mode and control power to all components when you toggle the master switch, or turn on the master switch and then toggle each unit switch in the appropriate order. I prefer the latter, since acting on the first suggestion could defeat the voltage spike-protection purpose of the PSS. Spike protection is provided to prevent power surges, such as those that can result when a major appliance kicks on.

A red LED indicator lights up when the master switch is on; the green spike protector indicator light remains on as long as the protection circuitry is functioning properly. The unswitched outlet remains on at all times so it can be used for your desk lamp or any other electrical device you keep at your computer-desk area that doesn't require the surge protection.

Filtration

The PSS has three independently filtered sections that prevent power-supply interaction between the computer and peripherals connected in each section. Of the six outlets provided, the printer and aux2 are in one section, the aux1 and monitor are in the next section and the computer outlet is in the final section. A sixth non-filtered but transient-protected outlet is also provided. (It should also be noted that the outlet spacing is sufficient to accommodate AC adapters.)

Spike Protection

While it can't protect the contents of your computer's memory against power failures or brownouts, the PSS does minimize transients and noise generated by other electrical devices. A 15-amp circuit breaker protects against overcurrent catastrophes.

Twist About

While this option isn't used too often, the Power Switching System unit itself provides a swivel base for my monitor, which sits on top of the unit, allowing me to adjust the viewing angle when I want to demonstrate a program for someone seated at my side.

Conclusion

If you're having to reach around and between various pieces of equipment in order to power up your system, you might find that the Tandy Power Switching System is an invaluable companion for your ST. ■

***Pamela Rice Hahn** has been an Atari enthusiast for five years. An active member of MVACE and the former editor of* The TCAUC Newsletter, *she continues to dispatch at a small-town police department to help support her family's computer habit, all the while dreaming of someday fulfilling her desire to become a full-time freelance writer.*

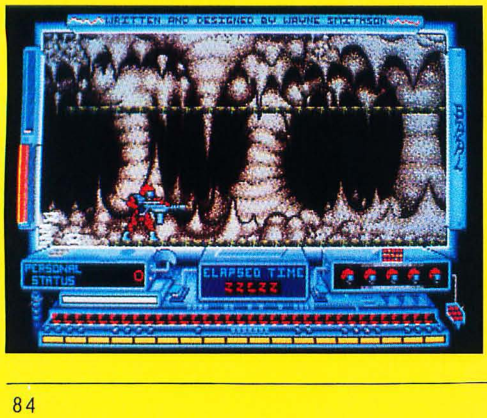
REVIEW

GAME SHELF

BAAL

Psygnosis, Ltd.
Distributed by Computer Software Service
2150 Executive Drive
Addison, IL 60101
(800) 669-4912
$29.95, Color only

Reviewed by Frank Eva



Baal **(pronounced "bale") could be alternatively titled** *Obliterator Meets Barbarian.* There are so many similarities between *Baal* and these two previous Psygnosis releases that a comparison becomes inevitable. Owners of these previous products will want to know what advantage there might be to adding *Baal* to their libraries. What does *Baal* have that *Obliterator* and *Barbarian* don't have?

In *Baal*, the player takes the role of Time Warrior, sent into the underground world of the evil Baal and his grotesque beasties. The Time Warrior is reminiscent of the *Obliterator* character, and the monsters he faces resemble those from *Barbarian*.

The player must enter Baal's domain and search for and abscond with pieces of a doomsday weapon that Baal intends to unleash upon unsuspecting humanity. Along the way, all sorts of evil creatures attempt to put an end to the Time Warrior's quest.

The graphics are nicely detailed. Several shades of the same color paint a haunting underworld empire. Like *Donkey Kong*, platforms and ladders are used to move around inside this domain. The background scrolls smoothly, but this is probably due to the fact that there are few animated objects for the program to keep track of.

The Time Warrior is clearly depicted as well as accurately and amusingly animated. Because the character's laser blaster is large and heavy, it must be carried in both arms. Consequently, he bobs up and down with each step, as if the burden were weighing him down.

Each creature is interesting and has its own method of thwarting the Time Warrior's progress: A flying frog, for example, drops green globules that adversely affect the personal energy status of the Time Warrior. Other adversaries are more difficult to outwit because they present more of a barrier. Since the Time Warrior cannot aim his blaster (he can only shoot horizontally), the shorter creatures can be hit only from a "down" ladder.

The program does support eight-way scrolling, but at first glance this graphic function is not readily apparent. Actually, while the Time Warrior is on foot, he can only move left/right or up/down. The diagonal movements are only possible when he has found a rocket suit (something that is not even hinted at in the documentation). Stopping inside an area bounded by red lines above and below will activate this area's Dr. Who-like gimmickry. A sign slides down that says, "Phone out of order," and when it goes up again, the Time Warrior is ready to rocket off in any of eight directions. If he doesn't find the correct booth in which to park his rocket suit, the energy will expend itself in an explosion, and one life will be lost.

Baal is strictly a one-player game. A joystick is used for input. While running, the diagonal positions of the stick relate to the Time Warrior's ability to somersault over dangerous obstacles, like land mines. These movements can be somewhat tricky, but since there seems to be no hurry, a player can take enough time to perform the maneuver without danger. (This may be misleading, since the player's elapsed time is displayed. It's just that the documentation again gives no insight as to the relevance of this status readout!)

The trigger button is required to differentiate a somersault from normal movement. Otherwise, it is used to fire the laser blaster in whichever direction the Time Warrior is currently facing. The player has a limited amount of energy for firing, but this limitation can be overcome by refueling at the few service stations found along the way. While the laser blaster is being refueled, the Time Warrior's personal energy field is also being recharged.

The player will discover, scattered throughout *Baal*'s domain, icons representing rocket fuel. Collecting one of these will qualify the player to use the rocket suit. Fuel icons seem to be more plentiful than laser refueling stations, but they are not always in the right place at the right time.

Several keyboard functions allow further configurations of the Time Warrior. After collecting ammunition packs, pressing the corresponding number key will activate an enhanced level of firepower. The blaster doesn't change; it just becomes more and more deadly. Other key codes allow the area coordinate display to be toggled on and off and to pause or restart the game.

In a sea of other software firms that believe they cannot afford to add goodies to their "generic" titles, it's nice to find that Psygnosis has maintained certain amenities that computer gamers have become accustomed to. For example, the top 50 scores can be saved to disk. Also, there is a rudimentary game-save function. Although it can be activated only at a refueling station, it comes in handy; a real lifesaver!

Finally, *Baal* features three unique domains, 250 screens, many interesting sound effects, an 11-minute musical introduction (you can skip it if you desire to jump right into the fray), 400 traps and more than 100 monsters!

In the final analysis, *Baal* doesn't really break enough new ground to be heartily recommended to owners of *Obliterator* or *Barbarian*. Other potential buyers will appreciate *Baal*'s price and the game's ability to transform its owner into a *Baal* addict! ■

Willow

Reviewed by Peter A. Smith

Mindscape, Inc.
3444 Dundee Road
Northbrook, IL 60062
(312) 480-7667
$42.95, Color only

Remember what life was like before the movie *Star Wars*? Remember when children's clothes, toys and games were based on original ideas, rather than movie characters and plots? These days, it is often the case that the toys are out before the movie, and I suspect the toy-selling potential is carrying more and more weight with Hollywood's decision-makers. This trend in movie paraphernalia has expanded (following the rules of sound entrepreneurship) to adult products, also. I must admit that the ALF slippers I received last Christmas are quite toasty, if not the most macho apparel I own.

But this is a game review, not a commentary on today's disposable society, and the product at hand is *Willow*. I saw the movie and liked it at lot. I was therefore anticipating great things from the game. Unfortunately, *Willow* is a product that lost its direction somewhere along the software path and ended up going nowhere.

Upon opening the package, I was slightly miffed at finding, along with the six pages of docs, a five-page Reference Card, a Registration Card, ads for "The Lucasfilm Fan Club," "The Willow Game" (a board game) and "The Willow Sourcebook." The Pepsi commercial at the beginning of the *Top Gun* videotape has hardened me somewhat to such things, but they still bother me. How much of the $42.95 cost of *Willow* (the computer game) went into producing these slick ads?

Anyway, I forged ahead, inserted Disk A into my Mega 4, turned it on and. . .got dumped to the desktop. Oh well, some software is funny, so I disconnected my hard disk. Still nothing. The program began to load, then *bam!* back to desktop. Luckily for all you review-hungry readers, I have a second system, so I inserted Disk A into my double-sided external drive of my one-megabyte 520 ST. The program began to load. And load. And the drive went "clank!" I frowned. "Clank!" I grimaced. *"Clank!"* I jabbed the power switch. Not being one to give up easily, I switched to an ancient (like, circa 1986) single-sided drive. Expecting the worst, I powered up. Murphy's Law can be used to your advantage. . .the program loaded and I was greeted with an attractive title screen.

Lesson one: Investigation has taught me that I was not alone in having problems getting this program to run. It seems Mindscape has used some sort of extended format that a lot of drives don't like. (I've used all sorts of extended-format schemes with my drives; my experience with the *Willow* disks was the first time I've had any trouble.) So make sure you get *Willow* from a dealer who is aware of the problem and who will refund your money if you can't get it to run.

Assuming you get through the loading process, what can you expect? *Willow* is broken up into seven mini-games that must be completed sequentially. In addition, you may play all but the final subgames individually for practice. The overall control of the game takes the form of a scroll relating the story of Willow. Some of the words on this scroll are highlighted, and clicking on them loads-in the various subgames or starts the complete game. (This is, in my experience, the first ST game to use hypertext as a control device. The AI gurus must be gaining on us.)

Loading each sequence from disk takes a long time and can get annoying. But that's not the problem with *Willow*. The problem is in the design of the game itself. Each of the subgames is dreadfully dull. One, the Daikini Crossroads, consists of pointing at one of two cages and pressing the controller button. If you have guessed right, you may continue. If not, you die. There are no clues to guide you, it is sheer guesswork.

Guesswork is what *Willow* is all about. Of the seven games, four are blind trial and error. For example, the first sequence is called Dungeons and is a first-person maze game. You have the choice of two or more of the following: going straight, to the left, to the right or backwards. Some of the passageways are stairways going up or down, some lead to cells. Should you blunder into a cell, there is no escape—scratch one life and start over. You can't see down any passageway before you select it, and there are no clues as to which of the passages are cells. The game is nothing but a mapping exercise. Later, you will come to a sequence called the Ice Caves, which is similar, aside from the fact that you cannot stop to map (since you're sliding through the caves on a shield).

The Spellcasting sequence offers 13 symbols. You must pick the right three in the right order. Pick one of the wrong ten and lose a life. Pick one of the right three in the wrong order, and you cannot finish without losing a life, although you can continue picking symbols in order to ascertain the correct three for the next try. The Daikini Crossroads has already been mentioned.

The Woods section takes a bit of thought. You run down pathways while trying to avoid Death Dogs and Nockmar Troops. Your only defenses are your magic acorns, which turn your opponents to stone. You can find more acorns if you stray from the path, but on the other hand, you move much more slowly when off the path. The problem with this section is its brevity. In the time you read this paragraph, you could have been out of the woods (or dead).

The Battle sequence is the game's only saving grace. This is a typical sideview swordfight. First, Madmartigan must duck under and jump over a variety of arrows, bombs and spears chucked his way. Then, when he reaches his foe, General Kael, the true fight commences. Controls are fairly simple: jump, duck, move and swing high, medium or low. I enjoyed this segment, but it could not offset the rest of the game. Buy *DeathSword*, an excellent sword-battle game from Epyx, instead.

Finally, there is the last battle. This variation of Spellcasting is an easy puzzle since you must have mastered Spellcasting before you reach it.

And thus ends *Willow*, with little fanfare. You have (assuming you're playing the full game) eight lives in which to get through all of this. The puzzles and mazes don't change from game to game unless you choose to scramble them. So, as you play over and over, you will get farther and farther, assuming you map correctly. Then, once finished, you may choose to scramble everything so you can play again.

It has crossed my mind that this product was intended to be a children's game. It isn't marked as such on the box, and at any rate, I wouldn't recommend it for children. Although the graphics are nice (some are digitized scenes from the movie), I'd guess that the segments would be too frustrating for younger people. For instance, it took 13 steps (looking at my map) to get out of the maze. I'd guess it took 15-20 tries to get through it. I was frustrated by then, especially considering the average one minute and 50 seconds it took between death and restarting. I doubt younger folk would stand for that sort of abuse.

For us oldsters, the obstacles in the various subgames are annoying, yet trivial. I opened the package at about 6:00 p.m. and finished playing at 10:00 p.m. I had no desire to try it again. I am afraid this product was rushed to completion, probably in an attempt to cash in on the *Willow* (the movie) videotape release. Try as I might, I cannot find the silver lining on this cloud. ■

Jug

Reviewed by Frank Eva

**MicroDeal
576 S. Telegraph
Pontiac, MI 48053
(313) 334-8729
$39.95, Color only**

***Jug* is an arcade adventure in which the** player assumes the role of an interactive humanoid composed of titanium fleximetal and other organic materials; kind of like a cross between Robocop and a Transformer.

Your mission as Jug is to seek out and destroy a deadly virus that is causing the brain within the living core of the planet Spiraeus to malfunction. The planet's immune system will attack and attempt to destroy you as if you are an infection.

The action screen is made up of a foreground and background that are individually moved at slightly different speeds (parallax scrolling), effectively simulating a third dimension. The foreground consists of multiple maze screens connected together by multidirectional scrolling and controlled by the movements of the Jug character.

A joystick allows Jug to be moved left or right as he rolls along on wheels like R2D2 of *Star Wars* fame. To make him fly, the joystick is pushed forward. When a piece of equipment or a fuel pod is available, a pull on the stick collects it. The mouse is used only to select game options, which include a one- or two-player (alternating) contest and three difficulty levels. The planet Spiraeus is divided into zones representing gradu-

continued on page 92

Sorcerer Lord

Reviewed by Peter A. Smith

by PSS
Distributed by Datasoft
19808 Nordhoff Place
Chatsworth, CA 91311
$34.95, Color only

It seems that since the beginning of personal computing, there has been a fairly high correlation between computer enthusiasts and wargamers. War games were plentiful on the 8-bit Atari machines, but a funny thing happened on the way to the 16-bit world: Fantasy role-playing games took over. How many real war games are there for the ST? A hard-core wargamer would snarl, "None," at this point. With a few exceptions, the ST owners have been left at base camp, so to speak.

But now there is *Sorcerer Lord.* The game has a fantasy setting, but you'll find no dungeons here, nor will you have to worry over hit points. Stripped of its finery, *Sorcerer Lord* is revealed to be a traditional war game: hexagonal movement, reinforcements, movement points, forced retreats, terrain effects—they're all here. So all is good in the world and we wargamers can rejoice, right? Well, not yet. . . .

The scenario goes something like this: The land of Galanor has been a peaceful place for the past few eons. The fact that the Shadowlands lie nearby no longer bothers anyone because the Old Race that dwells within has kept to itself for as long as history remembers. Over the centuries, a few men have entered the Shadowlands, learned the secrets of sorcery and escaped with that knowledge. These men went on to become Sorcerer Lords.

But now, the old Shadowlord has died and the new Shadowlord is hungry for power. His desire is to control the Rune Rings—ancient standing stones enchanted by the Old Race before man ever made his appearance. He has bred an army of wolf riders and warriors and has unleashed them on an unsuspecting world.

Thus begins your stint as the Sorcerer Lord. You must take command of the combined armies of men, Elven Lords and barbarians in an attempt to stem the tide of darkness that threatens to envelop the land. Okay, before all you armchair generals out there dismiss this game as just another fantasy rehash, let me assure you that the Shadowlord forces are homogenous, while the effect of race on the "good" side is basically terrain benefits (i.e., elves fight better and move more rapidly in the forests than do other races).

The game comes on one single-sided disk and is not copy-protected. Also included are the Player's Guide, a Reference Card for the IBM PC and Compatible/Atari ST/Amiga and a map of the Lands of Galanor.

GEM lovers, be forewarned! As you may have guessed by now, you'll find no windows, no drop-down menus, and you can send your mouse to the cleaners. The entire game is keyboard-controlled (nope, no joystick either). I am curious as to why PSS didn't at least support joystick movement. As it is, you use the number keys (or numeric keypad) to pick up, move and put down your units. All in all, the interface felt IBM-ish to me. Yes, ladies and gentlemen, we have another port on our hands.

The battlefield is two screens wide by three screens high (each screen being about 13 by ten hexagons), is visually appealing and has an invisible hex grid superimposed upon it. The terrain is diverse, from desert to icy wastes. Also included are fortresses, citadels and Rune Rings. The battlefield does not scroll, but instead uses some sort of page flipping to change among the six segments. This can make coordinating attacks across the segment boundaries extremely frustrating. There are strategic maps available, and they will help a bit, once you learn to decipher them.

The game is broken down into turns, and the turns into phases. Each turn represents one day. (Remember that; I'll be getting back to it.) The phases are Shadowlord Movement, Shadowlord Combat, Galanor Alliance, Galanor Movement and Galanor Combat. The Alliance phase is when reinforcements appear and when various Galanor armies are alerted. Only one scenario is offered, and the single option available to diversify the game is the choice of beginner, advanced and expert difficulty levels. The game ends after 40 turns (at the beginner difficulty level), when all the forces of one side have been eliminated or when one side's citadel has been captured.

At the start of the game, each fortress or citadel houses a leader and his designated troops; these troops are unable to move without their leader. Before a leader can initially move, he must be alerted either randomly during the Alliance phase or by the presence of another army. Each leader has a leadership rating and a sorcery rating. The leadership rating aids in combat, as might well be expected. The sorcery rating defines how well the leader can summon sorcery to aid in battle.

Sorcery in the game is abstract, and I still do not fully understand it. Basically, each side will bring sorcery to bear on a battle; the effects range from weak to devastating. How much sorcery a leader can bring to bear is determined by his sorcery rating and by the distance to the nearest Rune Ring (so states the Player's Guide). Why is it, then, that when two forces do battle twice on the same hex, the levels of sorcery they summon can be totally different? And why is it that a leader with a strong sorcery

continued on page 92

StarRay

Spinnaker Software
One Kendall Square
Cambridge, MA 02139
(617) 494-1200
$29.95, Color only

Reviewed by Maurice Molyneaux

The very first time I booted *StarRay*, I got the strangest feeling I'd played this game before.

In *StarRay*, you pilot the ship that provides the game's name. Your mission is pretty simple: Blast all the invaders to atoms before they kill you or destroy all the vital installations on the planet you are defending. The planet is many screens wide and wraps around horizontally so you can fly continuously in one direction and circle the world. Fortunately, you have a long-range radar screen that shows the entire planet's surface so you can locate enemies and installations that are under attack.

The StarRay can thrust left or right, move up and down the full height of the screen and is vulnerable to collisions with its enemies and the shots they fire. The types of attackers vary from wave to wave because there are actually seven different "worlds" to defend (you must survive three waves on each one), each with its own dangers.

For example, on the first planet, called Gorbaxa, there are Landers (gee, this sounds awfully familiar), Krellian Mother Ships that shatter into a bunch of tiny attack ships when you hit them and so forth. On the second planet, called Sirion, you'll encounter Landers and weird little beehives that split into a bunch of fragments that shortly turn into batlike creatures...sounds kinda like those Krellian Mother Ships, eh? If you spend too much time completing a wave, vicious Blue Hunters will start zipping around and try to kill you.

Do you get the feeling we've been here before? Maybe the bard said it best:

An alien invasion, by any other name....

Okay, maybe the bard didn't put it *quite* that way. But the quote fits because *StarRay*, by any other name, seems to be a graphically souped-up and modified version of the classic arcade game *Defender*.

There *are* differences between this game and *Defender*, but mostly they are of the cosmetic variety. For example, on each planet there will be a different kind of installation for you to defend, ranging from energy cells to forest robots and antigravity generators, but their basic function is identical to that of the humanoids in *Defender*. You must defend them, else the Landers will land on them (hence the name) and convert them into something nasty, like gun employments.

Unlike *Defender*, the Landers do not have to lift the target to the top of the screen in order to carry out their nefarious schemes. They merely have to park their buns on the target installation for a short time and the dreadful transformation will occur. If an installation is so corrupted, it is considered lost, and you should destroy it. If all the installations are corrupted and/or destroyed rather than being blown into deep space, as *Defender* would do, the screen slowly fades to black, and the game ends. No second chance.

The StarRay fighter is equipped with a rapid-fire forward-shooting laser and a limited supply of "vaporizers," which will destroy all opponents on the screen (but not their shots!). You should be careful of the installations when shooting at ground level. Fortunately, the installations are unaffected by your smart bombs...oops! I mean vaporizers! This is all very much like *Defender*; however, unlike its inspiration, *StarRay* does not feature any sort of hyperspace option.

Okay, okay, I've jumped on the *Defender* parallel more than enough. But just because it's a *Defender*-type game, does that make it bad?

Heck no!

StarRay is a solid piece of programming. The graphics move quickly and smoothly. The sound is good, but not outstanding. The thing that makes this game fun is that it's a well-balanced blend of smooth play action and good graphics.

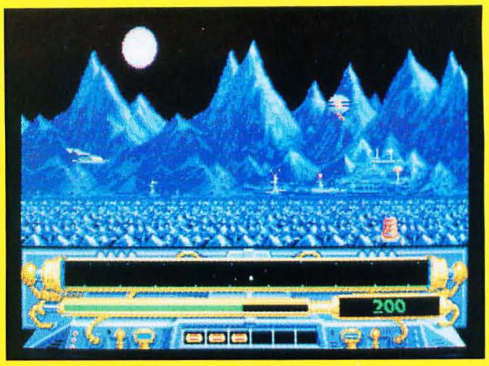
And speaking of graphics, in this department *StarRay* leaves *Defender* in the interstellar dust. No mere outline of a planet's surface in front of twinkling stars in this game. The scenery is loaded with detail. Zooming through the jungle of Sirion, you'll see no less than six levels of background detail scrolling by on the jungle floor, in addition to the tall trees is the background and the sky beyond. The sensation of movement and depth is very good.

Each planet appears for three waves, then it's off to the next one. There are a total of seven planets, each with its own unique look and its own pitfalls. The third planet, Sharlon, is covered in ice and snow. Ice floes fill the oceans. Landers are there, as always, as are the expected pod-type ships (that split into many ships when you hit them). But there's a new danger here: Shoot one of the funny-looking satellites floating halfway up the screen and it starts spinning wildly, generating interference that screws up your radar! Higher up in the game are collapsing cave passages, guided missiles and so on.

There's one drawback to these super-detailed graphics, and that's a cluttered screen. It's easy to crash into something you didn't see when you're zooming along at maximum speed. The smaller aliens can get lost in the background blur.

As far as the screen display goes, the radar scan, your shield level, score and number of vaporizers appear at the bottom of the screen rather than the top. I found that as I tried to keep the invaders from getting too low, my eyes tended to be fixed more toward the top of the screen than the bottom, and thus checking the status of my shields and so forth forced me to continuously look away from what I was doing, sometimes with disastrous results.

There are other aspects of *Star-*



Ray that differentiate it from *Defender*. First, you have only one ship and do not get extras. When you die, it's all over. The way *StarRay* makes up for what would otherwise be a fatal flaw is that the ship is equipped with a shield that can withstand a number of hits before it collapses. Your single ship can actually survive more hits than the total number of ships you start with in *Defender*.

When you blast a Lander, it doesn't always just vanish without a trace. Sometimes a Bonus Ball will appear in its wake. Picking these up (by touching them with your ship) will enhance the StarRay fighter in various ways. Bonus Balls are labeled with a letter that describes their function. "A" stands for improved acceleration, "C" gives you continuous fire for 100 shots and "I" provides ten seconds of invulnerability (great for ramming through a bunch of invaders). They can be destroyed by laser fire and disappear after a short time if you don't collect them.

Bonus Air Buses will appear if you are doing well. If you shoot these, they will drop a bonus object, which you must touch to collect.

Control can be either by keyboard or joystick, though I found the joystick to be more intuitive. The lasers are fired with the joystick fire button, but the vaporizers/smart bombs must be triggered with the keyboard. Which key? The space bar would be the obvious candidate, but did the programmers use it? Nope. They used the CapsLock key. Imagine trying to find the CapsLock key during that desperate moment when you need a vaporizer. Good luck!

The space bar pauses the game and brings up a menu of options allowing you to do such things as enable/disable the music and sound effects, restart the game at any of the first four planets and even enables a "Last Game Option," which will stop *StarRay* dead in its tracks at the end of the current game, preventing you from playing "just one more game!"

The game features background music that, while not bad, I found distracting. With it on I couldn't hear the sound effects well and found it easy to miss the sound that warns you when an installation is under attack. Fortunately, you can disable the music. You can also disable the sound effects (which makes the game play faster and harder), but I prefer the sound.

If you have one megabyte or more RAM, the title screen will be accompanied by digitized music, which can be output through an ST Replay cartridge (if you have one) and played through your stereo. Great, except that I don't particularly like the title music, and the other sounds in the game (including the background music) can't be output in this manner. The most annoying thing about the title music is that every time the game ends, the program reloads the title screen and music and plays it again. Fortunately, one button press kills it, but it's annoying nonetheless.

The "manual", which consists of instructions printed on the back of a small StarRay poster is written for the Amiga version. A small card featuring notes for the ST version accompanies this. The instructions are brief. . . too brief. There's a lot of glib writing, but not much useful information. For example, there is no mention of whether it is possible to recharge your shields or how to gain extra vaporizers. Neither is there even a hint that the destruction of all your installations will cause the end of the game.

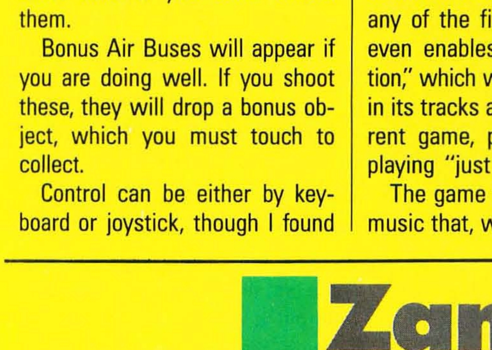
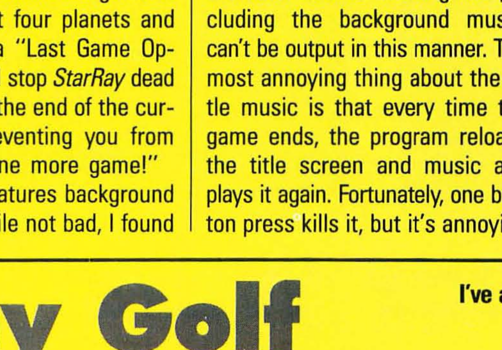
But, all in all, *StarRay* is a fun and playable game. It adds enough twists to keep it from being a blow-by-blow rerun of *Defender*, and the graphics add a lot to its appeal. If you liked *Defender*, you'll probably love *StarRay*.

I just wish there was a #%§@# hyperspace button! ■

Zany Golf

Electronic Arts
1820 Gateway Drive
San Mateo, CA 94404
(800) 245-4525
$39.95, Color only

Reviewed by Scott Wasser

I've always felt that the best computer simulations were those that enabled you to do something you couldn't or wouldn't do in real life. If not for our STs, how many of us would ever play in the World Series, pilot a jet fighter or slay a dragon?

On the other hand, who needs a program that simulates something like miniature golf? Is there anyone, other than someone who perhaps fears dying of boredom before the 18th hole, who has not experienced artificial-turf fairways and Day-Glo-colored golf balls? Probably not. So what would ever possess a computer-game designer named Will Harvey to create a miniature-golf simulation called *Zany Golf*?

Answer: a) magic carpets, b) a bouncing hamburger, c) disintegrator beams or d) a hole with a mind of its own.

continued on page 93

Tolkien's War in Middle Earth

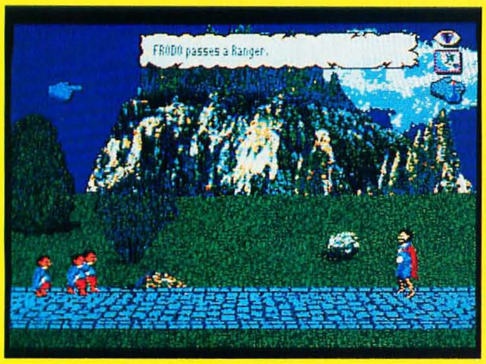
by J.R.R.
Distributed by Melbourne House
711 West 17th Street, Suite C9
Costa Mesa, CA 92627
(714) 631-1001
$49.99, Color only

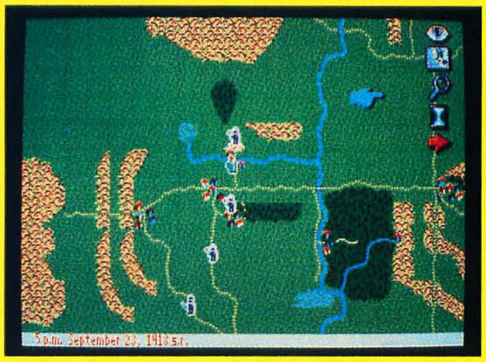
Reviewed by Frank Eva

Three rings for the Elven-kings under the sky, Seven for the Dwarf-Lords in their halls of stone, Nine for Mortal Men doomed to die, One for the Dark Lord on his dark throne, In the Land of Mordor where the Shadows lie. One Ring to rule them all, One Ring to find them, One Ring to bring them all and in the darkness bind them/In the land of Mordor where the Shadows lie.

And so begins what purports to be the definitive computer version of *The Lord of the Rings* by J.R.R. Tolkien, entitled *War in Middle Earth*, approved by the Tolkien estate itself! Actually, though, the computer game could hardly be expected to encompass all three stories of the trilogy, and the player finds himself somewhere between the first and last episodes.

The main character, Frodo Baggins, has just inherited a ring from his Uncle Bilbo Baggins' estate. The ring, according to the wizard Gandalf the Grey, is the One Ring. It was forged by Sauron the Dark Lord and is both powerful and evil. It contains much of Sauron's power, and if Sauron were to obtain it, he would defeat all of his enemies and subjugate the whole of Middle Earth. If Frodo (the ringbearer) were to attempt to use the One Ring against Sauron, he would be corrupted, ac-





Star Wars

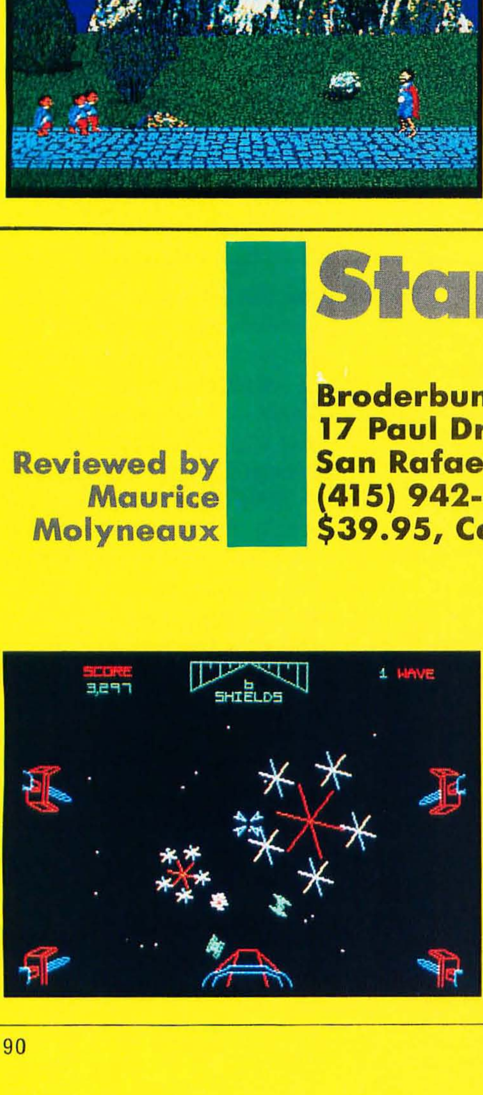
Broderbund Software
17 Paul Drive
San Rafael, CA 94903-2101
(415) 942-3200
$39.95, Color only

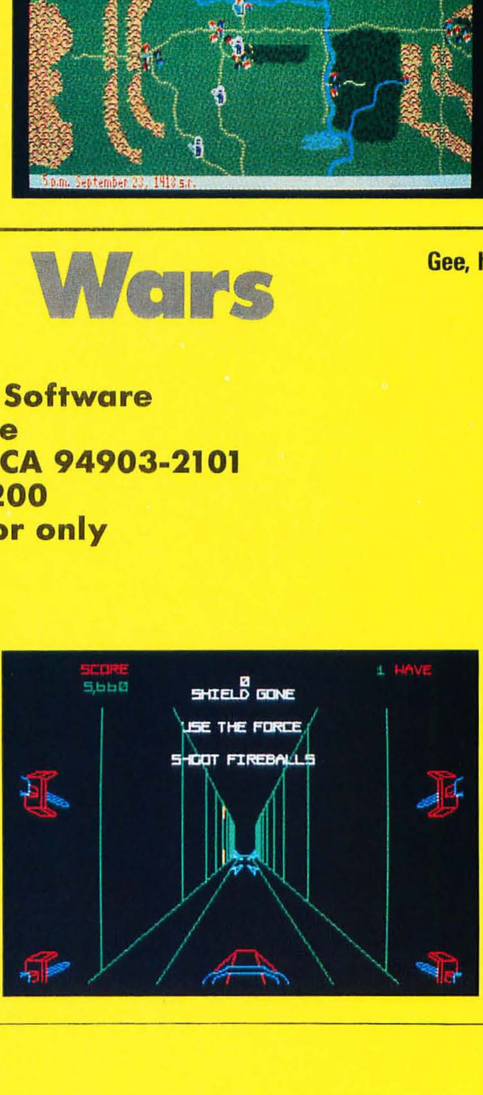
Reviewed by Maurice Molyneaux

Gee, has it really been over 12 years since the first time I saw Luke Skywalker put a proton torpedo up the Death Star's shaft? *Star Wars'* climactic attack on the planet-smashing battlestation is one of those sequences on film that is not soon forgotten. The dizzying run down the Death Star trench, while simpler as a visual effect than many of those seen in later *Star Wars* films, was extremely effective. How many of us wished *we* could pilot an X-wing fighter down and blast that battlestation to smithereens?

In the early 80s, Atari Inc. released a coin-op arcade game that let you do just that: play Luke Skywalker in the final assault on the Death Star. The game featured the *Star Wars* theme music and digitized bits of dialogue from the film in addition to nice graphics and gameplay. It was a big hit.

continued on page 93





tually becoming a Dark Lord himself.

Therefore, the quest is to destroy the One Ring. This can only be accomplished by tossing it into the Cracks of Doom (the volcano, Mt. Doom), where it was originally forged. Mt. Doom is in the center of Sauron's kingdom of Mordor, far to the southeast.

Along the way, armies of orcs, nazguls, dunlendings and wizards will attempt to wrest the One Ring away from the ringbearer. If this should happen, all is not lost, for the ring can be retrieved and the quest continued. However, the ultimate goal is best pursued by attempting to gather an army of friendly warriors to defend the ringbearer so that the One Ring can be protected.

War in Middle Earth is not just a fantasy role-playing adventure, but also a war strategy game in which the player must deploy the forces of good against bad to stem the tide of evil and facilitate performance of the quest. As far as role-playing goes, the player takes on the role of not one character, but many! He may leave his role of ringbearer to take on the role of Gandalf the Wizard or King Aaragorn, as well as powerful generals, such as Theoden, Lord of the Mark and Faramir, leader of the Rangers of Ithilien.

As a war strategy game goes, when a clash with an unfriendly army occurs, the player must decide whether members of his merry gang will engage, attack, withdraw or retreat. Not only can he do this for individual members of his troop, but also squads of friendly soldiers who have joined the holy crusade.

The game is played on a 36-screen map that scrolls in any direction. The scrolling is smooth and fast, something I can't say about the animation. After leaving the map, moving to the animation level reveals a small area of the map in which the characters can walk. Because of the lack of tastefully selected colors and clarity, this animation is substandard. The backgrounds look more like they were hand drawn by talented children on an 8-bit computer than actual digitizations (as claimed in the manual) from real life.

War in Middle Earth contains much in the way of pictures and maps, actually requiring three single-sided disks to hold everything. While this is a plus for multiple-drive owners, it's a hassle for the single-drive owner. It seems that just about every move requires a disk swap. For the two-drive owner, both drives can be used with little swapping, other than for the game-save function, which saves a single game to Disk One.

Let's look at some of the game's high points. First of all, *War in Middle Earth* is fantasy-rich! I can't imagine anyone who enjoys fantasy not being absolutely enthralled with the world of Tolkien as interpreted in this game. Not being a fan of *The Hobbit* or *The Lord of the Rings*, my collaborator has made me aware that the plot line of *War in Middle Earth* seems very Tolkienesque.

The user interface is easy to manipulate and actually makes playing the game a treat. There are three levels of play, two of which were previously alluded to. The "full map" level is an overview of the entire playing area. Forces at your command and opposing armies are represented by blinking dots. Time does not pass on this level. Icons of an eye, a scroll, a magnifying glass and an hourglass allow all sorts of functions, including saving and reloading a single game. (By the way, there is a game saved to the original disk, facilitating a rudimentary "restart." Once another game is saved, however, this feature is accomplished by the "load game" function.)

The campaign level represents a portion of Middle Earth approximately 250 miles by 150 miles. This is the 36-screen map, which can be multi-scrolled. Selecting this level's magnifying glass allows the player to enlarge a small area, at which time the program loads-in a background representing the area and animations of any characters that may be wandering through, thus moving up to the "animation" level.

At this level, selecting the "eye" icon allows the player to gaze into the Mirror of Galadriel and see the status of characters on-screen. The "provisions" icon facilitates taking/dropping/using objects, and the "map" icon allows the player to move down to the campaign level. All in all, it's a simple and intuitive interface.

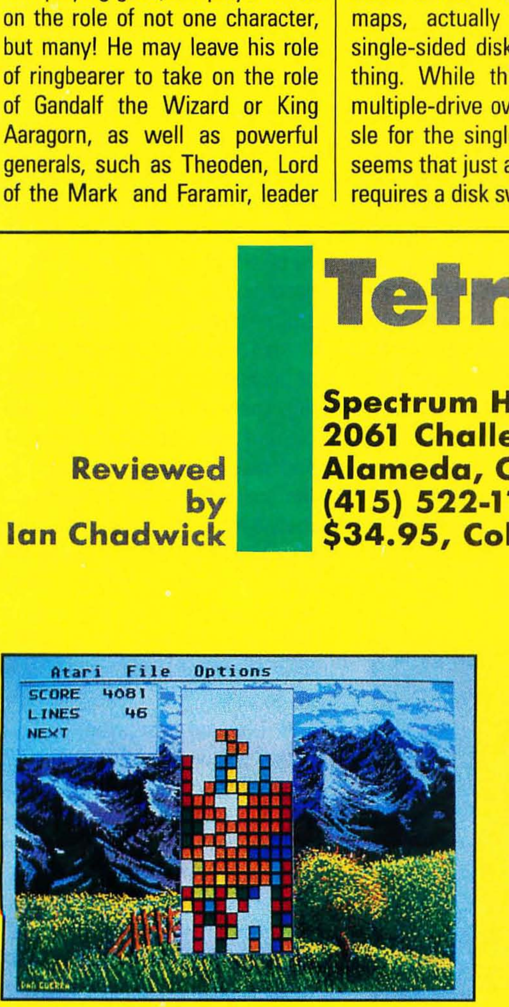
Last but not least, the game's excellent 46-page instruction manual is not only complete when it comes to explaining game play and the player's ultimate goal, but also provides much historical background for the uninitiated player.

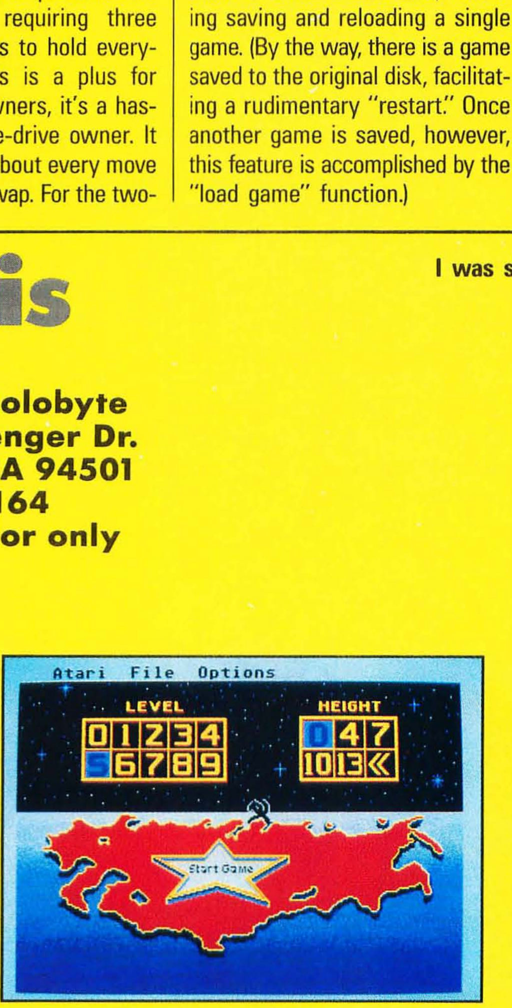
If the potential buyer can look beyond the somewhat chintzy graphics, *War in Middle Earth* is a fantasy-lover's ultimate contest—one that he will come back to again and again. ■

Tetris

Reviewed by Ian Chadwick

Spectrum Holobyte
2061 Challenger Dr.
Alameda, CA 94501
(415) 522-1164
$34.95, Color only

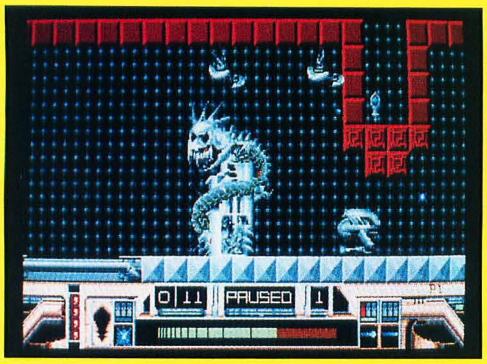


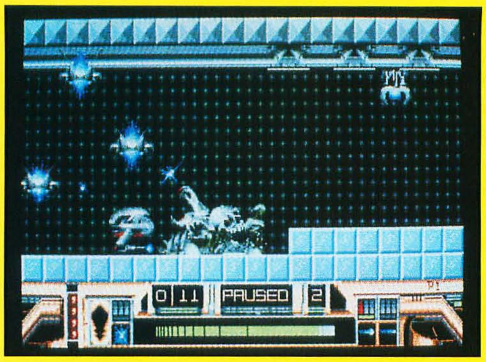


I was sitting in the Arbat Cafe on Kalinin Prospekt one night eating some zakuski when a KGB colonel, Yuri Lysenko, of the First Chief Directorate came to my table. I thought he was there simply to check my propiska, but he waved them away with no apparent interest. Instead, he asked if he could join me. One doesn't say no to the KGB, so I offered him the vacant chair. He sat down and signaled for the waiter to bring a bottle of Zolotoye Kolsto to the table.

The night lights from Red Square sparkled off the sluggish waters of the Moscow River and the metallic trill of a balalaika filled the air. Between salty spoonfuls of beluga caviar and hefty gulps of vodka, we discussed how glasnost was affecting his job. After all, if America was no longer the "Great Enemy" and the USSR was not the "Evil Empire," what purpose could

continued on page 94





JUG · CONTINUED

al increases in difficulty. Allowing selection of an overall difficulty level is, therefore, a nice feature that allows a more experienced player to begin at a higher level instead of having to fight his way from the beginning.

There are a number of weapons systems available that Jug can store and use on demand. These include high-energy plasma bolts, high-speed laser beams and Zeoquark particle emitters (smart bombs). Teleporters are two-way devices that can be used to instantaneously beam out of a dangerous section of the planet.

The status display at the bottom of the screen contains indicators that, when lit, register the level of damage sustained as well as the amount of available fuel and weapons systems stores.

In order to move from zone to zone, a "sector key" must be found. These devices, when installed in Jug, have the ability to dissolve solid walls. Also, certain areas of the planet are contaminated with radiation, and travel within them drains fuel supplies at a greater rate.

Jug's graphics are finely detailed; several shades of a color are used to give a feeling of depth and roundness. Every sprite is really a work of video art and many are uniquely animated. Also, it's nice to see the effort made to provide such a wide variety of onscreen antagonists.

MicroDeal has addressed at least one misgiving about several previous titles: the lack of a music toggle. In *Jug*, F7 allows just the volume of the music to be decreased, while F8 serves to increase it.

All in all, Jug is one heckuva good play! ■

SORCERER LORD · CONTINUED

rating, sitting on top of a Rune Ring, will sometimes bring only weak levels of sorcery to a battle? Beats me. This system needs either more work or more explanation.

The general rules of wargaming apply here: A unit must stop movement when it enters a hex containing an enemy unit. A unit in a fortress or citadel receives a defensive bonus. A unit is forced to retreat when it loses a battle. All the stuff of which wargames are made. What's that? Did I hear someone ask, "What about stacking?"

Is stacking allowed? You bet it is. Now let's figure out the scale of the game. If a turn is one day and the average distance an army can move in one turn is five hexes, then how big is a hex? Lessee. . .figure your average foot soldier can march at five miles an hour for eight hours a day. Sounds like 40 miles a day to me (and yes, I know that he'd be one hell of a tired soldier). Divide that 40 miles by five hexes, and you get eight miles a hex. What's the point? There is no stacking limit and the game's artificial intelligence routinely builds stacks of 40,000 or more troops. How does a 40,000-strong army fight in an eight-mile space?

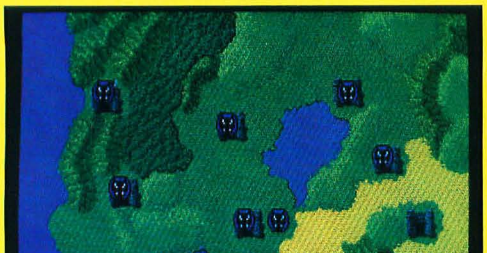
And therein lies the real rub for me. When playing *Sorcerer Lord*, you will see no lines of battle drawn. The war quickly degenerates into a disorganized game of cat-and-mouse, with each player's monster stack trying to catch a smaller enemy stack. Silly to watch, not much fun to play. I can't think of any way to win this game "traditionally." The Shadowlord side rarely splits a stack and by about the tenth turn will have concentrated the entirety of its forces into one or two hexes. I also must add that the Shadowlord forces can move up to seven hexes per turn and are not slowed down by most terrain.

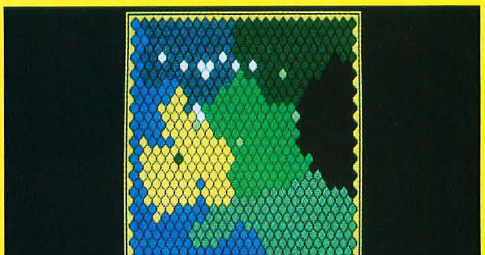
What compounds this particular problem is the "capture the flag" victory condition. If the Shadowlord's forces take your citadel, the game is over. The reverse is also true, but your forces only move an average of five hexes before terrain penalties.

And there are yet more problems. There is no Undo function. Once a unit is moved, that move cannot be taken back. There should be an opportunity to retract a move while still in control of that unit.

Also, following the movements of the enemy troops is disconcerting. There is no way to tell a single unit from a stack of ten units, and the pieces do not slide across the board, but rather vanish and reappear in the next hex. Since they all look identical, it can be difficult to ascertain which unit is moving where. Lastly, after a game is finished, the program ends. You have to reboot if you want another go at it.

If I sound dismayed by this game, it's because it *could* have been what so many of us are looking for. Had the user interface been more polished, a stacking limit imposed and a choice of scenarios offered, *Sorcerer Lord* might well have been a fine game. As it stands, I see little to recommend it. ■


TURN: 1 PHASE: MOVEMENT PLAYER: SHADOWLORD LEVEL: BEGINNER



ELVIOS OF IMRYTH ALERTED 1600 WARRIORS 800 RIDERS
TURN: 1 PHASE: ALLIANCE PLAYER: GALANOR LEVEL: BEGINNER

ZANY GOLF • CONTINUED

The answer is e) all of the above. What Harvey has created is his own fantasy version of miniature golf. It is a version that, without a computer, could never be played. I'll explain why shortly.

First, let me say that *Zany Golf* has the qualities of a good entertainment program because it is easy to learn, difficult to master and addictive. It also features state-of-the-art graphics and animation.

As in real miniature golf, the object in *Zany Golf* is to conquer the course in as few strokes as possible. Using only a mouse, you line up your shot, decide how hard to hit the ball and let fly.

The perspective is absolutely amazing. Just as the human eye can only focus on part of a given hole, *Zany Golf* only allows you to see a portion of a hole at a time. Moving the cursor to the edge of the screen will cause the image to scroll in that direction, revealing more of the hole.

Once the ball is struck, the screen scrolls automatically to follow its path. For the most part, the action is so smooth and realistic, you'll swear you're actually watching a rolling ball rather than some cathode ray tube pixels being excited by computer programming.

While *Zany Golf* is a wonderful example of computer programming, it really sparkles as an exercise in tickling the imagination. Anyone who has played miniature golf is familiar with the various obstacles that make it difficult to sink a putt. Twirling windmills and off-camber putting greens challenge even the most skillful golfers.

But those obstacles are nothing compared to what you'll encounter in *Zany Golf*. These obstacles were generated by Harvey's obviously fertile imagination, and most could never be duplicated on a real miniature golf course. Fortunately, there are also some equally fantastic aids that can help you guide your shot to the cup.

It's difficult to verbalize how wacky *Zany Golf* really is. But here's a glimpse at each of its nine holes anyway:

1. Windmill Hole—A twist on the miniature golf classic. This time the windmill is at the end of a dogleg right and at the top of an incline.
2. Hamburger Hole—A bouncing hamburger, complete with pickles and onions, gyrates up and down over the cup. If you time the bouncing just right and ricochet your ball off the ketchup bottle, you might make par.
3. Walls—Three walls rise and fall in sequence. You must hit the farthest wall dead-center while it's up to have a chance.
4. Pinball—You can't even take a shot at the hole until you first guide your ball through a giant pinball machine that features flippers and drop targets.
5. Fans—The maze that leads from tee to cup would be impossible to navigate, except for one thing: Strategically placed fans allow you to direct the ball by blowing it (using the mouse to make the fans spin).
6. Magic Carpets—Another "a-maze-ing" hole. This time the ball will move in the same direction you move the mouse—but only when it's on one of the magic carpets.
7. Castle—Try to get your ball into the entrance of a castle that sits perched atop a three-sided hill. The castle gate only opens for a few seconds every time a trumpet blasts.
8. Ant Hill—This one makes the castle's three-sided hill seem easy to navigate. Not only does the hill have more sides, but the hole on top of it moves at random.
9. Energy—The ultimate miniature golf challenge. Laser beams, particle rays and energy transporters make *Star Wars*' special effects look tame. It's nearly impossible to sink a putt on this one.

I could nitpick and find a few minor negative things to say about *Zany Golf*. But the bottom line is that the program is so ingenious, so well executed and so much fun that complaining about it would make me feel like a real "putts." ■

STAR WARS • CONTINUED

The game used color vector graphics to simulate a first-person perspective view from an X-wing fighter. On-screen, you could see the nose of your fighter and the laser cannons extending from your wingtips (out of view). To operate the fighter, you gripped a massive controller that turned from side to side like a steering wheel and had twin pilot handles you could pitch back and forth. This provided turning control as well as climbing and diving capability. On the controller were four triggers, all of which fired your lasers. If you used the thumb and index finger of both hands and rapidly pumped all these buttons, you could get all four of your X-wings' lasers to blaze like mad.

You only had one life, but your ship was equipped with shields, each of which could sustain one hit or collision. When all your shields were gone, one hit sent you to join Obi Wan Kenobi.

The game had different difficulty levels, but a typical "wave" went as follows. First you were flying in space above the Death Star, blasting TIE fighters and at the same time detonating the "fireball" shots they fired back at you. The TIE fighters swarmed like gnats and were difficult to hit on the higher levels. Darth Vader's custom TIE ship would cruise through, and while you couldn't destroy it, every time you hit his ship you got a hefty pile of points.

Once the TIE fighters had given up (or you vanquished all of them), your fighter plunged to the Death Star, where you would fly low over the surface, destroying laser bunkers and blasting the laser-firing caps on the battlestation's towers. If you could blast the tops off *all* the towers, you got a 50,000-point bonus. Worth it, but tricky to do. Of course, both the bunkers *and* towers shot fireballs at you, so you had to deal with them while trying to navigate between the towers!

The final part of the wave was the actual run down the Death Star trench. While the arcade game luckily didn't include Vader and his goons chasing you down this metal-walled canyon, it complicated the maneuver by forcing you to deal with fireballs fired from gun emplacements in the trench and threw in catwalks and wall segments for you to maneuver around. You might have to shoot a fireball while "porpoising" over and under catwalks and zigzagging among walls.

At the end of the trench was the thermal exhaust port. You knew when it was coming up because you'd hear Han Solo's voice yell, "Yahoo! You're all clear, kid!" If you missed it with your lasers, you hit the end of the trench, lost one shield and had to try the same trench all over again. If your shots hit the port, your fighter would race away from the Death Star, which would explode in a pretty display of vector-fireworks. Han's voice would congratulate you, you would gain one shield and then it would start all over again. . . but harder.

Now Broderbund has released for the ST the Domark Ltd. conversion of this arcade classic. The game is incredibly faithful to the coin-op original, from the title, level-select and high-score screens through the look of the game and the gameplay itself. The digitized dialogue is there, as are the TIE fighters, fireballs, towers, bunkers, catwalks and other imperial dangers. The graphics are rendered with lines only, giving the same feel as the vector graphics of the arcade game.

Aside from the good digitized voices, the game's sound effects are okay, but not great. The weakest sound of all is the destruction of the Death Star. It sounds too blinky-clinky, not at all like a big explosion. I've heard the ST sound chip do better.

There are a few minor differences between the arcade and ST versions: The laser cannons in the trench are flat squares on the walls and do not stick out like those in the arcade. Therefore, they are much harder to hit. I also noticed a complete lack of the coin-op's trench wall panels, which could be blasted for extra points. The TIE fighters are much easier to deal with too, as they don't seem to move about quite as zanily as their arcade counterparts.

The strategy required, however, is much the same as in the arcade. You have to concentrate on not letting the fireballs hit you. Fortunately, you can often steer out of their way, but you have to make sure that in so evading them you don't crash into a tower or catwalk. I found that if you have any trouble with the TIE fighters' shots hitting your shields, just hold the cursor at screen center and fire continuously; no fireball will hit your ship. This is because the fireballs always cross the center of the screen when they are about to strike your ship. This only works well with the TIEs, and you mustn't be maneuvering a lot. Down in the trench, you find that the fireballs stay at one level and to one side or the other of the trench, so it's simple to avoid them. . . if you don't sweat the catwalks. It's easy to miss the exhaust port, so the moment you hear Han's "Yahoo!" you should be firing like mad at it. Don't sweat any fireballs fired by guns around the port. When you hit the port, all such shots are rendered harmless.

For extra points, pound your shots into Darth's ship whenever possible, and particularly when he retreats toward the Death Star at the end of the TIE combat mode. You'll get 2,000 points for each hit.

High scores are entered with the game controller, arcade style, by "zapping" letters. The top score is saved to disk; all others are not. There doesn't seem to be a key that brings up the high-score table. Also, while the instructions don't mention it, the F10 key brings up the game's version number. I was unable to find a way to restart a game in the middle. To start over, you must first die.

This game is pretty good, but it has an Achilles heel: the controls. Control can be either by mouse or joystick, but neither is anything like the big controller used in the arcades. You can set either one to "airplane" controls, where pushing the stick/mouse forward causes the ship to dive and pulling back makes it climb, or to "normal", where pushing forward makes your ship go up and pulling back makes it go down. All fine and good, but there are problems. The mouse is responsive enough, but it's easy to move it too quickly and have it skip. I once jammed it forward to dive, moved it too fast and my ship only nosed down a hair; I plowed through three catwalks and *boom!*. . .game over.

The joystick is so bad that I hate to mention it in an otherwise positive review. It's so overresponsive that I, who will usually take a stick over a mouse any day, refuse to even allow a joystick to be in the same room as my ST when I'm playing this game. The version of this game that Domark Ltd. markets in Europe suffers from these same control deficiencies. It's too bad Broderbund didn't have the joystick control fixed up before they released it here.

The instructions are on a foldout sheet, like a poster. The directions are simple, showing you how to boot the game and use the options keys and controls, and also describing each portion of the attack in enough detail to get you going. This sheet includes directions for the C-64, ST and Amiga. You are instructed to boot the game from Drive A (it autoloads), but I found I could run it from the Desktop (I ran STARWARS.PRG from the Auto folder) as well. The disk is copy-protected.

I *have* to mention the packaging. The box is very nice, featuring an excellent cover painting by Ralph McQuarrie, who executed the now-famous preproduction/conceptual paintings for *Star Wars*. The back of the box features a photo from the trench sequence of the film as well as some game-screen shots. The point of interest is the little red "button" on the back of the box. Above it is a label that says: "To relive the *Star Wars* experience, use a little Force here." And what happens, you ask? The box plays a portion of the *Star Wars* theme! When UPS delivered the game to me, I heard the package before I even saw it. I told the delivery lady that I guessed it was a moot point to ask if the package was from Broderbund.

Star Wars is a good game. Perhaps it's a little dated now compared with the latest and greatest arcade hits and ST games, like Starglider II, but it's still fun to play. If you liked the arcade game, enjoy blasting spacecraft or just want to use a little "Force," buy it. ■

TETRIS • CONTINUED

there be in a huge, state-run espionage and security network?

Colonel Lysenko smiled at me. "We have not yet given up the hope of conquering you capitalists," he said as he produced two fat Cuban cigars.

I shook my head at the offer and replied smugly, "Well, if you can't do it through world revolution and subterfuge, then how? Surely you can't compete in the world markets. Your commercial production levels are far behind those of the West. You can't even supply your own people with the consumer goods we manufacture by the millions."

"True, we do not have the means to outproduce you, except in arms. But we have found a weakness, a critical flaw in your society that we can exploit."

"You feel secure enough to tell me, a journalist, this?" I figured it was just more disinformation, a soviet specialty.

"Of course. No one will believe you if you make it public. And even if they do, your culture is powerless to stop us."

"And what is this diabolical device you have?"

He blew a cloud of blue-grey smoke into the air and looked at me, eyes twinkling. "A game."

"A game? You mean like baseball? Come on, colonel, you can't be serious."

"Not a sport. A computer game."

That stopped me. A computer game? Who could ever imagine the dour, phlegmatic Russians at AcademySoft producing a computer game? Through my mind raced an image of American teenagers grappling daily with joysticks as the action heated up on a million videoscreens across the country. I hadn't seen even an XEGS in Moscow. Who was he kidding? Besides, what sort of game could they create? One about queuing in lines for bread? About the bureaucratic angles behind exit visas? "What kind of game?" I asked sarcastically. "Are you going to tell me you've invented PacMan?"

"Nothing so coarse. It was created at our Academy of Sciences, but no one realized the commerciai potential until a copy of it was installed in our central computer system and brought to my attention."

"What's the game about?" I had to admit, he had my attention.

"It's simple, really. Almost a toy. Groups of blocks fall from the top of the screen toward the bottom. The groups are simple collections of four square pieces,"—he lined up sugar cubes on the tablecloth to illustrate—"seven different groups. Your job is to rotate and move them as they fall so that they fit the best in the growing piles. When you fill a row completely with squares, it disappears and everything moves down one.

"Points accrue according to how far and how fast the pieces fall into place. When you remove enough filled rows, the level increases, and as it does, so do the game speed and the points awarded. There are nine levels, of which I've run through eight, but nine challenges the limits of my reflexes."

An admission of weakness from a KGB officer? Not likely. It sounded more like a challenge. And accustomed to the high-res graphics and action of video games in the West, I didn't think his description was even vaguely exciting. "Sounds easy to me." Not to mention boring.

He smiled that grin again, like a cat that's just cornered a particularly tasty mouse and is savoring the moment before the real fun begins. "Would you care to try it?"

"Of course." I took the bait.

We went in his car, a black Volga, for the short drive to Dherzhinsky Square. I was a little worried when we stopped in front of the infamous Lubyanka building, but it was only to let his driver go.

He steered me away from the KGB headquarters to the doors of Detsky Mir, the giant children's store across the street. Inside, he flashed his badge and took me to a small display on the second floor, where a suspiciously familiar computer, the Atariski Soyuz, was on display. In front of all the screens were familiar western faces: fellow journalists and embassy staff, pushing and shoving each other for a chance to play again. They looked haggard, glassy-eyed.

I moved closer to watch. It looked easy. Too easy. But the players were obviously hooked. What was the gimmick? Subliminal imagery? Subsonic sound?

I nodded my head. The guards rudely shouldered one of the players out of his seat and offered me the chair. The man protested weakly, but subsided at the sight of my host's uniform. I sat down and flexed my fingers. These were the hands that broke a million in Asteroids. Surely, little tumbling blocks would not defeat them!

The blocks began to fall. I lined them up and dropped them into their slots with a tap of the space bar. Nothing to it. They were falling too slowly. I started to get impatient, anxious for the blocks to come faster. Level two passed, then level three. The action was speeding up. I was conscious that the other games around me had paused as everyone watched my progress.

At level five I ran into trouble. I missed a slot and built up a pile of unfinished rows that might have doomed me, but I managed to get them settled and gone before the next level came.

At level six I began to sweat. The blocks were falling steadily, and I didn't have long enough to line them up and drop them into place. Still, I managed to make it to the end, although the bottom of the screen was littered with incomplete rows.

Level seven. My hands were shaking, my mouth went dry. Blocks fell like rain. I twisted them, slid them among their fellows, but not enough. I misjudged once too often. The tower of incomplete rows reached the top of the screen and the game ended. I sat back and sighed with a mixture of relief and frustration. Almost! The polite patter of applause surrounded me. Not bad for a beginner.

I couldn't stop there. I had to try again, break the barrier, crash through to the eighth level. No one complained as I started again. And again. And again. Somewhere in my head I heard laughter, and knew it was for me.

Early the next morning they finally dragged me away—I was still only on level eight. I was the only one playing; the store had long since closed and my KGB escort was left to take me back to the Rossiya Hotel. In my dreams, my fingers twitched as I stroked the keys. I knew Colonel Lysenko was right. We had no defenses against this thing. Once it hit the shelves, we'd be helpless addicts, mindlessly tapping the keys until we fell to the floor, victims of the deadly game as the Red Army marched in to effortlessly conquer us. I awoke, my body aching and sore. My fingers wouldn't stop wriggling as they played an imaginary keyboard.

I picked up the phone and called Colonel Lysenko. I had to play again. One evening and I was hooked.

"Did you enjoy my little demonstration?" He sounded tired. In the background I could hear him typing.

"Lysenko, I don't care what it costs. I *must* get a copy of that game."

He laughed, an unpleasant sound. "You'll have one soon, I promise. We've launched it in the West under the name *Tetris*. It should be available by the time you return home."

"You fiends!" I shouted. But Lysenko didn't respond. I listened carefully, thinking he had hung up, but I could hear his breathing and the incessant tapping of the keys in the background. Keys? Wait a second! "Lysenko, are you playing that game?"

"Yes," he muttered dejectedly.

"You mean...?" Then it struck me. The copy installed on the central KGB computer system! "You're hooked too?

"But why did you do this to me?"

"We hoped you'd prove immune. We figured if we could find someone who didn't get addicted, we could discover an antidote." He growled and cursed softly. I could picture the blocks falling on his screen.

"And now?"

"Now we keep looking." He hissed at his computer, and I lost his attention. I hung up the phone and tried to roll back to sleep, but the game tugged at me, demanding. It was going to be a long week, but I was determined not to let it get to me as it had overtaken him. I refused to be its next victim.

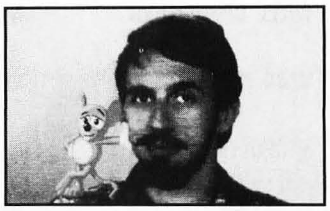
A few weeks later, I was out of Russia and home. I stopped at a small computer shop on Toronto's Yonge Street to buy a recent copy of ST-LOG. Inside the computer shop, I found the employees wandering around aimlessly, looking dazed, unresponsive. I knew why immediately: A copy of *Tetris* lay open on the counter and was being played on every computer in the store. I started to back out, but something wouldn't let me. My fingers began their dance. I couldn't stop myself. It was like a magnet pulling me to the keyboard. I sat down and the blocks began to fall. . . .

# # #

First there was *The Spy Who Came in From the Cold*, then there was *Red Storm Rising*, and now there's *Tetris*, the first piece of Soviet software to make its way across the ocean to the commercial wonderland of capitalist marketing. It's so basic in concept that it brought a sneer of contempt to the face of one employee at a computer store when I asked about it. He took one look and dismissed it as irrelevant. Too bad he never played it before he shortshrifted it. *Tetris* is annoyingly simple and intensely addictive. I love it.

It was developed, as mentioned, by Alexei Pazhitnov and Vadim Gerasimov of AcademySoft, the Computer Center of the USSR Academy of Sciences. It has been a major success on the PC and Mac already. There are several additional features, including high-score, clear and save, help, next-shape and statistics displays, tournament play and advanced (faster) mode.

Tetris is one of the best games on the ST, bar none—challenging, demanding and nonviolent. I highly recommend it. ■



Blissfully ignorant of the realities of time and space and plain old common sense, ***Maurice Molyneaux*** *hopes someone will someday discover "retroactive reincarnation" so that when he dies he can come back in a previous life as animation director Chuck Jones. His greatest fear would be to come back as Wile E. Coyote, and in the process have to learn some humility.*

Scott Wasser *has been a daily newspaper reporter and editor for the past 12 years, and has been interfacing with computers for the last four. He has written columns and feature stories about computer hardware, software and home electronics and is a regular reviewer for* ST-LOG.

Ian Chadwick *is a Toronto-based technical writer who lives in an increasingly small house with his wife, Susan, six cats, one dog, two rats and several field mice (which moved in recently despite the cats). And that's not to mention the neighborhood's stray cats, which take up residence as the mood moves them.*

Frank Eva *is an auditor by profession, but has been involved in the computer industry ever since his purchase of an Atari 400 many years ago. He has dabbled in programming and has had several text adventures published.*

INDEX TO ADVERTISERS

ADVERTISER:	PAGE:	READER SERVICE #
AVANT GARDE	63	108
BRE SOFTWARE	25	102
BETACAD	57	107
DATEL COMPUTERS	41	104
ILIAD SOFTWARE	49	105
MICROTYME	CV. #3	111
MIGRAPH	31	103
PROCO PRODUCTS	49	106
SOFTEK	13	101
WEDGEWOOD RENTAL	96	109

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

RENTING SOFTWARE ISN'T HARD!

It's as easy as picking up the phone and giving your order. If you have a credit card, it's even easier. The hardest part may be waiting for the mail to come!

We're having a special sale, with up to 80% off selected software. Call now for a complete list.

Call toll-free outside Texas: 1—800—433-2938
— Inside Texas call: 817—292-7396

WEDGWOOD RENTAL
5316 Woodway Drive
Fort Worth, Texas 76133

CIRCLE #109 ON READER SERVICE CARD.

```
The ST-LOG #36 diskette contains 25 magazine files.
They are listed below.


  FILENAME.EXT    FILE TYPE    COMMENTS
  ------------    ---------    --------

CMANSHIP.ARC contains:
  C36      .C      C             C-MANSHIP

CALCULTR.ARC contains:
  CALCULTR.ACC    ACCESSORY      PROG. CALCULATOR

CAPITAL.ARC contains:
  CAPITAL .PRG    RUN FILE       CAPITAL ST
  CAP     .RSC    RESOURCE
  USAMAPHI.PI3    DEGAS          PICTURE FILE
  USAMAPLO.PI1    DEGAS          PICTURE FILE

CHARFUNC.ARC contains:
  CHARFUNC.C      C              TOOLS FOR C-CHEST
  CHARTEST.C      C              TEST PROGRAM SOURCE

DOMINOES.ARC contains:
  DOMINO_2.PRG    RUN FILE       PD PARADE
  SCORE           DATA

SEASIDE.ARC contains:
  SEASIDE .PRG    RUN FILE       COMPUKID CONNECTION
  BASCODE .EXE    DATA
  C       .DAT    DATA
  CC      .DAT    DATA
  CRE     .DAT    DATA
  CREA    .DAT    DATA
  CREAT   .DAT    DATA
  CREATU  .DAT    DATA
  CREATUR .DAT    DATA
  CREATURX.DAT    DATA
  CREATUXX.DAT    DATA

UFOTO.ARC contains:
  PHOTO   .PRG    RUN FILE       U-F-OTO
  TOWN    .PI1    DEGAS          PICTURE FILE

VIDPOKER.ARC contains:
  VIDPOKER.PRG    RUN FILE       PD PARADE
  VIDPOKER.TXT    TEXT           INSTRUCTIONS

  ARCX    .TTP    RUN FILE       UNARCHIVING PROGRAM
  README  .DOC    TEXT           Disk instructions
  UNARCHIV.DOC    TEXT           Unarchiving instructions


Disk instructions:
   Only those files with .PRG, .TOS, or .TTP
   extensions may be run from the GEM desktop.
   Other programs may require additional software
   as shown below.  The files on this disk have
   been archived (compressed) into .ARC files.  To
   restore the programs to their runnable form,
   follow the instructions found in the
   UNARCHIV.DOC file.

   NOTE: Due to space limitations, the source code
   for many of the programs is not on this disk.  If
   you wish to obtain the source code files, send
   your original October '89 disk (after making a
   copy; the original will be erased), along with
   a postage-paid mailer to:

         ST-LOG OCTOBER SOURCE CODE
         P.O. BOX 1413-M.O.
         MANCHESTER, CT  06040-1413

WARNING: Be sure to read the appropriate magazine
         articles before attempting to run the programs
         on this disk.  Failure to do so may yield
         confusing results.

                 .EXT  DESCRIPTION
                 ----  -----------
                 .BAS  Requires GFA BASIC
                 .LST  Requires GFA BASIC
                 .C    Requires C compiler
                 .S    Requires 68000 assembler
                 .PAS  Requires Pascal compiler
```

Double Dragon —Aaargh!

BY JOHN SIFTON

Well, there it was on eight-year-old Matthew's Christmas list—*Double Dragon*. I checked with the local dealer on December 20th. "Come back in February," he said, "but phone first." So we—that is, Mom and Dad, both pacifists, both pro-feminist, both still stubbornly socialist children of the 60s—wound up putting an envelope addressed to Matthew under the tree on Christmas Eve. The note inside said: "I.O.U. Double Dragon. Merry Xmas, Santa."

"So when's Santa gonna ante up, Dad?" Matthew demanded Christmas morning, staring at me meaningfully.

"Well. . .uh. . .they said it might be in February, but. . .you know. . .uh. . .you never can be sure," I replied with my usual eloquence.

For the next six weeks, Matthew was a model of patience. He didn't ask me more than five or six times a day, seven days a week whether I'd phoned the computer store. And it was with no little relief that I went to the dealer in mid-February and picked up *Double Dragon*.

"Aaaaaalriiight!" roared Matthew when he saw the package. Up he thundered to the third floor, where my 1040 lies with its hard disk and a bookshelf on a home-made desk. As he ran, the house rocked; pictures tilted, three books sailed off the shelf. I stared heavenward, asking the gods of computerdom how many scallops of precious data had been sliced off my hard disk.

As we booted up the two single-sided disks, the theme music—a pulsing, neuron-pulverizing rock instrumental—blared from the color monitor. I knew it was time to leave.

But getting away from *Double Dragon* was not as easy as I'd expected. To grasp why, you have to understand the living and working arrangements in our house. My wife works nine to five and beyond for a union. And I, having quit my strategic-planning job in government, make a less-than-adequate living struggling to churn out great drama on the 1040.

This arrangement works pretty well under normal circumstances. But *Double Dragon*, in addition to bringing electronic chaos to the monitor screen, zapped our happily disorganized life with an electroshock of hyperactive nihilism that sent the family's collective synapses into hyperspasm.

Let me set the scene.

Dad's up on the third floor working on the computer, turning himself inside out struggling for the perfect, magical phrase that'll crystallize a scene and make the story editor purr without causing a catfight among the producers. The brakes of a school bus screech outside. Matthew and his friend, Guy, disembark and enter the house. Four small feet tip the Richter scale at about 5.8 as they run up the stairs. Two books fall off the shelf over the computer. The hard disk tips as if drunk. Dad feels beads of cold sweat forming on the small of his back.

"Hey, Dad!" Matthew says with a big grin, "Can Guy and I play Double Dragon?"

"No, I'm working," I answer grumpily.

"Guy's really looking forward to it," he pleads.

"I have to work. Maybe later." Dad's starting to feel like having a drink.

"When?"

"Say half an hour." Maybe a nice soothing scotch with one cube and a splash of water.

"How long's that?"

"Thirty minutes!" Forget the water in the scotch.

"I'll get the egg-timer!" Matthew, all grins, charges down the stairs with Guy.

"Don't bother," Dad says, voice even, stomach approaching meltdown as he parks the hard disk. "You can play now."

"Yaaaaaaaaay!" Matthew and Guy scream together as they stampede up the stairs, rocking four more books off the shelf.

Down in the kitchen, Dad drinks straight from the scotch bottle.

Sometimes, of course, Dad hangs tough, usually out of deadline desperation. "Maybe after supper," he says.

Four big eyes stare at him appealingly. "But Guy was really lookin' forward to it," Matthew says. Guy nods.

Dad girds his loins and says in his deepest paternal voice, "No, I've gotta get this done."

"You know, Dad, you work too hard," Matthew says sympathetically. "And you're a dork."

Over dinner that night, Mom turns to Dad. "Do you want your son to turn into a computer zombie?" she asks pointedly.

"Uh . . . ," Dad says.

"What's a computer zombie?" Matthew asks.

"A person who spends so much time playing on the computer that he can't do a thing but twitch and stare, stare and twitch, and his hand has a permanent joystick cramp," Mom says.

Matthew considers this for a moment. "Radical!"

The next morning, as Dad shaves he stares at the reflection of his eyes in the mirror. They're phosphorescent-gray from too much word processing.

Eventually, at Matthew's insistence, Mom and Dad play *Double Dragon.* They learn that one of its big attractions for Matthew is that it's a two-player game, so a guy doesn't have to face all that street violence alone.

The sophisticated premise is that you're two tough hombres who've agreed to team up to save your girlfriends from a gang of karate kidnappers. Dad has seen better graphics on the ST, but the ones here, along with the gruesome sound effects, are lurid enough to make your joystick sweat.

You can beat the baddies with the usual array of karate kicks, blows and leaps. Matthew quickly masters all the necessary joystick twitches. Dad doesn't.

If you feel inclined to make mashed potatoes out of your enemies, you can hurl oil drums or boulders at them. Several times, Matthew accidentally makes shepherd's pie out of Dad with this tactic.

Sometimes, Dad—bruised, beaten and virtually scoreless—gets to the final screen with Matthew, where the kidnapped girl hangs in semi-undress from manacles about ten feet above the floor. After pulverizing another army of baddies, Dad and Matthew must fight to the death for the girl. Matthew demolishes Dad. As romantic music plays, the girl, miraculously freed, climbs down the wall and goes over and kisses Matthew. A tiny three-dimensional heart glows over the victor and his woman.

Matthew and Dad smile. It's heartwarming to see such a moving illustration that love makes the world go 'round and that everything else is quite beside the point.

Take death, for example. It's wonderfully tidy here. The bad guys' corpses simply vanish. Matthew, however, can resurrect himself by pressing his fire button. Just what he needs to learn about the meaning of violence.

There is a hard-nosed lesson, however, on the limits of love. Spike and Hammer, the heroes, have white skin. Many of the baddies are brown or green. But it's just a game, right?

An especially nice touch are the scantily dressed ladies with the whips. They yelp most satisfyingly when you kick them to oblivion. But what's a little sexism and sadomasochism among eight-year-olds?

After Matthew has gone to bed, Dad and Mom, those pro-peace, pro-feminist, socialist children of the 60s, look at each other over the kitchen table. It's a dull nullity of a look, glazed with poignant helplessness and a tiny bright tint of anger. It's the look reserved for parents who know that TV, peer pressure and our crazed 80s culture have swallowed their child whole. Mom and Dad know how the Trojans felt after Hector died and the Greeks came bellowing out of that infamous horse. Mom and Dad know there's nothing to be done. They hope it'll all pass.

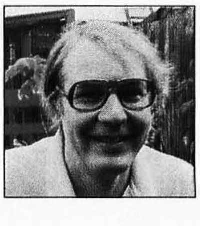
Mom recalls playing guns as a child. Dad remembers games of cowboys and Indians, Allies vs. Germans. He even remembers having his toy soldiers repeatedly machine-gun all his younger female cousin's plastic horses. Where are the dead horses now?

Just games, right? Kids need violent fantasy to work off the impotence they feel as the youngest, smallest members of a family, don't they? It all comes down to the innocent exuberance of childhood, doesn't it?

Mom and Dad aren't violent people. Of course, there's been at least one war, and usually several of them going on somewhere in the world since they were kids. And crimes of violence, including rape, maintain their steady rise. Racism too is alive and well and living just around the corner. But that has nothing to do with childhood games, does it?

Mom and Dad tell Matthew they don't like *Double Dragon*'s violence; yet they let him continue to play. Fortunately, after about six weeks he starts to lose interest. It's a little too easy, even for an eight-year-old.

So what's it all add up to? On a global scale, likely not very much. Probably just a niggling bump on the smoggy, war-torn, unjustly tilted road to the brave new world of the 21st century. ■



***John Sifton** is a freelance writer living in Ottawa, Canada. When his eight-year-old son, Matthew, permits, he uses his 1040ST to write fiction, screenplays, speeches, reports and just about anything else.*

ST-LOG *invites all authors to submit essays for possible use in the* Footnotes *department. Submissions should be between 1,000 and 1,500 words, and may be on any aspect of Atari computing. Any style or type of essay is acceptable—opinion, humor, personal experience—but creativity is a plus. Send your submission to:* Footnotes, *c/o* ST-LOG, *P.O. Box 1413-M.O., Manchester, CT 06040-1413.*

AUTHORIZED SERVICE CENTER FOR ALL ATARI PRODUCTS

MICROTYME

A DIVISION OF MICRO PERIPHERALS, INC.

4049-51 MARSHALL RD. • KETTERING, OHIO 45429

VISA • MasterCard • DISCOVER

HARDWARE

ST'S. . .IN STOCK!!!

Color Monitors	CALL
Mono Monitors	CALL
SF 314 Drive	CALL
GTS 100 Drive	CALL
IB 5¼ Drive	199
Navarone Scanners	CALL

PRINTER'S DEVIL HI-RES GDOS FONT & CLIP ART PACKS IN STOCK! (Great for Desktop Publishing!!)

MODEMS

SX-212 300/1200 bps	CALL
Avatex 1200E	79
Zoom 2400	125

ATARI ST SCANNERS, SOUND & VIDEO DIGITIZERS In Stock!

SUPRA 2400

HAYES® COMPATIBLE

$124.95

6' ST MODEM CABLE..........9
(With Purchase Of Supra)

HARD DISK DRIVES FOR ST'S

SUPRA 30 MB HARD DISK
$649
OTHERS........CALL

ICD 20 MB HARD DISK
$599
OTHERS, INCLUDING TAPE BACK-UP . . . CALL

PRINTERS

PANASONIC	**call for latest**
1180	CALL
1191	CALL
1124	CALL
Panasonic Brand Rib	CALL
STAR	**Call for latest**
NX-1000 . . . . NEW!	CALL
NX-1000 Color	CALL
1000 Ribbon (Blk)	6
1000 Ribbon (Color)	8
OLYMPIA	**simply, the best!**
NLQ modes use 18 x 24 matrix!	
NP-30 . . . . . 130 CPS	199
NP-80s . . . 240 CPS . . . changeable font cards	349
NP-136 . . . . . 15 inch	529

ACCESSORIES

ST Dust Covers	from 8
Mouse Mat	5
Power Strip w/Surge	15
Deluxe Power Strip w/Surge	24
Drive Master	32
Monitor Master	32
Mouse Master	29
EPYX 500 XJ Joystick	15
WICO Ergo Stick Joystick	17
Printer Stand-Heavy Duty	13
Mail Labels 3.5x15/16-500 pk	4
1000 pk	6
Compuserve Starter Kit	24
On-Line Encyclopedia Kit	36
Printer Cable 6'	14
Modem Cable 6'	14

MIDI

Midi Cables 5' to 25'	CALL
Software (Hybrid Arts etc.)	CALL

★ ST SOFTWARE ★

10th Frame Bowling	26
220 ST (Terminal Emulator)	15
3D Breakthru	26
AB Zoo	21
Advanced OCP Art Studio	31
Air Ball	26
Air Ball Construction Set	17
Algebra 1, 2, 3	ea 14
Aliants	19
All About America	41
Alt	21
Alternate Reality-The City	32
Alternate Reality-The Dungeon	32
America Cooks Series	ea 9
Architectural Design	19
Arctic Fox	26
Art Gallery 1, 2, 3	ea 19
Assem Pro	37
Autoduel	24
Award Maker	27
Balance of Power	32
Bally Hoo	27
Barbarian	25
Bards Tale	31
Base Two	45
Basketball (Two on Two)	26
Battle Droidz	25
Battlezone	19
Beyond Zork	34
Biology 1, 2, 3 or 4	ea 14
Bismarck	28
Black Lamp	17
Blockbuster	27
Boulderdash Construction Kit	17
Brataccas	15
Breach	27
Bridge 5.0	24
Bubble Ghost	24
Business Tools	26
Cad 3D (Ver. 2.0)	57
Calamus	175
Calamus Font Editor	62
Captain Blood	31
Carrier Command	29
Certificate Maker	25
Championship Baseball	27
Championship Wrestling	26
Chartpak	34
Chessmaster 2000	29
Chrono Quest	29
Circuit Maker 2	63
Clip Art 1, 2, 3, 4, 5, 6	ea 13
Club Backgammon	23
Color Computer Eyes	169
Colorburst 3000	25
Copyist Level 2	158
Cosmic Relief	26

Cracked	21
Crazy Cars	25
Cyber Control	39
Cyber Paint	48
Cyber VCR	45
Dark Castle	27
Data Manager ST	46
Datatrieve	33
DB Man	149
Death Sword	13
Deep Space	31
Defender of the Crown	31
Degas Elite	38
Demon's Winter	25
Desk Cart	67
Digi Drum	14
Dive Bomber	19
Dr. Drums (DR T)	19
Dr. Keys (DR T)	19
Drafix	129
Dungeon Master 2	18
Dyna Cadd	429
Easy Draw (Regular)	63
Easy Draw W/Supercharger	95
Easy Tools	32
Elite	22
Empire	34
Expert Opinion	72
EZ Score Plus	95
EZ Track Plus	43
F15 Strike Eagle	24
Falcon ST . . . (Low, Low)	CALL
Fast Basic	59
Fast Basic M Compiler	19
Fire and Forget	25
First Cadd 2.0	33
First Letters & Words	25
First Shapes	25
First Word Plus	59
Flash . . . (Great!)	18
Flight Simulator 2	32
Scenery Disks	ea 18
Font Disks (Pub Part) 1-6	ea 20
Fonts and Borders	24
Fontz ST	22
Foundations Waste	26
Fraction Action	24
G + Plus	21
Gateway	31
Gauntlet	31
Genesis (Molecular Modeler)	59
GFA Basic 3.0	59
GFA Basic Book	27
GFA Companion	32
GFA Compiler	38
GFA Draft Plus	49
GFA Quick Reference Manual	12

LARGEST SELECTION IN THE U.S.

★ OVER 1000 TITLES IN STOCK ★

IF YOU *SEE* IT CHEAPER IN ANOTHER AD CALL US ANYWAY. . .WE'RE PROBABLY ALREADY LOWER. . .THESE ADS TAKE 3 MOS. TO GET OUT.

GFA Ray Trace	59
Gladiator	25
Global Commander	28
Gold Of The Realm	24
Gold Rush	25
Goldrunner	26
Goldrunner 2	27
Goldrunner 2 Scenery Disks	ea 7
Gone Fishin'	28
Great Chefs Vol. 1, 2, & 3 Set	39
Gridiron (Football)	19
Gunship	26
Hard Disk Backup	23
Harrier Combat Simulator	34
Heroes Of The Lance	26
Hi Soft Basic Pro	109
High Roller	27
Hollywood Hijinx	19
Human Design Disk	19
Hunt for Red October	32
IB Copy	23
Impossible Mission 2	27
Indiana Jones Temple of Doom	31
Interlink ST	24
International Soccer	25
Into The Eagles Nest	24
Inventory Manager	52
Jet	33
Jinxter	27
Joust	18
Juggler . . . (New Version)	32
Karate Kid 2	27
Karateka	23
KCS Level 2	189
Kid Progs	27

Kinderama	24
Kings Quest 1, 2, 3 or 4	ea 31
Knickerbockers	12
LDW Power Spreadsheet	89
Label Master Elite	26
Lattice C	109
Leaderboard Dual Pack	15
Leatherneck	25
Leisure Suit Larry 1 or 2	24
Lock On	24
Lords of Conquest	13
Lords of the Rising Sun	31
Macro Mouse	25
Magic Sac Plus	89
Magic Sac Roms	CALL
Major Motion	25
Marble Madness	27
Mark Williams C	114
CSD Source Debug	39
Master Cad	119
Match Point	25
Mavis Beacon Teaches Typing	32
Megamax C (Laser C)	118
Mercenary	27
Metro Cross	16
Micro Kitchen Companion	26
Microleague Baseball	36
Microsoft Write	65
Midi Draw	63
Midi Maze	26
Midi Recording Studio (DR T)	36
Missile Command	18
Mixed Up Mother Goose	21
Multi Desk	19
Music Studio 88	34

★ ST SOFTWARE ★

N Vision	29
Neo Desk 2	33
New Tech Coloring Book	15
Night On The Town	22
Ninja	14
Obliterator	25
Ogre	27
Oids	24
Omnires	23
Orbiter	25
Page Stream	115
Paint Pro	33
Paintworks	14
Paperboy	25
Partner Fonts	21
Partner ST	43
PC Ditto 2 . . . . . Low	CALL
Perfect Match	27
Personal Pascal	66
Phantassie 1, 2 or 3	ea 26
Phasar 3.0	58
Pinball Wizard	24
Pirates of the Barbary Coast	17
Planetarium	33
Platoon	25
Police Quest 1, 2	32
Pool of Radiance	25
Prime Time	27
Print Master Plus	26
Prison	25
Pro Copy . . . (Latest Ver.)	28
Publisher ST	79
Q Ball	21
Quantum Paint Box	31
Quink	11
Rastan	25
Read & Rhyme	24
Renegade (Outcast)	14
Road Runner	26
Roadwars	22
Rockford	22
Santa Paravia	19
Scan Art	32
Scruples	29
SDI	13
Shadow	18
Shadowgate	34
Shard of Spring	27
Shuffleboard	19
Silent Service	24
Sinbad	19
Sky Fox	14
Space Quest 1 or 2	ea 31
Space Quest 3	37
Spectrum 512	41
Spelling Bee	19
Spiderman	7

Spy vs Spy 3 (Arctic Antics)	19
ST Disk Drives Inside & Out	18
ST Gem Programmers Ref Man	15
ST Internals Book	15
ST Intro to Midi Book	15
ST Machine Language Book	15
ST Pool	21
ST Talk Pro	17
STAC	44
STOS	39
Star Fleet 1	37
Star Raiders	19
Starglider 2	26
Stellar Crusade	36
Strip Poker 2	25
Sub Battle Simulator	25
Sundog	25
Super Base Professional	174
Super Star Ice Hockey	31
Swift Calc St	46
Take Note	52
Tanglewood	25
Terror Pods	25
Test Drive 1 or 2	24
Test Drive 2 Extra Disks	ea 14
Three Stooges	34
Thunder	26
Time Bandit	24
Top Gun	11
Trailblazer	32
True Basic	52
Tune Smith (DR T)	95
Tune Up	31
Turbo ST	32
TV Sports Football	31
Typhoon Thompson	21
Uninvited	31
Universal Item Selector	12
Universal Military Sim	31
Vampires Empire	20
Vegas Craps	24
Vegas Gambler	23
Video Titleing	22
Vip Professional (Gem)	129
War Ship	38
Wargame Construction Set	22
Winter Challenge	11
Wizards Crown	25
Word Perfect	159
Word Up	47
Word Writer ST	46
World Games	12
World Karate Championship	19
WWF Microleague Wrestling	29
Xevious	19
Zak McKracken	27
Zany Golf	26

HOURS: M-F 9 a.m.-9 p.m. EST
SAT 9 a.m.-5 p.m.

ALL 50 STATES CALL TOLL FREE
1-800-255-5835

For Order Status or
Tech. Info, Call (513) 294-6236

TERMS AND CONDITIONS

• NO EXTRA CHARGES FOR CREDIT CARDS! • Minimum order $15 • C.O.D: Yes, if all Shipping Charges are PRE-PAID • SHIPPING: Hardware, minimum $4; Software and most accessories, minimum $3 • Next day shipment available at extra charge • We ship to Alaska, Hawaii, Puerto Rico (UPS Blue Label Only), APO and FPO • Canadian orders, actual shipping plus 5%, minimum $5 • Ohio residents add 6% sales tax • Please allow 2 weeks for personal or company checks to clear • All defective products require a return authorization number to be accepted for repair or replacement • No free trials or credit • Returns subject to 15% re-stocking charge • Due to changing market conditions, call toll free for latest price and availability of product. FOR YOUR PROTECTION, WE CHECK ALL CREDIT CARD ORDERS FOR FRAUD.

CIRCLE #111 ON READER SERVICE CARD.

COMING
NEXT·MONTH

ANALOG Magazine
COMBINES WITH
ST LOG Magazine
IN AN
ALL NEW PUBLICATION...

LOOK FOR IT!

ST LOG
THE ATARI ST MONTHLY MAGAZINE